SunOS Reference Manual

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Preface

OVERVIEW

A man page is provided for both the naive user, and sophisticated user who is familiar with the SunOS operating system and is in need of on-line information. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

The following contains a brief description of each section in the man pages and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.

- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.

- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.

- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2 of this volume.

- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.

- Section 5 contains miscellaneous documentation such as character set tables, etc.

- Section 6 contains available games and demos.

- Section 7 describes various special files that refer to specific hardware peripherals, and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.

- Section 9 provides reference information needed to write device drivers in the kernel operating systems environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver–Kernel Interface (DKI).

- Section 9E describes the DDI / DKI, DDI-only, and DKI-only entry-point routines a developer may include in a device driver.

- Section 9F describes the kernel functions available for use by device drivers.

- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the intro pages for more information and detail about each section, and **man**(1) for more information about man pages in general.

NAME

This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS

This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full pathname is shown. Literal characters (commands and options) are in **bold** font and variables (arguments, parameters and substitution characters) are in *italic* font. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

- [ ] The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument *must* be specified.
- . . . Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example, ‘*filename* . . .’.
- | Separator. Only one of the arguments separated by this character can be specified at time.
- { } Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL

This section occurs only in subsection 3R to indicate the protocol description file. The protocol specification pathname is always listed in **bold** font.

DESCRIPTION

This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, functions and such, are described under USAGE.

IOCTL

This section appears on pages in Section 7 only. Only the device class which supplies appropriate parameters to the **ioctl**(2) system call is called **ioctl** and generates its own heading. **ioctl** calls for a specific device are listed alphabetically (on the man page for that specific device). **ioctl** calls are used for a particular class of devices all of which have an **io** ending, such as **mtio**(7).

OPTIONS

This lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

OPERANDS

This section lists the command operands and describes how they affect the actions of the command.

OUTPUT

This section describes the output - standard output, standard error, or output files - generated by the command.

RETURN VALUES

If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared as **void** do not return values, so they are not discussed in RETURN VALUES.

ERRORS

On failure, most functions place an error code in the global variable **errno** indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.

USAGE

This section is provided as a *guidance* on use. This section lists special rules, features and commands that require in-depth explanations. The subsections listed below are used to explain built-in functionality:

Commands
Modifiers
Variables
Expressions
Input Grammar

EXAMPLES

This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command line entry and machine response is shown. Whenever an example is given, the prompt is shown as

example%

or if the user must be super-user,

example#

Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS and USAGE sections.

ENVIRONMENT

This section lists any environment variables that the command or function affects, followed by a brief description of the effect.

EXIT STATUS

This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion and values other than zero for various error conditions.

FILES

This section lists all filenames referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.

ATTRIBUTES

This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. (See **attributes**(5) for more information.)

SEE ALSO

This section lists references to other man pages, in-house documentation and outside publications.

DIAGNOSTICS

This section lists diagnostic messages with a brief explanation of the condition causing the error. Messages appear in **bold** font with the exception of variables, which are in *italic* font.

WARNINGS

This section lists warnings about special conditions which could seriously affect your working conditions — this is not a list of diagnostics.

NOTES

This section lists additional information that does not belong anywhere else on the page. It takes the form of an *aside* to the user, covering points of special interest. Critical information is never covered here.

BUGS

This section describes known bugs and wherever possible suggests workarounds.

NAME

Intro, intro – introduction to maintenance commands and application programs

DESCRIPTION

This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.

Because of command restructuring for the Virtual File System architecture, there are several instances of multiple manual pages that begin with the same name. For example, the **mount**, pages – **mount**(1M), **mount_cachefs**(1M), **mount_hsfs**(1M), **mount_nfs**(1M), **mount_tmpfs**(1M), and **mount_ufs**(1M). In each such case the first of the multiple pages describes the syntax and options of the generic command, that is, those options applicable to all FSTypes (file system types). The succeeding pages describe the functionality of the FSType-specific modules of the command. These pages list the command followed by an underscore (_) and the FSType to which they pertain. Note that the administrator should not attempt to call these modules directly. The generic command provides a common interface to all of them. Thus the FSType-specific manual pages should not be viewed as describing distinct commands, but rather as detailing those aspects of a command that are specific to a particular FSType.

COMMAND SYNTAX

Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name [*option*(*s*)] [*cmdarg*(*s*)]

where:

name	The name of an executable file
option	– *noargletter*(*s*) or, – *argletter*< >*optarg* where < > is optional white space
noargletter	A single letter representing an option without an argument
argletter	A single letter representing an option requiring an argument
optarg	Argument (character string) satisfying preceding *argletter*
cmdarg	Pathname (or other command argument) *not* beginning with – or, – by itself indicating the standard input

ATTRIBUTES

See **attributes**(5) for a discussion of the attributes listed in this section.

SEE ALSO

getopt(1), **getopt**(3C), **attributes**(5)

DIAGNOSTICS

Upon termination, each command returns 0 for normal termination and non-zero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously ‘‘exit code,’’ ‘‘exit status,’’ or ‘‘return code,’’ and is described only where special conventions are involved.

NOTES

Unfortunately, not all commands adhere to the standard syntax.

Name	Description
ab2admin(1M)	command line interface for AnswerBook2 administration
accept(1M)	accept or reject print requests
acct(1M)	overview of accounting and miscellaneous accounting commands
acctcms(1M)	command summary from process accounting records
acctcon(1M)	connect-time accounting
acctcon1(1M)	See **acctcon**(1M)
acctcon2(1M)	See **acctcon**(1M)
acctdisk(1M)	See **acct**(1M)
acctdusg(1M)	See **acct**(1M)
acctmerg(1M)	merge or add total accounting files
accton(1M)	See **acct**(1M)
acctprc(1M)	process accounting
acctprc1(1M)	See **acctprc**(1M)
acctprc2(1M)	See **acctprc**(1M)
acctsh(1M)	shell procedures for accounting
acctwtmp(1M)	See **acct**(1M)
adbgen(1M)	generate adb script
addbadsec(1M)	map out defective disk blocks
add_drv(1M)	add a new device driver to the system
add_install_client(1M)	See **install_scripts**(1M)
admintool(1M)	system administration with a graphical user interface
aliasadm(1M)	manipulate the NIS+ aliases map
allocate(1M)	device allocation
arp(1M)	address resolution display and control
aset(1M)	monitors or restricts accesses to system files and directories
aset.restore(1M)	restores system files to their content before ASET is installed
aspppd(1M)	asynchronous PPP link manager
asppls(1M)	See **aspppd**(1M)
audit(1M)	control the behavior of the audit daemon

auditconfig(1M)	configure auditing
auditd(1M)	audit daemon
auditreduce(1M)	merge and select audit records from audit trail files
audit_startup(1M)	audit subsystem initialization script
auditstat(1M)	display kernel audit statistics
audit_warn(1M)	audit daemon warning script
automount(1M)	install automatic mount points
automountd(1M)	autofs mount / unmount daemon
autopush(1M)	configures lists of automatically pushed STREAMS modules
bdconfig(1M)	configures the bd (buttons and dials) stream
boot(1M)	start the system kernel or a standalone program
bootparamd(1M)	See **rpc.bootparamd**(1M)
bsmconv(1M)	enable / disable the Basic Security Module (BSM) on Solaris
bsmunconv(1M)	See **bsmconv**(1M)
cachefslog(1M)	Cache File System logging
cachefspack(1M)	pack files and file systems in the cache
cachefsstat(1M)	Cache File System statistics
cachefswssize(1M)	determine working set size for cachefs
captoinfo(1M)	convert a termcap description into a terminfo description
catman(1M)	create the formatted files for the reference manual
cfsadmin(1M)	administer disk space used for caching file systems with the Cache File-System (CacheFS)
cg14config(1M)	configure the cgfourteen device
chargefee(1M)	See **acctsh**(1M)
check(1M)	See **install_scripts**(1M)
chown(1M)	change owner
chroot(1M)	change root directory for a command
ckpacct(1M)	See **acctsh**(1M)
clear_locks(1M)	clear locks held on behalf of an NFS client
closewtmp(1M)	See **acct**(1M)
clri(1M)	clear inode
comsat(1M)	See **in.comsat**(1M)
conv_lp(1M)	convert LP configuration

conv_lpd(1M)	convert LPD configuration
crash(1M)	examine system images
cron(1M)	clock daemon
dcopy(1M)	See **clri**(1M)
dd(1M)	convert and copy a file
deallocate(1M)	device deallocation
devattr(1M)	lists device attributes
devconfig(1M)	configure device attributes
devfree(1M)	release devices from exclusive use
devinfo(1M)	print device specific information
devlinks(1M)	adds /dev entries for miscellaneous devices and pseudo-devices
devnm(1M)	device name
devreserv(1M)	reserve devices for exclusive use
df(1M)	report number of free disk blocks and files
dfmounts(1M)	display mounted resource information
dfmounts_nfs(1M)	display mounted NFS resource information
dfshares(1M)	list available resources from remote or local systems
dfshares_nfs(1M)	list available NFS resources from remote systems
df_ufs(1M)	report free disk space on ufs file systems
dhcpagent(1M)	daemon for client Dynamic Host Configuration Protocol (DHCP)
dhcpconfig(1M)	DHCP service configuration utility
dhtadm(1M)	DHCP configuration table management utility
disks(1M)	creates /dev entries for hard disks attached to the system
diskscan(1M)	perform surface analysis
dispadmin(1M)	process scheduler administration
dmesg(1M)	collect system diagnostic messages to form error log
dmi_cmd(1M)	DMI command line interface utility
dmiget(1M)	DMI command line retrieval utility
dminfo(1M)	report information about a device entry in a device maps file
dmispd(1M)	Sun Solstice Enterprise DMI Service Provider
dodisk(1M)	See **acctsh**(1M)
domainname(1M)	set or display name of the current domain

drvconfig(1M)	configure the /devices directory
du(1M)	summarize disk usage
edquota(1M)	edit user quotas for ufs file system
eeprom(1M)	EEPROM display and load utility
fdetach(1M)	detach a name from a STREAMS-based file descriptor
fdisk(1M)	create or modify fixed disk partition table
ff(1M)	list file names and statistics for a file system
ffbconfig(1M)	configure the FFB Graphics Accelerator
ff_ufs(1M)	list file names and statistics for a ufs file system
fingerd(1M)	See **in.fingerd**(1M)
firmware(1M)	bootable firmware programs and firmware commands
fmthard(1M)	populate VTOC on hard disks
fncheck(1M)	check for consistency between FNS data and NIS+ data
fncopy(1M)	copy FNS contexts, possibly from one naming service to another naming service
fncreate(1M)	create an FNS context
fncreate_fs(1M)	create FNS file system contexts
fncreate_printer(1M)	create new printers in the FNS namespace
fndestroy(1M)	destroy an FNS context
fnselect(1M)	select a specific naming service to use for the FNS Initial Context
fnsypd(1M)	update FNS context on an NIS master server
format(1M)	disk partitioning and maintenance utility
fsck(1M)	check and repair file systems
fsck_cachefs(1M)	check integrity of data cached with CacheFS
fsck_s5fs(1M)	file system consistency check and interactive repair
fsck_ufs(1M)	file system consistency check and interactive repair
fsdb(1M)	file system debugger
fsdb_ufs(1M)	ufs file system debugger
fsirand(1M)	install random inode generation numbers
fstyp(1M)	determine file system type
ftpd(1M)	See **in.ftpd**(1M)
fusage(1M)	disk access profiler

fuser(1M)	identify processes using a file or file structure
fwtmp(1M)	manipulate connect accounting records
gencc(1M)	create a front-end to the cc command
getdev(1M)	lists devices based on criteria
getdgrp(1M)	lists device groups which contain devices that match criteria
getent(1M)	get entries from administrative database
gettable(1M)	get DoD Internet format host table from a host
getty(1M)	set terminal type, modes, speed, and line discipline
getvol(1M)	verifies device accessibility
groupadd(1M)	add (create) a new group definition on the system
groupdel(1M)	delete a group definition from the system
groupmod(1M)	modify a group definition on the system
grpck(1M)	See **pwck**(1M)
halt(1M)	stop the processor
hostconfig(1M)	configure a system’s host parameters
htable(1M)	convert DoD Internet format host table
id(1M)	return user identity
ifconfig(1M)	configure network interface parameters
in.comsat(1M)	biff server
in.dhcpd(1M)	Dynamic Host Configuration Protocol server
inetd(1M)	Internet services daemon
in.fingerd(1M)	remote user information server
infocmp(1M)	compare or print out terminfo descriptions
in.ftpd(1M)	file transfer protocol server
init(1M)	process control initialization
in.lpd(1M)	BSD print protocol adaptor
in.named(1M)	internet domain name server
in.rarpd(1M)	DARPA Reverse Address Resolution Protocol server
in.rdisc(1M)	network router discovery daemon
in.rexecd(1M)	remote execution server
in.rlogind(1M)	remote login server
in.routed(1M)	network routing daemon
in.rshd(1M)	remote shell server
in.rwhod(1M)	system status server

install(1M)	install commands
installboot(1M)	install bootblocks in a disk partition
installf(1M)	add a file to the software installation database
install_scripts(1M)	scripts used to install the Solaris software
in.talkd(1M)	server for talk program
in.telnetd(1M)	DARPA TELNET protocol server
in.tftpd(1M)	Internet Trivial File Transfer Protocol server
in.tnamed(1M)	DARPA trivial name server
in.uucpd(1M)	UUCP server
iostat(1M)	report I/O statistics
kadb(1M)	a kernel debugger
kdmconfig(1M)	configure or unconfigure keyboard, display, and mouse options for OpenWindows and internationalization
kerbd(1M)	generates and validates Kerberos tickets for kernel RPC
kernel(1M)	UNIX system executable file containing basic operating system services
keyserv(1M)	server for storing private encryption keys
killall(1M)	kill all active processes
labelit(1M)	list or provide labels for file systems
labelit_hsfs(1M)	provide and print labels for hsfs file systems
labelit_ufs(1M)	provide and print labels for ufs file systems
lastlogin(1M)	See **acctsh**(1M)
leoconfig(1M)	initialize the Leo (ZX and TZX) Graphics Accelerator and download microcode
link(1M)	link and unlink files and directories
list_devices(1M)	list allocatable devices
listdgrp(1M)	lists members of a device group
listen(1M)	network listener daemon
lockd(1M)	network lock daemon
lockfs(1M)	change or report file system locks
lockstat(1M)	report kernel lock statistics
logins(1M)	list user and system login information
lpadmin(1M)	configure the LP print service
lpfilter(1M)	administer filters used with the LP print service

lpforms(1M)	administer forms used with the LP print service
lpget(1M)	get printing configuration
lpmove(1M)	move print requests
lpsched(1M)	start and stop the LP print service
lpset(1M)	set printing configuration in /etc/printers.conf or FNS
lpshut(1M)	See **lpsched**(1M)
lpsystem(1M)	register remote systems with the print service
lpusers(1M)	set printing queue priorities
luxadm(1M)	administration program for the PHOTON and SSA subsystems
mail.local(1M)	store mail in a mailbox
makedbm(1M)	make a dbm file, or get a text file from a dbm file
mibiisa(1M)	Sun SNMP Agent
mk(1M)	remake the binary system and commands from source code
mkfifo(1M)	make FIFO special file
mkfile(1M)	create a file
mkfs(1M)	construct a file system
mkfs_ufs(1M)	construct a ufs file system
mknod(1M)	make a special file
modinfo(1M)	display information about loaded kernel modules
modload(1M)	load a kernel module
modunload(1M)	unload a module
monacct(1M)	See **acctsh**(1M)
monitor(1M)	SPARC system PROM monitor
mount(1M)	mount or unmount file systems and remote resources
mountall(1M)	mount, unmount multiple file systems
mount_cachefs(1M)	mount CacheFS file systems
mountd(1M)	server for NFS mount requests and NFS access checks
mount_hsfs(1M)	mount hsfs file systems
mount_nfs(1M)	mount remote NFS resources
mount_pcfs(1M)	mount pcfs file systems
mount_s5fs(1M)	mount s5 file systems

mount_tmpfs(1M)	mount tmpfs file systems
mount_ufs(1M)	mount ufs file systems
mpstat(1M)	report per-processor statistics
mvdir(1M)	move a directory
named(1M)	See **in.named**(1M)
named-xfer(1M)	ancillary agent for inbound zone transfers
ncheck(1M)	generate a list of path names versus i-numbers
ncheck_ufs(1M)	generate pathnames versus i-numbers for ufs file systems
ndd(1M)	get and set driver configuration parameters
netstat(1M)	show network status
newfs(1M)	construct a new UFS file system
newkey(1M)	create a new Diffie-Hellman key pair in the publickey database
nfsd(1M)	NFS daemon
nfsstat(1M)	NFS statistics
nisaddcred(1M)	create NIS+ credentials
nisaddent(1M)	create NIS+ tables from corresponding /etc files or NIS maps
nisbackup(1M)	backup NIS+ directories
nis_cachemgr(1M)	NIS+ utility to cache location information about NIS+ servers
nisclient(1M)	initialize NIS+ credentials for NIS+ principals
nisd(1M)	See **rpc.nisd**(1M)
nisd_resolv(1M)	See **rpc.nisd_resolv**(1M)
nisinit(1M)	NIS+ client and server initialization utility
nislog(1M)	display the contents of the NIS+ transaction log
nispasswdd(1M)	See **rpc.nispasswdd**(1M)
nisping(1M)	send ping to NIS+ servers
nispopulate(1M)	populate the NIS+ tables in a NIS+ domain.
nisprefadm(1M)	NIS+ utility to set server preferences for NIS+ clients
nisrestore(1M)	restore NIS+ directory backup
nisserver(1M)	set up NIS+ servers.
nissetup(1M)	initialize a NIS+ domain
nisshowcache(1M)	NIS+ utility to print out the contents of the shared

	cache file
nisstat(1M)	report NIS+ server statistics
nisupdkeys(1M)	update the public keys in a NIS+ directory object
nlsadmin(1M)	network listener service administration
nscd(1M)	name service cache daemon
nslookup(1M)	query name servers interactively
nstest(1M)	DNS test shell
ntpdate(1M)	set the date and time via NTP
ntpq(1M)	standard Network Time Protocol query program
ntptrace(1M)	trace a chain of NTP hosts back to their master time source
nulladm(1M)	See **acctsh**(1M)
obpsym(1M)	Kernel Symbolic Debugging for OpenBoot Firmware
parse_dynamic_clustertoc(1M)	parse clustertoc file based on dynamic entries
passmgmt(1M)	password files management
patchadd(1M)	apply a patch package to a Solaris 2.x system
patchrm(1M)	remove a Solaris 2.x patch package and restore previously saved files
pbind(1M)	control and query bindings of processes to processors
pcmciad(1M)	PCMCIA user daemon
pfinstall(1M)	tests installation profiles
ping(1M)	send ICMP ECHO_REQUEST packets to network hosts
pkgadd(1M)	transfer software packages to the system
pkgask(1M)	stores answers to a request script
pkgchk(1M)	check package installation accuracy
pkgrm(1M)	remove a package from the system
pmadm(1M)	port monitor administration
pmconfig(1M)	configure the power management system
pntadm(1M)	DHCP network table management utility
ports(1M)	creates /dev entries and inittab entries for serial lines
powerd(1M)	power manager daemon
poweroff(1M)	See **halt**(1M)

praudit(1M)	print contents of an audit trail file
prctmp(1M)	See **acctsh**(1M)
prdaily(1M)	See **acctsh**(1M)
prtacct(1M)	See **acctsh**(1M)
prtconf(1M)	print system configuration
prtdiag(1M)	display system diagnostic information
prtvtoc(1M)	report information about a disk geometry and partitioning
psradm(1M)	set processors on-line or off-line
psrinfo(1M)	displays information about processors
psrset(1M)	creation and management of processor sets
putdev(1M)	edits device table
putdgrp(1M)	edits device group table
pwck(1M)	password / group file checkers
pwconv(1M)	installs and updates /etc/shadow with information from /etc/passwd
quot(1M)	summarize file system ownership
quota(1M)	display a user's ufs file system disk quota and usage
quotacheck(1M)	ufs file system quota consistency checker
quotaoff(1M)	See **quotaon**(1M)
quotaon(1M)	turn ufs file system quotas on and off
rarpd(1M)	See **in.rarpd**(1M)
rdate(1M)	set system date from a remote host
rdisc(1M)	See **in.rdisc**(1M)
reboot(1M)	restart the operating system
reject(1M)	See **accept**(1M)
rem_drv(1M)	remove a device driver from the system
removef(1M)	remove a file from software database
repquota(1M)	summarize quotas for a ufs file system
re-preinstall(1M)	installs the JumpStart software on a system
restricted_shell(1M)	See **rsh**(1M)
rexd(1M)	See **rpc.rexd**(1M)
rexecd(1M)	See **in.rexecd**(1M)
rlogind(1M)	See **in.rlogind**(1M)
rm_install_client(1M)	See **install_scripts**(1M)
rmmount(1M)	removable media mounter that automatically

	mounts a file system on a CD-ROM and floppy
rmt(1M)	remote magtape protocol module
route(1M)	manually manipulate the routing tables
routed(1M)	See **in.routed**(1M)
rpcbind(1M)	universal addresses to RPC program number mapper
rpc.bootparamd(1M)	boot parameter server
rpcinfo(1M)	report RPC information
rpc.nisd(1M)	NIS+ service daemon
rpc.nisd_resolv(1M)	NIS+ service daemon
rpc.nispasswdd(1M)	NIS+ password update daemon
rpc.rexd(1M)	RPC-based remote execution server
rpc.rstatd(1M)	kernel statistics server
rpc.rusersd(1M)	network username server
rpc.rwalld(1M)	network rwall server
rpc.sprayd(1M)	spray server
rpc.yppasswdd(1M)	server for modifying NIS password file
rpc.ypupdated(1M)	server for changing NIS information
rpld(1M)	x86 Network Booting RPL (Remote Program Load) Server
rquotad(1M)	remote quota server
rsh(1M)	restricted shell command interpreter
rshd(1M)	See **in.rshd**(1M)
rstatd(1M)	See **rpc.rstatd**(1M)
rtc(1M)	provide all real-time clock and GMT-lag management
runacct(1M)	run daily accounting
runacct(1M)	See **acctsh**(1M)
rusersd(1M)	See **rpc.rusersd**(1M)
rwall(1M)	write to all users over a network
rwalld(1M)	See **rpc.rwalld**(1M)
rwhod(1M)	See **in.rwhod**(1M)
sa1(1M)	See **sar**(1M)
sa2(1M)	See **sar**(1M)
sac(1M)	service access controller
sacadm(1M)	service access controller administration

sadc(1M)	See **sar**(1M)
sadmind(1M)	distributed system administration daemon
saf(1M)	Service Access Facility
sar(1M)	system activity report package
savecore(1M)	save a core dump of the operating system
sendmail(1M)	send mail over the internet
server_upgrade(1M)	upgrade clients of a heterogeneous OS server
setmnt(1M)	establish mount table
setuname(1M)	changes machine information
setup_install_server(1M)	See **install_scripts**(1M)
share(1M)	make local resource available for mounting by remote systems
shareall(1M)	share, unshare multiple resources
share_nfs(1M)	make local NFS file systems available for mounting by remote systems
showmount(1M)	show all remote mounts
showrev(1M)	show machine and software revision information
shutacct(1M)	See **acctsh**(1M)
shutdown(1M)	shut down system, change system state
snmpdx(1M)	Sun Solstice Enterprise Master Agent
snmpXdmid(1M)	Sun Solstice Enterprise SNMP-DMI mapper subagent
snoop(1M)	capture and inspect network packets
soconfig(1M)	configure transport providers for use by sockets
soladdapp(1M)	add an application to the Solstice application registry
soldelapp(1M)	remove an application from the Solstice application registry
solstice(1M)	access system administration tools with a graphical user interface
spray(1M)	spray packets
sprayd(1M)	See **rpc.sprayd**(1M)
ssaadm(1M)	administration program for the SPARCstorage Array
startup(1M)	See **acctsh**(1M)
statd(1M)	network status monitor
strace(1M)	print STREAMS trace messages

strclean(1M)	STREAMS error logger cleanup program
strerr(1M)	STREAMS error logger daemon
sttydefs(1M)	maintain line settings and hunt sequences for TTY ports
su(1M)	become super user or another user
sulogin(1M)	access single-user mode
suninstall(1M)	install the Solaris environment
swap(1M)	swap administrative interface
swmtool(1M)	install, upgrade, and remove software packages
sxconfig(1M)	configure contiguous memory for the SX video subsystem
sync(1M)	update the super block
syncinit(1M)	set serial line interface operating parameters
syncloop(1M)	synchronous serial loopback test program
syncstat(1M)	report driver statistics from a synchronous serial link
sysdef(1M)	output system definition
sysidconfig(1M)	execute system configuration applications, or define set of system configuration applications
sysidnet(1M)	See **sysidtool**(1M)
sysidnis(1M)	See **sysidtool**(1M)
sysidpm(1M)	See **sysidtool**(1M)
sysidroot(1M)	See **sysidtool**(1M)
sysidsys(1M)	See **sysidtool**(1M)
sysidtool(1M)	system configuration
syslogd(1M)	log system messages
sys-unconfig(1M)	undo a system’s configuration
talkd(1M)	See **in.talkd**(1M)
tapes(1M)	creates /dev entries for tape drives attached to the system
taskstat(1M)	prints ASET tasks status
tcxconfig(1M)	configure the default linearity of the 24-bit TrueColor Visual for OpenWindows on a system with an S24 frame buffer (TCX)
telinit(1M)	See **init**(1M)
telnetd(1M)	See **in.telnetd**(1M)
tftpd(1M)	See **in.tftpd**(1M)

tic(1M)	terminfo compiler
tnamed(1M)	See **in.tnamed**(1M)
ttyadm(1M)	format and output port monitor-specific information
ttymon(1M)	port monitor for terminal ports
tunefs(1M)	tune up an existing file system
turnacct(1M)	See **acctsh**(1M)
uadmin(1M)	administrative control
ufsdump(1M)	incremental file system dump
ufsrestore(1M)	incremental file system restore
umount(1M)	See **mount**(1M)
umountall(1M)	See **mountall**(1M)
unlink(1M)	See **link**(1M)
unshare(1M)	make local resource unavailable for mounting by remote systems
unshareall(1M)	See **shareall**(1M)
unshare_nfs(1M)	make local NFS filesystems unavailable for mounting by remote systems
useradd(1M)	administer a new user login on the system
userdel(1M)	delete a user’s login from the system
usermod(1M)	modify a user’s login information on the system
utmp2wtmp(1M)	See **acct**(1M)
utmpd(1M)	utmp and utmpx monitoring daemon
uucheck(1M)	check the uucp directories and permissions file
uucico(1M)	file transport program for the uucp system
uucleanup(1M)	uucp spool directory clean-up
uucpd(1M)	See **in.uucpd**(1M)
uusched(1M)	the scheduler for the uucp file transport program
uutry(1M)	See **Uutry**(1M)
Uutry(1M)	try to contact remote system with debugging on
uuxqt(1M)	execute remote command requests
vmstat(1M)	report virtual memory statistics
volcopy(1M)	make an image copy of file system
volcopy_ufs(1M)	make an image copy of a ufs file system
vold(1M)	Volume Management daemon to manage CD-ROM and floppy devices

wall(1M)	write to all users
whodo(1M)	who is doing what
wtmpfix(1M)	See **fwtmp**(1M)
xntpd(1M)	Network Time Protocol daemon
ypbind(1M)	NIS binder process
ypinit(1M)	set up NIS client
ypmake(1M)	rebuild NIS database
yppasswdd(1M)	See **rpc.yppasswdd**(1M)
yppoll(1M)	return current version of a NIS map at a NIS server host
yppush(1M)	force propagation of changed NIS map
ypserv(1M)	NIS server and binder processes
ypset(1M)	point ypbind at a particular server
ypstart(1M)	Start and stop NIS services
ypstop(1M)	See **ypstart**(1M)
ypupdated(1M)	See **rpc.ypupdated**(1M)
ypxfr(1M)	transfer NIS map from a NIS server to host
ypxfr_1perday(1M)	See **ypxfr**(1M)
ypxfr_1perhour(1M)	See **ypxfr**(1M)
ypxfr_2perday(1M)	See **ypxfr**(1M)
ypxfrd(1M)	See **ypserv**(1M)
zdump(1M)	time zone dumper
zic(1M)	time zone compiler

NAME

ab2admin – command line interface for AnswerBook2 administration

SYNOPSIS

ab2admin [–**h**] [–**H** *command*] [–**o** *command* [*arguments*]]

DESCRIPTION

ab2admin enables you to administer AnswerBook2 collections and documents on a specified AnswerBook2 server. You can install and uninstall AnswerBook1 and AnswerBook2 collections to a server, scan for locally installed collections and update the server database, and obtain a listing of collections and books.

ab2admin server management functions include: stopping the server, starting the server, restarting the server, turning the server log files on or off, and rotating the log files. You can configure the server to resolve links to books located on other AnswerBook2 servers.

ab2admin can also be used to control server access by adding or deleting users to the pool of administrative users. You can turn on or off the access control

ab2admin can connect to any AnswerBook2 server (local or remote). Certain functions, such as stop, start and restart apply only to the local AnswerBook2 server. If the AnswerBook2 server is protected by a password, then user id and password are required to initiate an administration task.

ab2admin can run interactively by typing **ab2admin** from the command line and then entering commands as prompted; or, it can be executed entirely from the command line using the –**o** option.

OPTIONS

The following options are supported:

–**h** Displays help.

–**H** *command* Displays help for *command.*

–**o** *command* [*arguments*]

access_off [**-m** *server*] [**-p** *server_port_number*]
Turns off the server access log file.

access_on [**-m** *server*] [**-p** *server_port_number*]
Turns on the server access log file.

add_coll -d *path* [**-m** *server*] [**-p** *server_port_number*]
Adds AnswerBook1 or AnswerBook2 collections into the specified AnswerBook2 server database.

add_server -M *additional_server* **-P** *additional_server_port_number* [**-m** *server*] [**-p** *server_port_number*]
Makes additional server known to the specified server.

add_admin -u *user_id* [**-m** *server*] [**-p** *server_port_number*]
Adds user to the authorized list of server administrators.

auth_off [**-m** *server*] [**-p** *server_port_number*]
Turns off the server administration verification.

auth_on [**-m** *server*] [**-p** *server_port_number*]
Turns on the server administration verification.

change_password -u *user_id* [**-m** *server*] [**-p** *server_port_number*]
Changes authorized user's password.

delete_coll -t *collection_title* [**-m** *server*] [**-p** *server_port_number*]
Deletes AnswerBook1 or AnswerBook2 collections from the specified server.

del_server -M *additional_server* **-P** *additional_server_port_number* [**-m** *server*] [**-p** *server_port_number*]
Delete additional server from list of servers known to the specified server.

del_admin -u *user_id* [**-m** *server*] [**-p** *server_port_number*]
Deletes user from the list of authorized server administrators.

error_off [**-m** *server*] [**-p** *server_port_number*]
Turns off the server error log file.

error_on [**-m** *server*] [**-p** *server_port_number*]
Turns on the server error log file.

install -d *path* [**-n** *package*]
Install package on specified server. See **pkgadd**(1M).

list [**-m** *server*] [**-p** *server_port_number*]
Lists AnswerBook1 and AnswerBook2 collections available on the specified server. The listing includes the books contained within collections.

modify_server_name -s *new_server_name* [**-m** *server*] [**-p** *server_port_number*]
Modifies the server's name.

modify_server_port -a *new_server_port_number* [**-m** *server*] [**-p** *server_port_number*]
Modifies the server's port number.

rotate_access[**-m** *server*] [**-p** *server_port_number*] [**-d** *path/filename*]
Saves and resets the server access log file.

rotate_error [**-m** *server*] [**-p** *server_port_number*] [**-d** *path/filename*]
Saves and resets the server error log file.

restart Restarts local AnswerBook2 server. Requires root access.

scan [**-m** *server*] [**-p** *server_port_number*]
Scans for locally installed collections (AnswerBook1 or AnswerBook2) and update the collections on specified server's database.

start Starts local AnswerBook2 server. Requires root access.

stop Stops local AnswerBook2 server. Requires root access.

uninstall [**-n** *package*]
Delete package on specified server. See **pkgadd**(1M).

view_access [**-m** *server*] [**-p** *server_port_number*]
Views the contents of the server access log file.

view_config [**-m** *server*] [**-p** *server_port_number*]
Views the configuration settings of the server.

view_error [**-m** *server*] [**-p** *server_port_number*]
Views the contents of the server error log file.

USAGE

quit Exit interactive mode.

bye Exit interactive mode.

exit Exit interactive mode.

EXAMPLES

To list the collections available on a server named **foo.com**, using port number **8888**:

```
example% ab2admin -o list -m foo.com -p 8888
```

To use **ab2admin** in interactive mode for the same operation as shown above:

```
example% ab2admin
>> list -m foo.com -p 8888
```

To install AnswerBook2 collection via **pkgadd**(1M):

```
example% pkgadd -d package_directory/SUNWabsdk
```

The collection directory structure will be copied into the system (by default) to **/opt/answerbooks/local. . .**

To install AnswerBook2 collection that has beed introduced to the system via **pkgadd**(1M) but did not get updated to the server database: (Note that **-d** *path* must include the **collinfo** file. Refer to “Using AnswerBook2 to View Online Information” in your information library.)

```
ab2admin -o add_coll -d /opt/answerbooks/english/solaris_2.6/SUNWabsdk
```

To inspect how an AnswerBook1 collection is defined:

```
example% cat /opt/SUNWans/ab_cardcatalog
:id=SUNWab_10_4: \
:version=: \
:title=Solaris XGL 3.1 AnswerBook: \
:tocpath=/net/elirium.Eng/export/answerbook/Solaris_2.4/SUNWAxg/toc: \
:pspath=/net/elirium.Eng/export/answerbook/Solaris_2.4/SUNWAxg/ps: \
:indexpath=/net/elirium.Eng/export/answerbook/Solaris_2.4/SUNWAxg/index
```

To install AnswerBook1 collections:

```
ab2admin -o install -d
/opt/answerbooks/english/solaris_2.6/SUNWabe/ab_cardcatalog
```

To inspect how an AnswerBook2 collection is defined:

```
example% cat /opt/answerabooks/english/solaris_2.6/SUNabsd/collinfo
dwCollections {
 coll.45.4 dwCollection
}

dwSetParam coll.45.4 {
 location /opt/answerbooks/english/solaris_2.6/SUNWabsdk
 title "Solaris 2.6 Software Developer AnswerBook Vol 1"
 type EbtCollection
}
```

FILES

/var/log/ab2/catalog/local.socat	Catalog file
/var/log/ab2/catalog/remote.socat	Catalog file
/var/log/ab2/catalog/delegate.socat	Catalog file
/var/log/ab2/catalog/libcat.socat	Catalog file
/var/log/ab2/logs/access_8888.log	Default access log file
/var/log/ab2/logs/errors_8888.log	Default error log file
/usr/lib/ab2/dweb/data/config/ab2_collections.template	AnswerBook2 collection database
/var/log/ab2/catalog/ab1_cardcatalog	AnswerBook1 collection database
/usr/lib/ab2/dweb/data/config/admin_passwd	File containing username:password

ATTRIBUTES

See **attributes**(5) for a discussion of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWab2u

SEE ALSO

pkgadd(1M), **pkgrm**(1M), **attributes**(5)

NAME

accept, reject – accept or reject print requests

SYNOPSIS

accept *destination* ...
reject [**–r** *reason*] *destination* ...

DESCRIPTION

accept allows the queueing of print requests for the named destinations.

reject prevents queueing of print requests for the named destinations.

Use **lpstat –a** to check if destinations are accepting or rejecting print requests.

accept and **request** must be run on the print server; they have no meaning on a client system.

OPTIONS

The following options are supported for **reject**.

–r *reason* Assigns a reason for rejection of print requests for *destination*. Enclose *reason* in quotes if it contains blanks. *reason* is reported by **lpstat –a**. By default, *reason* is **unknown reason** for existing destinations, and **new printer** for destinations added to the system but not yet accepting requests.

OPERANDS

The following operands are supported.

destination The name of the destination accepting or rejecting print requests. Destination specifies the name of a printer or class of printers (see **lpadmin**(1M)). Specify *destination* using atomic name. See **printers.conf**(4) for information regarding the naming conventions for atomic names.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

FILES

/var/spool/lp/* LP print queue.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu
CSI	Enabled (see **NOTES**)

SEE ALSO

enable(1), **lp**(1), **lpstat**(1), **lpadmin**(1M), **lpsched**(1M), **printers.conf**(4), **attributes**(5)

NOTES

accept and **reject** only effect queuing on the print server's spooling system. Requests made from a client system remain queued in the client system's queuing mechanism until they are cancelled or accepted by the print server's spooling system.

accept is CSI-enabled except for the *destinations* name.

NAME

acct, acctdisk, acctdusg, accton, acctwtmp, closewtmp, utmp2wtmp – overview of accounting and miscellaneous accounting commands

SYNOPSIS

/usr/lib/acct/acctdisk

/usr/lib/acct/acctdusg [–**u***filename*] [–**p***filename*]

/usr/lib/acct/accton [*filename*]

/usr/lib/acct/acctwtmp *reason filename*

/usr/lib/acct/closewtmp

/usr/lib/acct/utmp2wtmp

DESCRIPTION

Accounting software is structured as a set of tools (consisting of both C programs and shell procedures) that can be used to build accounting systems. **acctsh**(1M) describes the set of shell procedures built on top of the C programs.

Connect time accounting is handled by various programs that write records into **/var/adm/wtmp**, as described in **utmp**(4). The programs described in **acctcon**(1M) convert this file into session and charging records, which are then summarized by **acctmerg**(1M).

Process accounting is performed by the system kernel. Upon termination of a process, one record per process is written to a file (normally **/var/adm/pacct**). The programs in **acctprc**(1M) summarize this data for charging purposes; **acctcms**(1M) is used to summarize command usage. Current process data may be examined using **acctcom**(1).

Process accounting records and connect time accounting records (or any accounting records in the **tacct** format described in **acct**(4)) can be merged and summarized into total accounting records by **acctmerg** (see **tacct** format in **acct**(4)). **prtacct** (see **acctsh**(1M)) is used to format any or all accounting records.

acctdisk reads lines that contain user ID, login name, and number of disk blocks and converts them to total accounting records that can be merged with other accounting records.

acctdusg reads its standard input (usually from **find / –print**) and computes disk resource consumption (including indirect blocks) by login.

accton without arguments turns process accounting off. If *filename* is given, it must be the name of an existing file, to which the kernel appends process accounting records (see **acct**(2) and **acct**(4)).

acctwtmp writes a **utmp**(4) record to *filename*. The record contains the current time and a string of characters that describe the *reason*. A record type of **ACCOUNTING** is assigned (see **utmp**(4))**.** *reason* must be a string of 11 or fewer characters, numbers, **$**, or spaces. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp  "acctg on" /var/adm/wtmp
acctwtmp  "acctg off" /var/adm/wtmp
```

For each user currently logged on, **closewtmp** puts a false **DEAD_PROCESS** record in the **/var/adm/wtmp** file. **runacct** (see **runacct**(1M)) uses this false **DEAD_PROCESS** record so that the connect accounting procedures can track the time used by users logged on before **runacct** was invoked.

For each user currently logged on, **runacct** uses **utmp2wtmp** to create an entry in the file **/var/adm/wtmp**, created by **runacct**. Entries in **/var/adm/wtmp** enable subsequent invocations of **runacct** to account for connect times of users currently logged in.

OPTIONS

–**u***filename* Records consisting of those filenames for which **acctdusg** charges no one are placed in *filename* (a potential source for finding users trying to avoid disk charges).

–**p***filename* Specify a password file, *filename*. This option is not needed if the password file is **/etc/passwd**.

ENVIRONMENT

If any of the **LC_**∗ variables (**LC_TYPE**, **LC_MESSAGES**, **LC_TIME**, **LC_COLLATE**, **LC_NUMERIC**, and **LC_MONETARY**) (see **environ**(5)) are not set in the environment, the operational behavior of **acct** for each corresponding locale category is determined by the value of the **LANG** environment variable. If **LC_ALL** is set, its contents are used to override both the **LANG** and the other **LC_**∗ variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how **acct** behaves.

LC_CTYPE determines how **acct** handles characters. When **LC_CTYPE** is set to a valid value, **acct** can display and handle text and filenames containing valid characters for that locale. **acct** can display and handle Extended Unix Code (EUC) characters where any character can be 1, 2, or 3 bytes wide. **acct** can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_TIME determines how **acct** handles date and time formats. In the "C" locale, date and time handling follows the U.S. rules.

FILES

/etc/passwd	used for login name to user ID conversions
/usr/lib/acct	holds all accounting commands listed in sub-class 1M of this manual
/var/adm/pacct	current process accounting file
/var/adm/wtmp	login / logoff history file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

acctcom(1), **acctcms**(1M), **acctcon**(1M), **acctmerg**(1M), **acctprc**(1M), **acctsh**(1M), **fwtmp**(1M), **runacct**(1M), **acct**(2), **acct**(4), **passwd**(4), **utmp**(4), **attributes**(5), **environ**(5)

System Administration Guide

NOTES

The **acctdusg** command can process a maximum of

3000 distinct users

during a single invocation of the command. If at some point the actual number of users exceeds the maximum, the command will not succeed.

NAME

acctcms – command summary from process accounting records

SYNOPSIS

/usr/lib/acct/acctcms [**–a** [**–o**] [**–p**]] [**–c**] [**–j**] [**–n**] [**–s**] [**–t**] *filename* . . .

DESCRIPTION

acctcms reads one or more *filename*s, normally in the form described in **acct**(4). It adds all records for processes that executed identically-named commands, sorts them, and writes them to the standard output, normally using an internal summary format.

OPTIONS

–a Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, "hog factor", characters transferred, and blocks read and written, as in **acctcom**(1). Output is normally sorted by total kcore-minutes.

The following options may be used only with the **–a** option:

–o Output a (non-prime) offshift-time-only command summary.

–p Output a prime-time-only command summary.

When **–o** and **–p** are used together, a combination prime-time and non-prime-time report is produced. All the output summaries will be total usage except number of times executed, CPU minutes, and real minutes, which will be split into prime and non-prime.

–c Sort by total CPU time, rather than total kcore-minutes.

–j Combine all commands invoked only once under "***other".

–n Sort by number of command invocations.

–s Any filenames encountered hereafter are already in internal summary format.

–t Process all records as total accounting records. The default internal summary format splits each field into prime and non-prime time parts. This option combines the prime and non-prime time parts into a single field that is the total of both, and provides upward compatibility with old style **acctcms** internal summary format records.

EXAMPLES

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
example% acctcms  filename ... > today
example% cp  total  previoustotal
example% acctcms  –s  today  previoustotal  > total
example% acctcms  –a  –s  today
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWaccu

SEE ALSO

acctcom(1), **acct**(1M), **acctcon**(1M), **acctmerg**(1M), **acctprc**(1M), **acctsh**(1M), **fwtmp**(1M), **runacct**(1M), **acct**(2), **acct**(4), **utmp**(4), **attributes**(5)

NOTES

Unpredictable output results if **–t** is used on new style internal summary format files, or if it is not used with old style internal summary format files.

NAME

acctcon, acctcon1, acctcon2 – connect-time accounting

SYNOPSIS

/usr/lib/acct/acctcon [–**l** *lineuse* [–**o** *reboot*]

/usr/lib/acct/acctcon1 [–**p**] [–**t**] [–**l** *lineuse*] [–**o** *reboot*]

/usr/lib/acct/acctcon2

DESCRIPTION

acctcon converts a sequence of login / logoff records to total accounting records (see the **tacct** format in **acct**(4)). The login / logoff records are read from standard input. The file **/var/adm/wtmp** is usually the source of the login / logoff records, however, because it may contain corrupted records or system date changes, it should first be fixed using **wtmpfix**. The fixed version of file **/var/adm/wtmp** can then be redirected to **acctcon**. The **tacct** records are written to standard output.

acctcon is a combination of the programs **acctcon1** and **acctcon2**. **acctcon1** converts login / logoff records, taken from the fixed **/var/adm/wtmp** file, to ASCII output. **acctcon2** reads the ASCII records produced by **acctcon1** and converts them to **tacct** records. **acctcon1** can be used with the –**l** and –**o** options, described below, as well as with the –**p** and –**t** options.

OPTIONS

–**p** Print input only, showing line name, login name, and time (in both numeric and date / time formats).

–**t** **acctcon1** maintains a list of lines on which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The –**t** flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files.

–**l** *lineuse* *lineuse* is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hangup, termination of **login**(1) and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions. See **init**(1M) and **utmp**(4)**.**

–**o** *reboot* *reboot* is filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

EXAMPLES

The **acctcon** command is typically used as follows:

```
example% acctcon –l lineuse –o reboots < tmpwtmp > ctacct
```

The **acctcon1** and **acctcon2** commands are typically used as follows:

```
example% acctcon1 –l lineuse –o reboots < tmpwtmp | sort +1n +2 > ctmp
example% acctcon2 < ctmp > ctacct
```

FILES

/var/adm/wtmp login / logoff summary

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

acctcom(1), **login**(1), **acct**(1M), **acctcms**(1M), **acctmerg**(1M), **acctprc**(1M), **acctsh**(1M), **fwtmp**(1M), **init**(1M), **runacct**(1M), **acct**(2), **acct**(4), **utmp**(4), **attributes**(5)

System Administration Guide

NOTES

The line usage report is confused by date changes. Use **wtmpfix** (see **fwtmp**(1M)), with the **/var/adm/wtmp** file as an argument, to correct this situation.

The **acctcon**, **acctcon1**, and **acctcon2** commands can process a maximum of

6000 distinct sessions
1000 distinct terminal lines
2000 distinct login names

during a single invocation of any given command. If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

NAME

acctmerg – merge or add total accounting files

SYNOPSIS

/usr/lib/acct/acctmerg [**–a**] [**–i**] [**–p**] [**–t**] [**–u**] [**–v**] [*filename*] . . .

DESCRIPTION

acctmerg reads its standard input and up to nine additional files, all in the **tacct** format (see **acct**(4)) or an ASCII version thereof. It merges these inputs by adding records whose keys (normally user ID and name) are identical, and expects the inputs to be sorted on those keys.

OPTIONS

–a Produce output in ASCII version of **tacct**.

–i Input files are in ASCII version of **tacct**.

–p Print input with no processing.

–t Produce a single record that totals all input.

–u Summarize by user ID, rather than by user ID and name.

–v Produce output in verbose ASCII format, with more precise notation for floating-point numbers.

EXAMPLES

The following sequence is useful for making "repairs" to any file kept in this format:

```
example% acctmerg –v  <filename1 >filename2
```

Edit *filename2* as desired

```
example% acctmerg –i  <filename2 >filename1
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

acctcom(1), **acct**(1M), **acctcms**(1M), **acctcon**(1M), **acctprc**(1M), **acctsh**(1M), **fwtmp**(1M), **runacct**(1M), **acct**(2), **acct**(4), **utmp**(4), **attributes**(5)

System Administration Guide

NAME

acctprc, acctprc1, acctprc2 – process accounting

SYNOPSIS

/usr/lib/acct/acctprc

/usr/lib/acct/acctprc1 [*ctmp*]

/usr/lib/acct/acctprc2

DESCRIPTION

acctprc reads the standard input, in the form described by **acct**(4), and converts it to total accounting records (see the **tacct** record in **acct**(4)). **acctprc** divides CPU time into prime time and non-prime time and determines mean memory size (in memory segment units). **acctprc** then summarizes the **tacct** records, according to user IDs, and adds login names corresponding to the user IDs. The summarized records are then written to the standard output. **acctprc1** reads input in the form described by **acct**(4), adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (tics), non-prime CPU time (tics), and mean memory size (in memory segment units). If *ctmp* is given, it is expected to contain a list of login sessions sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file, just as **acctprc** does. The information in *ctmp* helps it distinguish between different login names sharing the same user ID.

From the standard input, **acctprc2** reads records in the form written by **acctprc1**, summarizes them according to user ID and name, then writes the sorted summaries to the standard output as total accounting records.

EXAMPLES

The **acctprc** command is typically used as shown below:

```
example% acctprc  < /var/adm/pacct  > ptacct
```

The **acctprc1** and **acctprc2s** commands are typically used as shown below:

```
example% acctprc1  ctmp  </var/adm/pacct
example% acctprc2 > ptacct
```

FILES

/etc/passwd system password file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

acctcom(1), **acct**(1M), **acctcms**(1M), **acctcon**(1M), **acctmerg**(1M), **acctsh**(1M), **cron**(1M), **fwtmp**(1M), **runacct**(1M), **acct**(2), **acct**(4), **utmp**(4), **attributes**(5)

NOTES

Although it is possible for **acctprc1** to distinguish among login names that share user IDs for commands run normally, it is difficult to do this for those commands run from **cron**(1M), for example. A more precise conversion can be done using the **acctwtmp** program in **acct**(1M). **acctprc** does not distinguish between users with identical user IDs.

A memory segment of the mean memory size is a unit of measure for the number of bytes in a logical memory segment on a particular processor.

The **acctprc**, **acctprc1**, and **acctprc2** commands can process a maximum of

6000 distinct sessions
1000 distinct terminal lines
2000 distinct login names

during a single invocation of any given command. If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

NAME

acctsh, chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, runacct, shutacct, startup, turnacct – shell procedures for accounting

SYNOPSIS

/usr/lib/acct/chargefee *login-name number*

/usr/lib/acct/ckpacct [*blocks*]

/usr/lib/acct/dodisk [**–o**] [*filename* . . .]

/usr/lib/acct/lastlogin

/usr/lib/acct/monacct *number*

/usr/lib/acct/nulladm *filename* . . .

/usr/lib/acct/prctmp *filename*

/usr/lib/acct/prdaily [**–c**] [**–l**] [*mmdd*]

/usr/lib/acct/prtacct *filename* [”*heading*”]

/usr/lib/acct/runacct [*mmdd*] [*mmdd state*]

/usr/lib/acct/shutacct [”*reason*”]

/usr/lib/acct/startup

/usr/lib/acct/turnacct on| **off**| **switch**

DESCRIPTION

chargefee Command

chargefee can be invoked to charge a *number* of units to *login-name*. A record is written to **/var/adm/fee**, to be merged with other accounting records by **runacct**.

ckpacct Command

ckpacct should be initiated using **cron**(1M) to periodically check the size of **/var/adm/pacct**. If the size exceeds *blocks*, **500** by default, **turnacct** will be invoked with argument **switch**. If the number of free disk blocks in the **/var** file system falls below **500**, **ckpacct** will automatically turn off the collection of process accounting records via the **off** argument to **turnacct**. When at least **500** blocks are restored, the accounting will be activated again on the next invocation of **ckpacct**. This feature is sensitive to the frequency at which **ckpacct** is executed, usually by **cron**.

dodisk Command

dodisk should be invoked by **cron** to perform the disk accounting functions.

lastlogin Command

lastlogin is invoked by **runacct** to update **/var/adm/acct/sum/loginlog**, which shows the last date on which each person logged in.

monacct Command

monacct should be invoked once each month or each accounting period. *number* indicates which month or period it is. If *number* is not given, it defaults to the current month (01–12). This default is useful if **monacct** is to executed using **cron**(1M) on the first day of each month. **monacct** creates summary files in **/var/adm/acct/fiscal** and restarts the summary files in **/var/adm/acct/sum**.

nulladm Command

nulladm creates *filename* with mode 664 and ensures that owner and group are **adm**. It is called by various accounting shell procedures.

prctmp Command

prctmp can be used to print the session record file (normally **/var/adm/acct/nite/ctmp** created by **acctcon1** (see **acctcon**(1M)).

prdaily Command

prdaily is invoked by **runacct** to format a report of the previous day's accounting data. The report resides in **/var/adm/acct/sum/rprt/mmdd** where *mmdd* is the month and day of the report. The current daily accounting reports may be printed by typing **prdaily**. Previous days' accounting reports can be printed by using the *mmdd* option and specifying the exact report date desired.

prtacct Command

prtacct can be used to format and print any total accounting (**tacct**)file.

runacct Command

runacct performs the accumulation of connect, process, fee, and disk accounting on a daily basis. It also creates summaries of command usage. For more information, see **runacct**(1M).

shutacct Command

shutacct is invoked during a system shutdown to turn process accounting off and append a *reason* record to **/var/adm/wtmp**.

startup Command

startup can be invoked when the system is brought to a multi-user state to turn process accounting on.

turnacct Command

turnacct is an interface to **accton** (see **acct**(1M)) to turn process accounting **on** or **off**. The **switch** argument moves the current **/var/adm/pacct** to the next free name in **/var/adm/pacct***incr* (where *incr* is a number starting with **1** and incrementing by one for each additional **pacct** file), then turns accounting back on again. This procedure is called by **ckpacct** and thus can be taken care of by the **cron** and used to keep **pacct** to a reasonable size. **shutacct** uses **turnacct** to stop process accounting. **startup** uses **turnacct** to start process accounting.

OPTIONS

–c This option prints a report of exceptional resource usage by command, and may be used on current day's accounting data only.

–l This option prints a report of exceptional usage by login id for the specified date. Previous daily reports are cleaned up and therefore inaccessible after each invocation of **monacct**.

–o Use **acctdusg** (see **acct**(1M)) to do a slower version of disk accounting by login directory. *filename*s specifies the one or more filesystem names where disk accounting will be done. If *filename*s are used, disk accounting will be done on these filesystems only. If the **–o** option is used, *filename*s should be mount points of mounted filesystems. If the **–o** option is omitted, *filename*s should be the special file names of mountable filesystems.

FILES

/usr/lib/acct	holds all accounting commands listed in section 1M of this manual
/usr/lib/acct/ptecms.awk	contains the limits for exceptional usage by command name
/usr/lib/acct/ptelus.awk	contains the limits for exceptional usage by login ID
/var/adm/acct/fiscal	fiscal reports directory
/var/adm/acct/nite	working directory
/var/adm/acct/sum	summary directory contains information for **monacct**
/var/adm/acct/sum/loginlog	file updated by last login
/var/adm/fee	accumulator for fees
/var/adm/pacct	current file for per-process accounting
/var/adm/pacct*incr*	used if **pacct** gets large and during execution of daily accounting procedure
/var/adm/wtmp	login / logoff summary

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

acctcom(1), **acct**(1M), **acctcms**(1M), **acctcon**(1M), **acctmerg**(1M), **acctprc**(1M), **cron**(1M), **fwtmp**(1M), **runacct**(1M), **acct**(2), **acct**(4), **utmp**(4), **attributes**(5)

NOTES

The **dodisk** command can process a maximum of

3000 distinct users

during a single invocation of the command. If at some point the actual number of users exceeds the maximum, the command will not succeed.

NAME

adbgen – generate adb script

SYNOPSIS

/usr/lib/adb/adbgen *filename***.adb** . . .

DESCRIPTION

adbgen makes it possible to write **adb**(1) scripts that do not contain hard-coded dependencies on structure member offsets. The input to **adbgen** is a file named *filename***.adb** which contains header information, then a null line, then the name of a structure, and finally an **adb** script. **adbgen** only deals with one structure per file; all member names are assumed to be in this structure. The output of **adbgen** is an **adb** script in *filename*. **adbgen** operates by generating a C program which determines structure member offsets and sizes, which in turn generate the **adb** script.

The header lines, up to the null line, are copied verbatim into the generated C program. Typically these are **#include** statements, which include the headers containing the relevant structure declarations.

The **adb** script part may contain any valid **adb** commands (see **adb**(1)), and may also contain **adbgen** requests, each enclosed in **{}**s. Request types are:

- Print a structure member. The request form is **{***member***,***format***}**. *member* is a member name of the *structure* given earlier, and *format* is any valid **adb** format request. For example, to print the **p_pid** field of the *proc* structure as a decimal number, you would write **{p_pid,d}**.
- Reference a structure member. The request form is **{*****member***,***base***}**. *member* is the member name whose value is desired, and *base* is an **adb** register name which contains the base address of the structure. For example, to get the **p_pid** field of the *proc* structure, you would get the *proc* structure address in an **adb** register, for example **<f**, and write **{*p_pid,<f}**.
- Tell **adbgen** that the offset is ok. The request form is **{OFFSETOK}**. This is useful after invoking another **adb** script which moves the **adb** *dot*.
- Get the size of the *structure*. The request form is **{SIZEOF}**. **adbgen** replaces this request with the size of the structure. This is useful in incrementing a pointer to step through an array of structures.
- Calculate an arbitrary C expression. The request form is **{EXPR,** *expression***}**. **adbgen** replaces this request with the value of the expression. This is useful when more than one structure is involved in the script.
- Get the offset to the end of the structure. The request form is **{END}**. This is useful at the end of the structure to get **adb** to align the *dot* for printing the next structure member.

adbgen keeps track of the movement of the **adb** *dot* and generates **adb** code to move forward or backward as necessary before printing any structure member in a script. **adbgen**’s model of the behavior of **adb**’s *dot* is simple: it is assumed that the first line of the script is of the form *struct_address* / *adb text* and that subsequent lines are of the form + / *adb text*. The **adb** *dot* then moves in a sane fashion. **adbgen** does not check the script to ensure that these limitations are met. **adbgen** also checks the size of the structure member against the size of the **adb** format code and warns you if they are not equal.

EXAMPLES

If there were an include file **x.h** which contained

```
struct x {
        char    *x_cp;
        char    x_c;
        int     x_i;
};
```

then an **adbgen** file (call it **script.adb**) to print the file **x.h** would be:

```
#include "x.h"

x
./"x_cp"16t"x_c"8t"x_i"n{x_cp,X}{x_c,C}{x_i,D}
```

After running **adbgen** the output file **script** would contain:

```
16t"x_c"8t"x_i"nXC3+D"" ./"x_cp"16t"x_c"8t"x_i"nXC3+D
```

To invoke the script you would type:

```
example% adb program
x$<script
```

FILES

/usr/platform/*platform-name***/lib/adb/***
platform-specific **adb** scripts for debugging the kernel

/usr/lib/adb/* **adb** scripts for debugging the kernel

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

SEE ALSO

adb(1), **kadb**(1M), **attributes**(5)

DIAGNOSTICS

Warnings are given about structure member sizes not equal to **adb** format items and about badly formatted requests. The C compiler complains if you reference a structure member that does not exist. It also complains about an ampersand before array names; these complaints may be ignored.

NOTES

platform-name can be found using the **–i** option of **uname**(1).

BUGS

adb syntax is ugly; there should be a higher level interface for generating scripts.

Structure members which are bit fields cannot be handled because C will not give the address of a bit field. The address is needed to determine the offset.

NAME

addbadsec – map out defective disk blocks

SYNOPSIS

addbadsec [**–p**] [**–a** *blkno* [*blkno* . . .]] [**–f** *filename*] *raw_device*

DESCRIPTION

addbadsec is used by the system administrator to map out bad disk blocks. Normally, these blocks are identified during surface analysis, but occasionally the disk subsystem reports unrecoverable data errors indicating a bad block. A block number reported in this way can be fed directly into **addbadsec**, and the block will be remapped. **addbadsec** will first attempt hardware remapping. This is supported on SCSI drives and takes place at the disk hardware level. If the taget is an IDE drive, then software remapping is used. In order for software remapping to succeed, the partition must contain an alternate slice and there must be room in this slice to perform the mapping.

–p Causes **addbadsec** to print the current software map. The output shows the defective block and the assigned alternate. This option cannot be used to print the hardware map.

–a Adds the specified blocks to the hardware or software map. If more than one block number is specified, the entire list should be quoted and block numbers should be separated by white space.

–f Adds the specified blocks to the hardware or software map. The bad blocks are listed, one per line, in the specified file.

It should be understood that bad blocks lead to data loss. Remapping a defective block does not repair a damaged file. If a bad block occurs to a disk-resident file system structure such as a superblock, the entire slice might have to be recovered from a backup.

FILES

The raw device should be **/dev/rdsk/c?t?d?p0**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWcsu

SEE ALSO

fdisk(1M), **fmthard**(1M), **diskscan**(1M), **attributes**(5)

NAME

add_drv – add a new device driver to the system

SYNOPSIS

add_drv [–**b** *basedir*] [–**c** *class_name*] [–**i** *'identify_name...'*] [–**m** *'permission','...'*]
[–**n**] [–**f**] [–**v**] *device_driver*

DESCRIPTION

The **add_drv** command is used to inform the system about newly installed device drivers.

Each device on the system has a name associated with it. This name is represented by the **name** property for the device. Similarly, the device may also have a list of driver names associated with it. This list is represented by the **compatible** property for the device.

The system determines which devices will be managed by the driver being added by examining the contents of the **name** property and the **compatible** property (if it exists) on each device. If the value in the **name** property does not match the driver being added, each entry in the **compatible** property is tried, in order, until either a match occurs or there are no more entries in the **compatible** property.

In some cases, adding a new driver may require a reconfiguration boot. See the **NOTES** section.

OPTIONS

–**b** *basedir* Sets the path to the root directory of the diskless client. Used on the server to execute **add_drv** for a diskless client. The client machine must be rebooted to install the driver.

–**c** *class_name* The driver being added to the system exports the class *class_name.*

–**i** *'identify_name'* A white-space separated list of aliases for the driver *device_driver*.

–**m** *'permission'* Specify the file system permissions for device nodes created by the system on behalf of *device_driver*.

–**n** Do not try to load and attach *device_driver*, just modify the system configuration files for the *device_driver*.

–**f** Normally if a reconfiguration boot is required to complete the configuration of the driver into the system, **add_drv** will not add the driver. The force flag forces **add_drv** to add the driver even if a reconfiguration boot is required. See the –**v** flag.

–**v** The verbose flag causes **add_drv** to provide additional information regarding the success or failure of a driver's configuration into the system. See the **EXAMPLES** section.

EXAMPLES

Example 1

The following example adds the **SUNW,example** driver to the system, with an alias name of **SUNW,alias**. It assumes the driver has already been copied to **/usr/kernel/drv**.

```
example# add_drv –m '* 0666 bin bin','a 0644 root sys' \
        –i 'SUNW,alias' SUNW,example
```

Every minor node created by the system for the **SUNW,example** driver will have the permission **0666**, and be owned by user **bin** in the group **bin**, except for the minor device **a**, which will be owned by **root**, group **sys**, and have a permission of **0644**.

Example 2 The following example adds the driver to the client **/export/root/sun1**. The driver is installed and loaded when the client machine, **sun1**, is rebooted. This second example produces the same result as the first, except the changes are on the diskless client, **sun1**, and the client must be rebooted for the driver to be installed.

```
example# add_drv –m '* 0666 bin bin','a 0644 root sys' \
        –i 'SUNW,alias' -b /export/root/sun1 \
        SUNW,example
```

Example 3 The following example illustrates the case where a new driver is added for a device that is already managed by an existing driver. Consider a device that is currently managed by the driver **dumb_framebuffer**. The **name** and **compatible** properties for this device are as follows:

```
name="display"
compatible="whizzy_framebuffer", "dumb_framebuffer"
```

If **add_drv** is used to add the **whizzy_framebuffer** driver, the following will result.

```
example# add_drv whizzy_framebuffer
Error: Could not install driver (whizzy_framebuffer)
Device managed by another driver.
```

If the –**v** flag is specified, the following will result.

```
example# add_drv -v whizzy_framebuffer
Error: Could not install driver (whizzy_framebuffer)
Device managed by another driver.

Driver installation failed because the following
entries in /devices would be affected:

    /devices/iommu@f,e0000000/sbus@f,e0001000/display[:*]
    (Device currently managed by driver "dumb_framebuffer")

The following entries in /dev would be affected:

    /dev/fbs/dumb_framebuffer0
```

If the –**v** and –**f** flags are specified, the driver will be added resulting in the following.

```
example# add_drv -vf whizzy_framebuffer

A reconfiguration boot must be performed to complete the
installation of this driver.

The following entries in /devices will be affected:
        /devices/iommu@f,e0000000/sbus@f,e0001000/display[:*]
        (Device currently managed by driver "dumb_framebuffer"

    The following entries in /dev will be affected:

        /dev/fbs/dumb_framebuffer0
```

The above example is currently only relevant to devices exporting a generic device name.

EXIT STATUS

add_drv returns **0** on success and **1** on failure.

FILES

/kernel/drv	boot device drivers
/usr/kernel/drv	other drivers that could potentially be shared between platforms
/platform/‘uname -i‘/kernel/drv	platform-dependent drivers
/etc/driver_aliases	driver aliases file
/etc/driver_classes	driver classes file
/etc/minor_perm	minor node permissions
/etc/name_to_major	major number binding

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

boot(1M), **devlinks**(1M), **disks**(1M), **drvconfig**(1M), **kernel**(1M), **modinfo**(1M), **ports**(1M), **rem_drv**(1M), **tapes**(1M), **driver.conf**(4), **system**(4), **attributes**(5), **ddi_create_minor_node**(9F)

Writing Device Drivers

NOTES

Aliases may require quoting (with double-quotes) if they contain numbers.

It is possible to add a driver for a device already being managed by a different driver, where the driver being added appears in the device’s **compatible** list before the current driver. In such cases, a reconfiguration boot is required (see **boot**(1M) and **kernel**(1M)). After the reconfiguration boot, device nodes in **/devices**, entries in **/dev**, and references to these files may no longer be valid (see the **–v** flag). If a reconfiguration boot would be required to complete the driver installation, **add_drv** will fail unless the **–f** option is specified. See **Example 3** in the **EXAMPLES** section.

BUGS

add_drv will accept a full pathname for *device_driver*. However, the kernel does not use the full pathname; it only uses the final component and searches the internal driver search path for the driver. This can lead to the kernel loading a different driver than expected.

For this reason, it is **not** recommended that you use **add_drv** with a full pathname. See **kernel**(1M) for more information on the driver search path.

NAME

admintool – system administration with a graphical user interface

SYNOPSIS

/usr/bin/admintool

DESCRIPTION

admintool is a graphical user interface that enables you to accomplish several system administration tasks on a local system. Membership in the sysadmin group (gid 14) is used to restrict access to administrative tasks. Members of the sysadmin group can use **admintool** to create, delete, and modify local system files. Non-members have read-only permissions (where applicable).

Help is available by using the Help button.

Note: **admintool** is not the tool for a distributed environment. It is used for local adminstration.

USAGE

admintool allows you to do the following tasks:

Manage users — Use **admintool** to add, delete, or modify user accounts. **admintool** makes the appropriate changes to the system’s **/etc/passwd** file (see **passwd**(4)).

Manage groups — Use **admintool** to add, delete, or modify groups. **admintool** makes the appropriate changes to the system’s **/etc/group** file (see **group**(4)).

Manage hosts — Use **admintool** to add, delete, or modify hosts. **admintool** makes the appropriate changes to the system’s **/etc/hosts** file (see **hosts**(4)).

Manage printers — Use **admintool** to add or delete access to a printer, or to modify a system’s printer access. **admintool** makes the appropriate changes to the system’s **/etc/lp** directory.

Manage serial port services — Use **admintool** to enable or disable serial port services. **admintool** sets up the software services necessary to use a modem or terminal attached to a system’s serial port.

Manage software — Use **admintool** to add or remove software. **admintool** adds software from a product CD or on a hard disk to an installed system, or removes software from an installed system.

EXIT STATUS

admintool terminates with exit status **0**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap

SEE ALSO

group(4), **hosts**(4), **passwd**(4), **attributes**(5)

Solaris Advanced User's Guide

WARNINGS

If you use **admintool** to add a host, your local system and your site uses a network name service such as NIS or NIS+, **admintool** host operations may not have the desired effect. This is because information in the network name service will take precedence over the information in the local **/etc/hosts** file, which is where **admintool** updates information.

NOTES

admintool modifies files on the local system, i.e., the system on which you are running **admintool**. **admintool** does not modify or update global networked databases such as NIS or NIS+.

NAME

aliasadm – manipulate the NIS+ aliases map

SYNOPSIS

aliasadm –a *alias expansion* [*options comments*] [*optional flags* . . .]
aliasadm –c *alias expansion* [*options comments*] [*optional flags* . . .]
aliasadm –d *alias* [*optional flags* . . .]
aliasadm –e *alias* [*optional flags* . . .]
aliasadm –l *alias* [*optional flags* . . .]
aliasadm –m *alias* [*optional flags* . . .]
aliasadm [–**I**] [–**D** *domainname*] [–**f** *filename*] [–**M** *mapname*]

DESCRIPTION

aliasadm makes changes to the alias map.

The alias map is an NIS+ table object with four columns:

alias The name of the alias as a null terminated string.

expansion The value of the alias as it would appear in a **sendmail /etc/aliases** file.

options A list of options applicable to this alias. The only option currently supported is **CANON**. With this option, if the user has requested an inverse alias lookup, and there is more than one alias with this expansion, this alias is given preference.

comments An arbitrary string containing comments about this alias. The **sendmail**(1M) command reads this map in addition to the NIS aliases map and the local **/etc/aliases** database.

OPTIONS

–a Add an alias.

–c Change an alias.

–d Delete an alias.

–e Edit the alias map.

–I Initialize the NIS+ aliases database.

–l List the alias map.

–m Print or match an alias.

–D *domainname* Edit the map in domain *domainname* instead of the current domain.

–f *filename* When editing or listing the database, use *filename* instead of invoking the editor.

–M *mapname* Edit *mapname* instead of *mail_aliases*.

FILES

/etc/aliases mail aliases for the local host in ASCII format

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWnisu

SEE ALSO

sendmail(1M), **attributes**(5)

NAME

allocate – device allocation

SYNOPSIS

allocate [**–s**] [**–U** *uname*] *device*

allocate [**–s**] [**–U** *uname*] **–g** *dev–type*

allocate [**–s**] [**–U** *uname*] **–F** *device*

DESCRIPTION

allocate manages the ownership of devices through its allocation mechanism. It ensures that each device is used by only one qualified user at a time.

The *device* argument specifies the device to be manipulated. To preserve the integrity of the device's owner, the allocate operation is executed on all the device special files associated with that device.

The argument *dev–type*, is the device type to be operated on. The argument *dev–type*, can only be used with the –**g** option.

The default allocate operation, allocates the device special files associated with *device* to the uid of the current process.

If the –**F** option is specified, the device cleaning program is executed when allocation is performed. This cleaning program is found in **/etc/security/lib.** The name of this program is found in the **device_allocate**(4) entry for the device in the *dev–exec* field.

OPTIONS

–**g** *dev–type*　Allocate a non–allocated device with a device–type matching *dev–type***.**

–**s**　Silent. Suppresses any diagnostic output.

–**F** *device*　Reallocate the device allocated to another user. This option is often used with –**U** to reallocate a specific device to a specific user. Only the super user is permitted to use this option.

–**U** *uname*　Use the user ID *uname* instead of the user ID of the current process when performing the allocate operation. Only the super user is permitted to use this option.

DIAGNOSTICS

allocate returns an nonzero exit status in the event of an error.

FILES

/etc/security/device_allocate
/etc/security/device_maps
/etc/security/dev/∗
/etc/security/lib/∗

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

bsmconv(1M), **device_allocate**(4), **device_maps**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

arp – address resolution display and control

SYNOPSIS

arp *hostname*

arp –a

arp –d *hostname*

arp –f *filename*

arp –s *hostname ether_address* [**temp**] [**pub**] [**trail**]

DESCRIPTION

The **arp** program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (see **arp**(7P)).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

OPTIONS

–a Display all of the current ARP entries. The definition for the flags in the table are:

- **P** Publish; includes IP address for the machine and the addresses that have explicitly been added by the -s option. ARP will respond to ARP requests for this address.
- **S** Static; not learned for the ARP protocol.
- **U** Unresolved; waiting for ARP response.
- **M** Mapping; only used for the multicast entry for **224.0.0.0**

–d Delete an entry for the host called *hostname*. This option may only be used by the super-user.

–f Read the file named *filename* and set multiple entries in the ARP tables. Entries in the file should be of the form

hostname ether_address [**temp**] [**pub**] [**trail**]

(see option **–s** for argument definitions).

–s Create an ARP entry for the host called *hostname* with the Ethernet address *ether_address*. The Ethernet address is given as six hexadecimal bytes separated by colons. The entry will be permanent unless the word **temp** is given in the command. If the word **pub** is given, the entry will be published. For instance, this system will respond to ARP requests for *hostname* even though the hostname is not its own. The word **trail** indicates that trailer encapsulations may be sent to this host. **arp –s** can be used for a limited form of proxy **ARP** when a host on one of the directly attached networks is not physically present on the subnet. Another machine can then be configured to respond to **ARP** requests using **arp –s**. This is useful in certain **SLIP** or **PPP** configurations.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ifconfig(1M), **arp**(7P), **attributes**(5)

NAME

aset – monitors or restricts accesses to system files and directories

SYNOPSIS

aset [**–p**] [**–d** *aset_dir*] [**–l** *sec_level*] [**–n** *user@host*] [**–u** *userlist_file*]

DESCRIPTION

The Automated Security Enhancement Tool (ASET) is a set of administrative utilities that can improve system security by allowing the system administrators to check the settings of system files, including both the attributes (permissions, ownership, etc.) and the contents of the system files. It warns the users of potential security problems and, where appropriate, sets the system files automatically according to the security level specified.

The security level for **aset** can be specified by setting the **–l** command line option or the **ASETSECLEVEL** environment variable to be one of 3 values: **low**, **med**, or **high**. All the functionality operates based on the value of the security level.

At the **low** level, **aset** does not alter any system behavior. It merely performs a number of checks and reports any potential security weaknesses.

At the **med** level, **aset** modifies some of the settings of system files and parameters, thus restricting system access, to reduce the risks from security attacks. Again it will report the security weaknesses and the modifications performed to restrict access. This does not affect the operations of system services. All the system applications and commands will maintain all of their original functionality.

At the **high** level, further restrictions are made to system access, rendering a very defensive system. Security practices which are not normally required are included. Many system files and parameters settings are modified to minimum access permissions. At this level, security is the foremost concern, higher than any other considerations that affect system behavior. The vast majority of system applications and commands will maintain their functionality, although there may be a few that exhibit behaviors that are not familiar in normal system environment.

More exact definitions of these levels (what exactly **aset** will do at each level) can be found in the administrator manual. The **asetenv**(4) file and the *master files* (see **asetmasters**(4)) determine to a large extent what **aset** performs at each level, and can be used by the experienced administrators to redefine the definitions of the levels to suit their particular needs. These files are provided by default to fit most security conscious environments and in most cases provide adequate security safeguards without modification. They are, however, designed in a way that can be easily edited by experienced administrators with specific needs.

aset can be periodically activated at the specified security level with default definitions using the **–p** option. **aset** will be automatically activated at a frequency specified by the administrator starting from a designated future time (see **asetenv**(4)). Without the **–p** option, **aset** will operate only once immediately.

OPTIONS

−p Schedule **aset** to be executed periodically. This adds an entry for **aset** in the **/etc/crontab** file. The **PERIODIC_SCHEDULE** environment variable in the **/usr/aset/asetenv** file is used to define the time for execution. See **crontab**(1) and **asetenv**(4). If a **crontab**(1) entry for **aset** already exists, a warning is produced in the execution log.

−d *aset_dir* Specify a working directory other than **/usr/aset** for ASET. **/usr/aset** is the default working directory. It is where ASET is installed, and is the *root* directory of all ASET utilities and data files. If another directory is to be used as the ASET working directory you can either define it with the −**d** option, or by setting the **ASETDIR** environment variable before invoking **aset**. The command line option, if specified, overwrites the environment variable.

−l *sec_level* Specify a security level (**low**, **med**, or **high**) for **aset** to operate at. The default level is **low**. Each security level is explained in detail above. The level can also be specified by setting the **ASETSECLEVEL** environment variable before invoking **aset**. The command line option, if specified, overwrites the environment variable.

−n *user@host* Notify *user* at machine *host*. Send the output of **aset** to *user* through e-mail. If this option is not specified, the output is sent to the standard output. Note that this is not the reports of ASET, but rather an execution log including error messages if there are any. This output is typically fairly brief. The actual reports of ASET are found in the **/usr/aset/reports/latest** directory. See the −**d** option.

−u *userlist_file* Specify a file containing a list of users. **aset** will perform environment checks (for example, **UMASK** and **PATH** variables) on these users. By default, **aset** only checks for **root**. *userlist_file* is an ASCII text file. Each entry in the file is a line that contains only one user name (login name).

USAGE

The following paragraphs discuss the features provided by ASET. Hereafter, each feature is referred to as a *task*. The first task, **tune**, is intended to be executed only once per installation of ASET. The other tasks are intended to be executed periodically at the specified frequency.

tune Task

This task is used to tighten system file permissions. In standard releases, system files or directories have permissions defined to maximize open information sharing. In a more security conscious environment, the administrator may want to redefine these permission settings to more restrictive values. **aset** allows resetting of these permissions, based on the specified security level. Generally, at the low level the permissions are set to what they should be as released. At the medium level the permissions are tightened to ensure reasonable security that is adequate for most environments. At the high level they are further tightened to very restrictive access. The system files affected and the respective restrictions at different levels are configurable, using the **tune.low**, **tune.med**, and **tune.high** files. See **asetmasters**(4).

cklist Task

System directories that contain relatively static files (that is, their contents and attributes do not change frequently) are examined and compared with a master description file. The **/usr/aset/masters/cklist.***level* files are automatically generated the first time the **cklist** task is executed. See **asetenv**(4). Any discrepancy found is reported. The directories and files are compared based on the following:

- owner and group
- permission bits
- size and checksum (if file)
- number of links
- last modification time

The lists of directories to check are defined in **asetenv**(4), based on the specified security level, and are configurable using the **CKLISTPATH_LOW**, **CKLISTPATH_MED**, and **CKLISTPATH_HIGH** environment variables. Typically, the lower level lists are subsets of the higher level lists.

usrgrp Task

aset checks the consistency and integrity of user accounts and groups as defined in the **passwd** and **group** databases, respectively. Any potential problems are reported. Potential problems for the **passwd** file include:

- **passwd** file entries are not in the correct format.
- User accounts without a password.
- Duplicate user names.
- Duplicate user IDs. Duplicate user IDs are reported unless allowed by the **uid_alias** file. See **asetmasters**(4)).
- Invalid login directories.
- If C2 is enabled, check C2 hidden **passwd** format.

Potential problems for the group file include:

- Group file entries not in the right format.
- Duplicate group names.
- Duplicate group IDs.
- Null group passwords.

aset checks the local **passwd** file. If the **YPCHECK** environment variable is set to **true**, **aset** also checks the NIS **passwd** files. See **asetenv**(4). Problems in the NIS passwd file are only reported and not corrected automatically. The checking is done for all three security levels except where noted.

sysconf Task

aset checks various system configuration tables, most of which are in the **/etc** directory. **aset** checks and makes appropriate corrections for each system table at all three levels except where noted. The following discussion assumes familiarity with the various system tables. See the manual pages for these tables for further details.

The operations for each system table are:

/etc/hosts.equiv
The default file contains a single "+" line, thus making every known host a trusted host, which is not advised for system security. **aset** performs the following operations:

Low Warns the administrators about the "+" line.

Medium
High Warns about and deletes that entry.

/etc/inetd.conf
The following entries for system daemons are checked for possible weaknesses.

tftp(1) does not do any authentication. **aset** ensures that **in.tftpd**(1M) is started in the right directory on the server and is not running on clients. At the low level, it gives warnings if the mentioned condition is not true. At the medium and high levels it gives warnings, and changes (if necessary) the **in.tftpd** entry to include the **–s /tftpboot** option after ensuring the directory **/tftpboot** exists.

ps(1) and **netstat**(1M) provide valuable information to potential system crackers. These are disabled when **aset** is executed at a high security level.

rexd is also known to have poor authentication mechanism. **aset** disables **rexd** for medium and high security levels by commenting out this entry. If **rexd** is activated with the **–s** (secure RPC) option, it is not disabled.

/etc/aliases
The decode alias of UUCP is a potential security weakness. **aset** disables the alias for medium and high security levels by commenting out this entry.

/etc/default/login
The **CONSOLE=** line is checked to allow root login only at a specific terminal depending on the security level:

Low No action taken.

Medium
High Adds the following line to the file:
CONSOLE=/dev/console

/etc/vfstab
aset checks for world-readable or writeable device files for mounted file systems.

/etc/dfs/dfstab
aset checks for file systems that are exported without any restrictions.

/etc/ftpusers
At high security level, **aset** ensures **root** is in **/etc/ftpusers** (create if necessary), thus disallowing **ftp**(1) to be used as root.

/var/adm/utmp
/var/adm/utmpx **aset** makes these files not world-writeable for the high level (some applications may not run properly with this setting.)

/.rhosts The usage of a **.rhosts** file for the entire system is not advised. **aset** gives warnings for the low level and moves it to **/.rhosts.bak** for levels medium and high.

env Task **aset** checks critical environment variables for **root** and users specified with the −**u** *userlist_file* option by parsing the **/.profile**, **/.login**, and **/.cshrc** files. This task checks the **PATH** variable to ensure that it does not contain '**.**' as a directory, which makes an easy target for *trojan horse* attacks. It also checks that the directories in the **PATH** variable are not world-writeable. Furthermore, it checks the **UMASK** variable to ensure files are not created as readable or writeable by world. Any problems found by these checks are reported.

eeprom Task Newer versions of the EEPROM allow specification of a **secure** parameter. See **eeprom**(1M). **aset** recommends that the administrator sets the parameter to **command** for the medium level and to **full** for the high level. It gives warnings if it detects the parameter is not set adequately.

firewall Task At the high security level, **aset** takes proper measures such that the system can be safely used as a firewall in a network. This mainly involves disabling IP packets forwarding and making routing information invisible. Firewalling provides protection against external access to the network.

ENVIRONMENT

ASETDIR Specify ASET's working directory. Defaults to **/usr/aset**.

ASETSECLEVEL Specify ASET's security level. Defaults to **low**.

TASKS Specify the tasks to be executed by **aset**. Defaults to **all tasks**.

FILES

/usr/aset/reports directory of ASET reports

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWast

SEE ALSO

crontab(1), **ftp**(1), **ps**(1), **tftp**(1), **eeprom**(1M), **in.tftpd**(1M), **netstat**(1M), **asetenv**(4), **asetmasters**(4), **attributes**(5)

System Administration Guide

NAME

aset.restore – restores system files to their content before ASET is installed

SYNOPSIS

aset.restore [**–d** *aset_dir*]

DESCRIPTION

aset.restore restores system files that are affected by the Automated Security Enhancement Tool (ASET) to their pre-ASET content. When ASET is executed for the first time, it saves and archives the original system files in the **/usr/aset/archives** directory. The **aset.restore** utility reinstates these files. It also deschedules ASET, if it is currently scheduled for periodic execution. See **asetenv**(4).

Note: If you have made changes to system files after running ASET, these changes are lost when you run **aset.restore**. If you want to be absolutely sure that you keep the existing system state, it is recommended that you back-up your system before using **aset.restore**.

You should use **aset.restore**, under the following circumstances:

> You want to remove ASET permanently and restore the original system (if you want to deactivate ASET, you can remove it from scheduling).
>
> You are unfamiliar with ASET and want to experiment with it. You can use **aset.restore** to restore the original system state.
>
> When some major system functionality is not working properly and you suspect that ASET is causing the problem; you may want to restore the system to see if the problem persists without ASET.

aset.restore requires root privileges to execute.

OPTIONS

–d *aset_dir* Specify the working directory for ASET. By default, this directory is **/usr/aset**. With this option the **archives** directory will be located under *aset_dir*.

FILES

/usr/aset/archives archive of system files prior to executing **aset**

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWast

SEE ALSO

aset(1M), **asetenv**(4), **attributes**(5)

System Administration Guide

NAME

aspppd, aspppls – asynchronous PPP link manager

SYNOPSIS

/usr/sbin/aspppd [**–d** *debug-level*]

/usr/sbin/aspppls

DESCRIPTION

aspppd is the link manager for the asynchronous data link protocol specified in RFC1331, *The Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links*. It is a user level daemon that works in concert with the IP-Dialup driver (**ipdcm**) and PPP streams module (**ppp**(7M)) to provide IP network services over an analog modem using dialed voice grade telephone lines. The link manager automates the process of connecting to a peer (remote) host when PPP service with that host is required. The connection process can be initiated either by sending an IP datagram to a (disconnected) peer host or by receiving a notification that a peer host desires to establish a connection.

aspppls is the *login service* that connects the peer host machine to **aspppd**. **aspppls** is invoked by the serial port monitor when a peer machine logs into a PPP-enabled account. Its purpose is to cause the link manager to accept the incoming call.

OPTIONS

–d *debug-level* The *debug-level* is a number between 0 and 9. Higher numbers give more detailed debugging information. The output is written to the log file **/etc/log/asppp.log**.

USAGE

The link manager is invoked at boot time if the configuration file **/etc/asppp.cf** is present. After parsing the configuration file and building a *path* object for each peer host, it sleeps until (1) an IP datagram is routed to one of the **ipd** or **ipdptp** interfaces (see **ppp**(7M)), or (2) it is notified by the login service that a peer host is attempting to make a connection.

In the first case, it consults the UUCP database, dials the modem, logs into the peer host, establishes the PPP data link, brings up IP, and forwards the IP datagram that initiated the process.

In the second case, the link manager opens the file descriptor supplied by the login service, establishes the PPP data link, and brings up IP.

If the link manager determines that there has been no IP traffic for the period specified by the **inactivity_timeout** keyword, it disconnects the link by bringing down IP and PPP and closing the connection with the peer host.

The link manager can be reinitialized by sending it the -HUP signal (with **kill**(1) for example), which causes it to disconnect all open PPP links and reread the configuration file.

Path

A *path* is an object that contains the state of a connection with a peer host. Information such as system names, interface names, timeout values, and other attributes are kept in the path object. There exists a path for each potential peer host. Paths are defined in the configuration file.

Interfaces

The link manager supports two types of IP layer interfaces; the point-to-multipoint interface (**ipd**) and the point-to-point interface (**ipdptp**) (see **ppp**(7M)).

The point-to-multipoint interface logically connects the host machine to a network containing one or more peer hosts. IP traffic to or from any of the peer hosts is routed through the point-to-multipoint interface. When an **ipd** interface is configured, only one IP address, that of the host, is assigned. In other words, it behaves very similarly to an Ethernet interface, although the broadcast capability is not supported. This type of interface is well suited for a dial in PPP server.

The point-to-point interface logically connects the host machine with one peer host. Only IP traffic to or from the peer host is routed through this interface. When an **ipdptp** interface is configured, two IP addresses are assigned. This type of interface is well suited to support a remote, or nomadic, machine.

An interface must be fully configured and enabled (that is, up) before an IP datagram will be routed to it. It's also true that a point-to-multipoint interface must be fully configured and enabled before the link manager will associate an incoming connection with it. It's not necessary, however, for a point-to-point interface to be configured and enabled before an incoming connection will be assigned to it. A point-to-point interface that is "plumbed", but otherwise not configured or enabled (that is, down), can be used to accept an incoming connection if the path associated with the potential connection contains a dynamic interface specification (for example, **interface ipdptp***). In this case the link manager will select a disabled (down) interface, configure the host and peer addresses, bring it up, and assign it for the duration of the connection.

Routing

Special attention should paid to routing issues that may arise if a host has more than one interface configured and enabled. By definition, a host with more then one enabled interface is a *router*, and the routing daemon (typically **in.routed**) will advertise the routes provided by the PPP interfaces. This is normally acceptable behavior for a dial in server, but can cause network disruptions if not administered properly.

To prevent routing information packets (RIP) from flowing over point-to-point interfaces, specify the **norip** keyword followed by the interface name in the **/etc/gateways** file. These entries, for example, prevent RIP from being sent over ipdptp0 and ipdptp1:

```
norip   ipdptp0
norip   ipdptp1
```

See **in.routed**(1M) for further information.

Authentication

The link manager can be configured to support either the Password Authentication Protocol (PAP) or the Challenge Handshake Authentication Protocol (CHAP) as specified in RFC1334. Both protocols can be configured simultaneously, in which case, CHAP has precedence. A single host may participate as an authenticator (the local host requests that the peer host authenticate itself) or an authenticatee (the local host has been asked by the peer host to authenticate itself) or as both. It is also possible for a host to be an authenticator for one protocol and an authenticatee for the other protocol.

PAP is a simple protocol similar to a standard login/password type of authentication. The PAP authenticator sends a message to its peer requesting that the peer authenticate itself. The peer responds with an authenticate request packet that contains an id and a password (both in plaintext). The id and password are matched against a local copy, and if they match, the connection is established. If they don't match, the connection is dropped.

CHAP does not pass any plaintext authentication data across the link. The CHAP authenticator sends a challenge packet to the peer that contains a random string. The peer then takes the string in the challenge packet and computes a response string that is a function of the challenge string and a shared secret key. The peer then sends a response packet back to the authenticator. The authenticator computes a string based on the original challenge string and the shared secret key and matches that result with the received response. If they match, the connection is established. Otherwise the connection is dropped.

Configuration File

The primary purpose of the **/etc/asppp.cf** configuration file is to define each path used by the link manager to establish and maintain communication with a peer system.

The file consists of a sequence of tokens separated by white space (blanks, tabs, and new lines). There are no record boundaries or any other constraints on the placement of the tokens. If a token begins with a pound sign (#), all characters between the pound sign and the next newline (\n) are ignored (that is, they are treated as a comment). Alphanumeric tokens are case insensitive and are translated by the lexical analyzer into lower case before further processing.

A *string* is a single token that does not contain embedded white space. The standard ANSI C \ escape sequence may be used to embed special characters (see an ANSI C manual for a list of escaped special characters). Use \s for the space character. If a pound sign appears at the beginning of a *string*, it must be escaped (\#) to avoid interpretation as a comment. A NULL (\0) will truncate the *string*.

Groups of tokens are assembled into units known as *paths* (essentially a human-readable form of the path object). A path begins with the keyword **path** and ends at the token found before any subsequent **path** (or **defaults**) keyword or at the last token in the file. The tokens comprising a path are further partitioned into small groups consisting mostly of keyword/value pairs that define the attributes of the current path. If a particular keyword/value pair is not listed for a path, the default value is assumed.

The token sequences that begin with the substrings **ipcp_** or **lcp_** refer to PPP initial configuration options as specified in RFC1332, *The PPP Internet Protocol Control Protocol (IPCP)*. See the RFC for a more complete definition of these options.

The following is an alphabetic list of the token sequences that can be contained in a configuration file. Required sequences are noted.

Keywords

chap_name *string*

One or more octets representing the identification of this host. The name should not be NUL or CR/LF terminated. The name is sent to the authenticator in a response packet. Place this key/value pair in the authenticatee's configuration file.

chap_peer_secret *string*
One or more octets, preferably at least sixteen, that contain the secret key that is used with the challenge value to generate the string to match with the response received from the peer. Place this key/value pair in the authenticator's configuration file.

chap_peer_name *string*
One or more octets representing the identification of the peer transmitting the packet. The name should not be NUL or CR/LF terminated. The name is received from the peer in a response packet. Place this key/value pair in the authenticator's configuration file.

chap_secret *string*
One or more octets, preferably at least sixteen, that contain the secret key that is used with the received challenge value to generate the response sent to the authenticator. Place this key/value pair in the authenticatee's configuration file.

debug_level *number*
number is between 0 and 9. Higher numbers give more detailed debugging information as shown in the table below. The output is written to the **/etc/log/asppp.log** file. The value set by the **debug_level** keyword overrides the **-d** command line option.

level	meaning
0	errors only
1	minimal information
4	some uucp chat-script info
5	all uucp chat-script info
7	maximum uucp info
8	PPP message traces
9	Raw IP packets

defaults Indicates that all following token sequences up the next **path** keyword, or the end of file, set default attributes that affect subsequently defined paths.

default_route
When the IP layer corresponding to the current path is fully operational, add the peer IP address to the route table as the default destination. The route is removed when the IP layer is brought down. Note: the **default_route** keyword is only installed by point-to-point interfaces.

ifconfig *parameters*
(Required) The **ifconfig** keyword and associated *parameters* are passed to the shell for evaluation and execution. It's used to define an interface. See the **ifconfig**(1M) man page for more information.

inactivity_timeout *seconds*
seconds is the maximum number of seconds that the connection associated

with the current path can remain idle before it is terminated. **0** may be specified to indicate no timeout. The default is 120 seconds.

interface (**ipd***n* | **ipdptp***n* | **ipdptp***)
: (Required) Associates a specific point-to-multipoint or point-to-point interface as denoted by the non-negative integer *n* with the current path. The third form, **ipdptp***, indicates that the interface associated with the path is a dynamic interface that will be selected at connect time from a pool of previously configured, inactive (down) point-to-point interfaces.

ipcp_async_map *hex-number*
: Specifies the async control character map for the current path. The *hex-number* is the natural (that is, big endian) form representation of the four octets that comprise the map. The default value is ffffffff.

ipcp_compression (**vj** | **off**)
: Indicates whether IP compression is enabled or not. If enabled (**vj**), the Van Jacobson compression algorithm is used. The default is compression (**vj**).

lcp_compression (**on** | **off**)
: Indicates whether PPP address, control, and protocol field compression is enabled or not. If enabled, both the address and control field compression and the protocol field compression options are set. The default is compression (**on**).

lcp_mru *number*
: *number* specifies a desired maximum receive unit packet size in octets. The default is 1500.

negotiate_address (**on** | **off**)
: Indicates whether or not local IP address assignment is obtained through negotiation and assigned dynamically. If enabled, the local address will be obtained from the remote end of the PPP link. If so obtained, any local address other than 0.0.0.0 can be used to initially configure the interface. The default is to not negotiate (**off**).

pap_id *string*
: One or more octets that represent the name of the host which will be sent to the authenticator. To indicate a zero length string, do not include the keyword. Place this key / value pair in the authenticatee's configuration file.

pap_password *string*
: One or more octets that indicate the password for this host which will be sent to the authenticator. To indicate a zero length string, do not include the keyword. Place this key / value pair in the authenticatee's configuration file.

pap_peer_id *string*
: One or more octets that indicate the name of the peer to be authenticated. To indicate a zero length string, do not include the keyword. Place this key / value pair in the authenticator's configuration file.

pap_peer_password *string*

One or more octets that indicate the password to be used for authentication. To indicate a zero length string, do not include the keyword. Place this key/value pair in the authenticator's configuration file.

path (Required) Indicates that all following token sequences are to be grouped together as attributes of this (current) path. The collection of attributes comprising the current path are terminated by the occurrence of a subsequent **path** or **defaults** keyword or by the end of file.

peer_ip_address *IP-address*

(Required for point-to-multipoint paths) Associates the *IP-address* with the current path. The value is ignored if the path specifies a point-to-point interface. The *IP-address* may be in "dotted decimal", hexadecimal, or symbolic (that is, hostname) format.

peer_system_name *name*

(Required) Associates the peer system *name* with the current path. The name is used to look up modem and peer specific information for outbound connections in the UUCP **/etc/uucp/Systems** file. For incoming connections, the appropriate path is determined by matching *name* with the login name that was used to obtain the connection (that is, an entry in the **/etc/passwd** file specifies *name* in the username field).

require_authentication (**off** | **pap** [**chap**] | **chap** [**pap**])

Indicates that the local host is the authenticator, and that the peer is required to authenticate itself. If either **pap** or **chap** is present, the peer must participate in the authentication protocol or the connection will be terminated. If both **pap** and **chap** are present, then the local host will try to negotiate **chap**, and if that fails, the connection will be terminated. The local host will not try to negotiate **pap**.
The default does not require authentication (**off**).

If **pap** is required, then the **pap_peer_id** and **pap_peer_password** keywords and values should be specified for the associated path. If they are not specified, the corresponding values are set to the null string. If **chap** is required then the **chap_peer_name** and **chap_peer_secret** keywords and values must be specified for the associated path.

version *n* Specifies that the contents of the configuration file correspond to format version *n*. If this keyword is present, it must be the first keyword in the file. If absent, the version is assumed to be 1. This document contains the definition of the version 1 format for the configuration file.

will_do_authentication (**off** | **pap** [**chap**] | **chap** [**pap**])

Indicates that the local host is a potential authenticatee and is willing to participate in the specified authentication protocol. If both **pap** and **chap** are present then the local host is willing to participate in either authentication protocol. The default does not participate in authentication (**off**).

If **pap** is available, then the **pap_id** and **pap_password** keywords and values should be specified for the associated path. If they are not specified, the corresponding values are set to the null string. If **chap** is available then the **chap_name** and **chap_secret** keywords and values must be specified for the associated path.

EXAMPLES

The following are examples of typical configuration files.

Remote Machine

In this example, the remote machine is most likely a nomadic or home machine with a single modem.

```
#
# Dial in to two servers
#
ifconfig ipdptp0 plumb nomad1 dialin1 private up
path
        interface ipdptp0
        peer_system_name Pdialin1
        will_do_authentication pap
        pap_id nomad1
        pap_password secret

ifconfig ipdptp1 plumb nomad1 dialin2 private up
path
        interface ipdptp1
        peer_system_name Pdialin2
        lcp_mru 1006
```

Dial In Servers

This example shows a dial in server supporting a point-to-multipoint interface. There may be several modems attached to this server. Note that the network addressed by the **ipd** interface will be advertised by the router, and all traffic destined for that network will be routed through this host. For that reason, it is not wise to support multiple dial in servers with point-to-multipoint interfaces to the same network.

```
#
# A point-to-multipoint dial in server
#
ifconfig ipd0 plumb dialin1 netmask + up
defaults
        interface ipd0
        inactivity_timeout 900   # 15 minutes
        require_authentication chap pap
        chap_peer_name nomads
path
        peer_system_name Pnomad1
        chap_peer_secret abcd
        pap_peer_id nomad1
        pap_peer_password secret
        peer_ip_address nomad1
```

```
path
        peer_system_name Pnomad2
        chap_peer_secret a\sspace
        peer_ip_address nomad2
path
        peer_system_name Pnomad3
        inactivity_timeout 0                # No timeout for this host
        chap_peer_secret \#123;.
        peer_ip_address nomad3
path
        peer_system_name Pnomad4
        chap_peer_secret My\sSecret#Word
        peer_ip_address nomad4
```

This is another dial in server that supports dynamic point-to-point interfaces. Usually the server has one modem for each interface. One advantage of using dynamic interfaces is that (host) routes will only be advertised when an interface is up. Therefore, multiple dial in servers can be supported.

```
#
# A dynamic point-to-point dial in server
#
ifconfig ipdptp0 plumb dialin2 client1 down
ifconfig ipdptp1 plumb dialin2 client2 down
ifconfig ipdptp2 plumb dialin2 client3 down
defaults
        interface ipdptp*
        inactivity_timeout 900
        debug_level 5

path
        peer_system_name Pnomad1
path
        peer_system_name Pnomad2
path
        peer_system_name Pnomad3
path
        peer_system_name Pnomad4
```

FILES

/etc/asppp.cf	configuration file
/etc/log/asppp.log	message log file
/etc/uucp/Devices	
/etc/uucp/Dialers	
/etc/uucp/Sysfiles	
/etc/uucp/Systems	
/tmp/.asppp.fifo	communication path between **aspppd** and **aspppls**
/usr/sbin/aspppd	link manager

/usr/sbin/aspppls login service

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWapppu

SEE ALSO

kill(1), **ifconfig**(1M), **in.routed**(1M), **attributes**(5), **ppp**(7M)

TCP/IP and Data Communications Administration Guide

NAME

audit – control the behavior of the audit daemon

SYNOPSIS

audit –n | –s | –t

DESCRIPTION

The **audit** command is the general administrator's interface to maintaining the audit trail. The audit daemon may be notified to read the contents of the **audit_control**(4) file and re-initialize the current audit directory to the first directory listed in the **audit_control** file or to open a new audit file in the current audit directory specified in the **audit_control** file as last read by the audit daemon. The audit daemon may also be signaled to close the audit trail and disable auditing.

OPTIONS

–n Signal audit daemon to close the current audit file and open a new audit file in the current audit directory.

–s Signal audit daemon to read audit control file. The audit daemon stores the information internally.

–t Signal audit daemon to close the current audit trail file, disable auditing and die.

DIAGNOSTICS

The **audit** command will exit with **0** upon success and a positive integer upon failure.

FILES

/etc/security/audit_user
/etc/security/audit_control

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

bsmconv(1M), **praudit**(1M), **audit**(2), **audit_control**(4), **audit_user**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

This command does not modify a process's preselection mask. It only affects which audit directories are used for audit data storage and to specify the minimum size free.

NAME

auditconfig – configure auditing

SYNOPSIS

auditconfig *option* **. . .**

DESCRIPTION

auditconfig provides a command line interface to get and set kernel audit parameters.

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

OPTIONS

–chkconf Check the configuration of kernel audit event to class mappings. If the runtime class mask of a kernel audit event does not match the configured class mask, a mismatch is reported.

–conf Configure kernel audit event to class mappings. Runtime class mappings are changed to match those in the audit event to class database file.

–getfsize Return the maximum audit file size in bytes and the current size of the audit file in bytes.

–setfsize *size* Set the maximum size of an audit file to *size* bytes. When the size limit is reached, the audit file is closed and another is started.

–getcond Display the kernel audit condition. The condition displayed is the literal string **auditing** meaning auditing is enabled and turned on (the kernel audit module is constructing and queuing audit records) or **noaudit** meaning auditing is enabled but turned off (the kernel audit module is not constructing and queuing audit records), or **disabled** meaning that the audit module has not been enabled. See **auditon**(2) and **auditd**(1M) for further information.

–setcond[**auditing | noaudit**]
Set the kernel audit condition to the *condition* specified where *condition* is the literal string **auditing** indicating auditing should be enabled or **noaudit** indicating auditing should be disabled.

–getclass *event* Display the preselection mask associated with the specified kernel audit event. *event* is the kernel event number or event name.

–setclass *event audit_flag*[,*audit_flag* . . .]
Map the kernel event *event* to the classes specified by *audit_flags*. *event* is an event number or name. An *audit_flag* is a two character string representing an audit class. See **audit_control**(4) for further information.

–lsevent Display the currently configured (runtime) kernel and user level audit event information.

–getpinfo *pid* Display the audit ID, preselection mask, terminal ID and audit session ID for the specified process.

–setpmask *pid flags*
Set the preselection mask of the specified process. **flags** is the ASCII representation of the flags similar to that in **audit_control**(4).

–setsmask *asid flags*
Set the preselection mask of all processes with the specified audit session ID.

–setumask *auid flags*
Set the preselection mask of all processes with the specified audit ID.

–lspolicy Display the kernel audit policies with a description of each policy.

–getpolicy Display the kernel audit policy.

–setpolicy[+|-]*policy_flag*[,*policy_flag* ...]
Set the kernel audit policy. A policy *policy_flag* is literal strings that denotes an audit policy. A prefix of **+** adds the policies specified to the current audit policies. A prefix of **-** removes the policies specified from the current audit policies. The following are the valid policy flag strings (**auditconfig –lspolicy** also lists the current valid audit policy flag strings):

arge Include the **execv**(2) system call environment arguments to the audit record. This information is not included by default.

argv Include the **execv**(2) system call parameter arguments to the audit record. This information is not included by default.

cnt Do not suspend processes when audit resources are exhausted. Instead, drop audit records and keep a count of the number of records dropped. By default, process are suspended until audit resources become available.

group Include the supplementary group token in audit records. By default, the group token is not included.

path Add secondary path tokens to audit record. These are typically the pathnames of dynamically linked shared libraries or command interpreters for shell scripts. By default, they are not included.

trail Include the trailer token in every audit record. By default, the trailer token is not included.

seq Include the sequence token as part of every audit record. By default, the sequence token is not included. The sequence token attaches a sequence number to every audit record.

EXAMPLES

```
#
# map kernel audit event number 10 to the "fr" audit class
#
% auditconfig –setclass 10 fr
```

```
#
# turn on inclusion of exec arguments in exec audit records
#
% auditconfig –setpolicy +argv
```

EXIT STATUS

0 Successful completion.

1 An error occurred.

FILES

/etc/security/audit_event
/etc/security/audit_class

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

auditd(1M), **bsmconv**(1M), **praudit**(1M), **auditon**(2), **execv**(2), **audit_class**(4), **audit_control**(4), **audit_event**(4), **attributes**(5)

NAME

auditd – audit daemon

SYNOPSIS

/usr/sbin/auditd

DESCRIPTION

The audit daemon controls the generation and location of audit trail files. If auditing is desired, **auditd** reads the **audit_control**(4) file to get a list of directories into which audit files can be written and the percentage limit for how much space to reserve on each filesystem before changing to the next directory.

If **auditd** receives the signal **SIGUSR1**, the current audit file is closed and another is opened. If **SIGHUP** is received, the current audit trail is closed, the **audit_control** file reread, and a new trail is opened. If **SIGTERM** is received, the audit trail is closed and auditing is terminated. The program **audit**(1M) sends these signals and is recommended for this purpose.

Each time the audit daemon opens a new audit trail file, it updates the file **audit_data**(4) to include the correct name.

Auditing Conditions

The audit daemon invokes the program **audit_warn**(1M) under the following conditions with the indicated options:

audit_warn soft *pathname*

The file system upon which *pathname* resides has exceeded the minimum free space limit defined in **audit_control**(4). A new audit trail has been opened on another file system.

audit_warn allsoft

All available file systems have been filled beyond the minimum free space limit. A new audit trail has been opened anyway.

audit_warn hard *pathname*

The file system upon which *pathname* resides has filled or for some reason become unavailable. A new audit trail has been opened on another file system.

audit_warn allhard *count*

All available file systems have been filled or for some reason become unavailable. The audit daemon will repeat this call to **audit_warn** every twenty seconds until space becomes available. *count* is the number of times that **audit_warn** has been called since the problem arose.

audit_warn ebusy

There is already an audit daemon running.

audit_warn tmpfile

The file **/etc/security/audit/audit_tmp** exists, indicating a fatal error.

audit_warn nostart
The internal system audit condition is **AUC_FCHDONE**. Auditing cannot be started without rebooting the system.

audit_warn auditoff
The internal system audit condition has been changed to not be **AUC_AUDITING** by someone other than the audit daemon. This causes the audit daemon to exit.

audit_warn postsigterm
An error occurred during the orderly shutdown of the auditing system.

audit_warn getacdir
There is a problem getting the directory list from **/etc/security/audit/audit_control**.

The audit daemon will hang in a sleep loop until this file is fixed.

FILES

/etc/security/audit/audit_control
/etc/security/audit/audit_data

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

audit(1M), **audit_warn**(1M), **bsmconv**(1M), **praudit**(1M), **auditon**(2), **auditsvc**(2), **audit.log**(4), **audit_control**(4), **audit_data**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

auditreduce – merge and select audit records from audit trail files

SYNOPSIS

auditreduce [*options*] [*audit-trail-file* . . .]

DESCRIPTION

auditreduce allows you to select or merge records from audit trail files. Audit files may be from one or more machines.

The merge function merges together audit records from one or more input audit trail files into a single output file. The records in an audit trail file are assumed to be sorted in chronological order (oldest first) and this order is maintained by **auditreduce** in the output file.

Unless instructed otherwise, **auditreduce** will merge the entire audit trail, which consists of all the audit trail files in the directory structure *audit_root_dir* / ∗ / files (see **audit_control**(4) for details of the structure of the audit root). Unless stated with the **-R** or **-S** option, *audit_root_dir* defaults to **/etc/security/audit.** By using the file selection options it is possible to select some subset of these files, or files from another directory, or files named explicitly on the command line.

The select function allows audit records to be selected on the basis of numerous criteria relating to the record's content (see **audit.log**(4) for details of record content). A record must meet all of the *record-selection-option* criteria to be selected.

Audit Trail Filename Format

Any audit trail file not named on the command line must conform to the audit trail filename format. Files produced by the audit system already have this format. Output file names produced by **auditreduce** are in this format. It is:

*start-time***.***end-time***.***suffix*

where *start-time* is the 14-character timestamp of when the file was opened, *end-time* is the 14-character timestamp of when the file was closed, and *suffix* is the name of the machine which generated the audit trail file, or some other meaningful suffix (e.g., **all**, if the file contains a combined group of records from many machines). The *end-time* may be the literal string **not_terminated**, to indicate that the file is still being written to by the audit system. Timestamps are of the form *yyyymmddhhmmss* (year, month, day, hour, minute, second). The timestamps are in Greenwich Mean Time (GMT).

OPTIONS

File Selection Options

The file selection options indicate which files are to be processed and certain types of special treatment.

–**A** All of the records from the input files will be selected regardless of their timestamp. This option effectively disables the –**a**, –**b**, and –**d** options. This is useful in preventing the loss of records if the –**D** option is used to delete the input files after they are processed. Note, however, that if a record is *not* selected due to another option, then –**A** will not override that.

−C Only process complete files. Files whose filename *end-time* timestamp is **not_terminated** are not processed (such a file is currently being written to by the audit system). This is useful in preventing the loss of records if **−D** is used to delete the input files after they are processed. It does not apply to files specified on the command line.

−D *suffix*
Delete input files after they are processed. The files are only deleted if the entire run is successful. If **auditreduce** detects an error while reading a file, then that file is not deleted. If **−D** is specified, **−A**, **−C** and **−O** are also implied. *suffix* is given to the **−O** option. This helps prevent the loss of audit records by ensuring that all of the records are written, only complete files are processed, and the records are written to a file before being deleted. Note that if both **−D** and **−O** are specified in the command line, the order of specification is significant. The *suffix* associated with the latter specification is in effect.

−M *machine*
Allows selection of records from files with *machine* as the filename suffix. If **−M** is not specified, all files are processed regardless of suffix. **−M** can also be used to allow selection of records from files that contain combined records from many machines and have a common suffix (such as **all**).

−O *suffix*
Direct output stream to a file in the current **audit_root_dir** with the indicated suffix. *suffix* may alternatively contain a full pathname, in which case the last component is taken as the suffix, ahead of which the timestamps will be placed, ahead of which the remainder of the pathname will be placed.

If the **-O** option is not specified, the output is sent to the standard output. When **auditreduce** places timestamps in the filename, it uses the times of the first and last records in the merge as the *start-time* and *end-time*.

−Q Quiet. Suppress notification about errors with input files.

−R *pathname*
Specify the pathname of an alternate audit root directory *audit_root_dir* to be *pathname*. Therefore, rather than using **/etc/security/audit**/∗/files by default, *pathname*/∗/files will be examined instead.

−S *server*
This option causes **auditreduce** to read audit trail files from a specific location (server directory). *server* is normally interpreted as the name of a subdirectory of the audit root, therefore **auditreduce** will look in *audit_root_dir*/*server*/files for the audit trail files.

But if *server* contains any '/' characters, it is the name of a specific directory not necessarily contained in the audit root. In this case, *server*/files will be consulted.

This option allows archived files to be manipulated easily, without requiring that they be physically located in a directory structure like that of **/etc/security/audit**.

–**V** Verbose. Display the name of each file as it is opened, and how many records total were written to the output stream.

Record Selection Options

The record selection options listed below are used to indicate which records are written to the output file produced by **auditreduce.**

Note: Multiple arguments of the same type are not permitted.

–**a** *date-time*
Select records that occurred at or after *date-time*. The *date-time* argument is described under **Option Arguments**, below. *date-time* is in local time. The –**a** and –**b** options can be used together to form a range.

–**b** *date-time*
Select records that occurred before *date-time*.

–**c** *audit-classes*
Select records by audit class. Records with events that are mapped to the audit classes specified by *audit-classes* are selected. Audit class names are defined in **audit_class**(4). The *audit-classes* can be a comma separated list of *audit flags* like those described in **audit_control**(4). Using the *audit flags*, one can select records based upon success and failure criteria.

–**d** *date-time*
Select records that occurred on a specific day (a 24-hour period beginning at 00:00:00 of the day specified and ending at 23:59:59). The day specified is in local time. The time portion of the argument, if supplied, is ignored. Any records with timestamps during that day are selected. If any hours, minutes, or seconds are given in *time*, they are ignored. –**d** can not be used with –**a** or –**b.**

–**e** *effective-user*
Select records with the specified *effective-user*.

–**f** *effective-group*
Select records with the specified *effective-group*.

–**g** *real-group*
Select records with the specified *real-group*.

–**j** *subject-ID*
Select records with the specified *subject-ID* where *subject-ID* is a process ID.

–**m** *event*
Select records with the indicated *event*. The *event* is the literal string or the *event* number.

–**o** *object_type=objectID_value*
Select records by object type. A match occurs when the record contains the information describing the specified *object_type* and the object ID equals the value specified by *objectID_value*. The allowable object types and values are as follows:

file=*pathname*

Select records containing file system objects with the specified *pathname*, where *pathname* is a comma separated list of regular expressions. If a regular expression is preceeded by a tilda (˜), files matching the expression are excluded from the output. For example, the option file="˜/usr/openwin,/usr,/etc" would select all files in **/usr** or **/etc** except those in **/usr/openwin.** The order of the regular expressions is important because auditreduce processes them from left to right, and stops when a file is known to be either selected or excluded. Thus the option file= **/usr**, **/etc**, **˜/usr/openwin** would select all files in **/usr** and all files in **/etc.** Files in **/usr/openwin** are not excluded because the regular expression **/usr** is matched first. Care should be given in surrounding the *pathname* with quotes so as to prevent the shell from expanding any tildas.

msgqid=*ID*
Select records containing message queue objects with the specified *ID* where *ID* is a message queue ID.

pid=*ID* Select records containing process objects with the specified *ID* where *ID* is a process ID. Note: Process are objects when they are receivers of signals.

semid=*ID*
Select records containing semaphore objects with the specified *ID* where *ID* is a semaphore ID.

shmid=*ID*
Select records containing shared memory objects with the specified *ID* where *ID* is a shared memory ID.

sock=*port_number* | *machine*
Select records containing socket objects with the specified *port_number* or the specified *machine* where *machine* is a machine name as defined in **hosts**(4).

–**r** *real-user*
Select records with the specified *real-user.*

–**u** *audit-user*
Select records with the specified *audit-user.*

When one or more *filename* arguments appear on the command line, only the named files are processed. Files specified in this way need not conform to the audit trail filename format. However, –**M**, –**S**, and –**R** may not be used when processing named files. If the *filename* is ‘‘–’’ then the input is taken from the standard input.

Option Arguments

audit-trail-file
An audit trail file as defined in **audit.log**(4). An audit trail file not named on the command line must conform to the audit trail file name format. Audit trail files produced as output of **auditreduce** are in this format as well. The format is:

start-time . *end-time* . *suffix*

start-time is the 14 character time stamp denoting when the file was opened. *end-time* is the 14 character time stamp denoting when the file was closed. *end-time* may also be the literal string **not_terminated**, indicating the file is still be written to by the audit daemon or the file was not closed properly (a system crash or abrupt halt occurred). *suffix* is the name of the machine that generated the audit trail file (or some other meaningful suffix; e.g. **all** would be a good suffix if the audit trail file contains a combined group of records from many machines).

date-time

The *date-time* argument to −**a**, −**b**, and −**d** can be of two forms: An absolute *date-time* takes the form:
yyyymmdd [*hh* [*mm* [*ss*]]]

where *yyyy* specifies a year (with 1970 as the earliest value), *mm* is the month (01-12), *dd* is the day (01-31), *hh* is the hour (00-23), *mm* is the minute (00-59), and *ss* is the second (00-59). The default is 00 for *hh*, *mm* and *ss*.

An offset can be specified as: **+***n***d** | **h** | **m** | **s** where *n* is a number of units, and the tags **d**, **h**, **m**, and **s** stand for days, hours, minutes and seconds, respectively. An offset is relative to the starting time. Thus, this form can only be used with the −**b** option.

event The literal string or ordinal event number as found in **audit_event**(4). If *event* is not found in the **audit_event** file it is considered invalid.

group The literal string or ordinal group ID number as found in **group**(4).
If *group* is not found in the **group** file it is considered invalid. *group* may be negative.

pathname

A regular expression describing a pathname.

user The literal username or ordinal user ID number as found in **passwd**(4). If the username is not found in the **passwd** file it is considered invalid. *user* may be negative.

EXAMPLES

praudit(1M) is available to display audit records in a human-readable form.

This will display the entire audit trail in a human-readable form:

```
% auditreduce | praudit
```

If all the audit trail files are being combined into one large file, then deleting the original files could be desirable to prevent the records from appearing twice:

```
% auditreduce −V −D /etc/security/audit/combined/all
```

This will print what user **milner** did on April 13, 1988. The output will be displayed in a human-readable form to the standard output:

```
% auditreduce −d 19880413 −u milner | praudit
```

The above example may produce a large volume of data if **milner** has been busy. Perhaps looking at only login and logout times would be simpler. The −**c** option will select records from a specified class:

```
% auditreduce –d 19880413 –u milner –c lo | praudit
```

To see **milner**'s login / logout activity for April 13, 14, and 15 the following is used. The results are saved to a file in the current working directory. Note that the name of the output file will have **milnerlo** as the *suffix*, with the appropriate timestamp prefixes. Note that the long form of the name is used for the **–c** option:

```
% auditreduce –a 19880413 –b +3d –u milner –c login_logout –O milnerlo
```

To follow **milner**'s movement about the file system on April 13, 14, and 15 the **chdir** record types could be viewed. Note that in order to get the same time range as the above example we needed to specify the **–b** time as the day after our range. This is because **19880416** defaults to midnight of that day, and records before that fall on **0415**, the end-day of the range.

```
% auditreduce –a 19880413 –b 19880416 –u milner –m AUE_CHDIR | praudit
```

In this example the audit records are being collected in summary form (the login / logout records only). The records are being written to a summary file in a different directory than the normal audit root to prevent the selected records from existing twice in the audit root.

```
% auditreduce –d 19880330 –c lo –O /etc/security/audit_summary/logins
```

If activity for user ID 9944 has been observed, but that user is not known to the system administrator, then the following example will search the entire audit trail for any records generated by that user. **auditreduce** will query the system as to the current validity of ID 9944, and print a warning message if it is not currently active:

```
% auditreduce –O /etc/security/audit_suspect/user9944 –u 9944
```

FILES

/etc/security/audit/*server***/files/*** location of audit trails, when stored

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

bsmconv(1M), **praudit**(1M), **audit.log**(4), **audit_class**(4), **audit_control**(4), **group**(4), **hosts**(4), **passwd**(4), **attributes**(5)

DIAGNOSTICS

auditreduce will print out error messages if there are command line errors and then exit. If there are fatal errors during the run **auditreduce** will print an explanatory message and exit. In this case the output file may be in an inconsistent state (no trailer or partially written record) and **auditreduce** will print a warning message before exiting. Successful invocation returns **0** and unsuccessful invocation returns **1**.

Since **auditreduce** may be processing a large number of input files, it is possible that the machine-wide limit on open files will be exceeded. If this happens, **auditreduce** will print a message to that effect, give information on how many file there are, and exit.

If **auditreduce** prints a record's timestamp in a diagnostic message, that time is in local time. However, when filenames are displayed, their timestamps are in GMT.

BUGS

Conjunction, disjunction, negation, and grouping of record selection options should be allowed.

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

audit_startup – audit subsystem initialization script

SYNOPSIS

/etc/security/audit_startup

DESCRIPTION

The **audit_startup** script is used to initialize the audit subsystem before the audit deamon is started. This script is configurable by the system administrator, and currently consists of a series of **auditconfig**(1M) commands to set the system default policy, and download the initial event to class mapping.

SEE ALSO

auditconfig(1M), **auditd**(1M), **bsmconv**(1M), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

auditstat – display kernel audit statistics

SYNOPSIS

auditstat [**-c** *count*] [**-h** *numlines*] [**-i** *interval*] [**-n**] [**-v**]

DESCRIPTION

auditstat displays kernel audit statistics. The fields displayed are as follows:

aud
The total number of audit records processed by the **audit**(2) system call.

ctl
This field is obsolete.

drop
The total number of audit records that have been dropped. Records are dropped according to the kernel audit policy. See **auditon**(2), AUDIT_CNT policy for details.

enq
The total number of audit records put on the kernel audit queue.

gen
The total number of audit records that have been constructed (not the number written).

kern
The total number of audit records produced by user processes (as a result of system calls).

mem
The total number of Kbytes of memory currently in use by the kernel audit module.

nona
The total number of non-attributable audit records that have been constructed. These are audit records that are not attributable to any particular user.

rblk
The total number of times that **auditsvc**(2) has blocked waiting to process audit data.

tot
The total number of Kbytes of audit data written to the audit trail.

wblk
The total number of times that user processes blocked on the audit queue at the high water mark.

wrtn
The total number of audit records written. The difference between **enq** and **wrtn** is the number of outstanding audit records on the audit queue that have not been written.

OPTIONS

-c *count*
Display the statistics a total of *count* times. If *count* is equal to zero, statistics are displayed indefinitely. A time interval must be specified.

-h *numlines*

Display a header for every *numlines* of statistics printed. The default is to display the header every 20 lines. If *numlines* is equal to zero, the header is never displayed.

-i *interval*

Display the statistics every *interval* where *interval* is the number of seconds to sleep between each collection.

-n

Display the number of kernel audit events currently configured.

-v

Display the version number of the kernel audit module software.

EXIT STATUS

auditstat returns 0 upon success and 1 upon failure.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

auditconfig(1M), **praudit**(1M), **bsmconv**(1M), **audit**(2), **auditon**(2), **auditsvc**(2), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

audit_warn – audit daemon warning script

SYNOPSIS

/etc/security/audit_warn [*option* [*arguments*]]

DESCRIPTION

The **audit_warn** script processes warning or error messages from the audit daemon. When a problem is encountered, the audit daemon, **auditd**(1M) calls **audit_warn** with the appropriate arguments. The *option* argument specifies the error type.

The system administrator can specify a list of mail recipients to be notified when an audit_warn situation arises by defining a mail alias called *audit_warn* in **aliases**(4). The users that make up the *audit_warn* alias are typically the **audit** and **root** users.

OPTIONS

allhard *count*
: Indicates that the hard limit for all filesystems has been exceeded *count* times. The default action for this option is to send mail to the *audit_warn* alias only if the *count* is **1**, and to write a message to the machine console every time. It is recommended that mail *not* be sent every time as this could result in a the saturation of the file system that contains the mail spool directory.

allsoft
: Indicates that the soft limit for all filesystems has been exceeded. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console.

auditoff
: Indicates that someone other than the audit daemon changed the system audit state to something other than **AUC_AUDITING**. The audit daemon will have exited in this case. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console.

ebusy
: Indicates that the audit daemon is already running. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console.

getacdir *count*
: Indicates that there is a problem getting the directory list from **audit_control**(4). The audit daemon will hang in a sleep loop until the file is fixed. The default action for this option is to send mail to the *audit_warn* alias only if *count* is **1**, and to write a message to the machine console every time. It is recommended that mail *not* be sent every time as this could result in a the saturation of the file system that contains the mail spool directory.

hard *filename*

Indicates that the hard limit for the file has been exceeded. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console.

nostart

Indicates that auditing could not be started. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console. Some administrators may prefer to modify *audit_warn* to reboot the system when this error occurs.

postsigterm

Indicates that an error occurred during the orderly shutdown of the audit daemon. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console.

soft *filename*

Indicates that the soft limit for *filename* has been exceeded. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console.

tmpfile

Indicates that the temporary audit file already exists indicating a fatal error. The default action for this option is to send mail to the *audit_warn* alias and to write a message to the machine console.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

audit(1M), **auditd**(1M), **bsmconv**(1M), **aliases**(4), **audit.log**(4), **audit_control**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

automount – install automatic mount points

SYNOPSIS

/usr/sbin/automount [**–t** *duration*] [**-v**]

DESCRIPTION

The **automount** utility installs **autofs** mount points and associates an **automount** map with each mount point. The **autofs** file system monitors attempts to access directories within it and notifies the **automountd**(1M) daemon. The daemon uses the map to locate a file system, which it then mounts at the point of reference within the **autofs** file system. You can assign a map to an **autofs** mount using an entry in the **/etc/auto_master** map or a direct map.

If the file system is not accessed within an appropriate interval (five minutes by default), the **automountd** daemon unmounts the file system.

The file **/etc/auto_master** determines the locations of all **autofs** mount points. By default, this file contains four entries:

```
# Master map for automounter
#
+auto_master
/net          -hosts        -nosuid
/home         auto_home
/xfn          -xfn
```

The **+auto_master** entry is a reference to an external NIS or NIS+ master map. If one exists, then its entries are read as if they occurred in place of the **+auto_master** entry. The remaining entries in the master file specify a directory on which an **autofs** mount will be made followed by the automounter map to be associated with it. Optional mount options may be supplied as an optional third field in the each entry. These options are used for any entries in the map that do not specify mount options explicitly. The **automount** command is usually run without arguments. It compares the entries **/etc/auto_master** with the current list of **autofs** mounts in **/etc/mnttab** and adds, removes or updates **autofs** mounts to bring the **/etc/mnttab** up to date with the **/etc/auto_master**. At boot time it installs all **autofs** mounts from the master map. Subsequently, it may be run to install **autofs** mounts for new entries in the master map or n direct map, or to perform unmounts for entries that have been removed from these maps.

OPTIONS

–t *duration* Specify a *duration*, in seconds, that a file system is to remain mounted when not in use. The default is 5 minutes.

–v Verbose mode. Notify of **autofs** mounts, unmounts or other non-essential information.

USAGE

Map Entry Format

A simple map entry (mapping) takes the form:

key [*–mount-options*] *location* . . .

where **key** is the full pathname of the directory to mount when used in a direct map, or the simple name of a subdirectory in an indirect map. *mount-options* is a comma-separated list of **mount** options, and *location* specifies a file system from which the directory may be mounted. In the case of a simple NFS mount, the options that can be used are as specified in **mount_nfs**(1M), and *location* takes the form:

*host***:***pathname*

host is the name of the host from which to mount the file system, and *pathname* is the absolute pathname of the directory to mount.

Options to other file systems are documented on the other **mount_**∗ reference manual pages, for example, **mount_cachefs**(1M).

Replicated File Systems

Multiple *location* fields can be specified for replicated NFS file systems, in which case **automount** and the kernel will each try to use that information to increase availability. If the read-only flag is set in the map entry, **automount** mounts a list of locations that the kernel may use, sorted by several criteria. When a server does not respond, the kernel will switch to an alternate server. The sort ordering of **automount** is used to determine how the next server is chosen. If the read−only flag is not set, **automount** will mount the best single location, chosen by the same sort ordering, and new servers will only be chosen when an unmount has been possible, and a remount is done. Servers on the same local subnet are given the strongest preference, and servers on the local net are given the second strongest preference. Among servers equally far away, response times will determine the order if no weighting factors (see below) are used.

If the list includes server locations using both the NFS Version 2 Protocol and the NFS Version 3 Protocol, **automount** will choose only a subset of the server locations on the list, so that all entries will be the same protocol. It will choose servers with the NFS Version 3 Protocol so long as an NFS Version 2 Protocol server on a local subnet will not be ignored. See the *NFS Administration Guide* for additional details.

If each *location* in the list shares the same *pathname* then a single *location* may be used with a comma-separated list of hostnames:

*hostname***,***hostname*...**:***pathname*

Requests for a server may be weighted, with the weighting factor appended to the server name as an integer in parentheses. Servers without a weighting are assumed to have a value of zero (most likely to be selected). Progressively higher values decrease the chance of being selected. In the example,

```
man  −ro  alpha,bravo,charlie(1),delta(4):/usr/man
```

hosts **alpha** and **bravo** have the highest priority; host **delta**, the lowest.

Note: Server proximity takes priority in the selection process. In the example above, if the server **delta** is on the same network segment as the client, but the others are on different network segments, then **delta** will be selected — the weighting value is ignored. The weighting has effect only when selecting between servers with the same network proximity.

In cases where each server has a different export point, you can still apply the weighting. For example:

```
man –ro alpha:/usr/man  bravo,charlie(1):/usr/share/man  delta(3):/export/man
```

A mapping can be continued across input lines by escaping the NEWLINE with a ‘ \ ’ (backslash). Comments begin with a ’#’ (number sign) and end at the subsequent NEWLINE.

Map Key Substitution

The ’&’ (ampersand) character is expanded to the value of the **key** field for the entry in which it occurs. In this case:

```
jane          sparcserver:/home/&
```

the **&** expands to **jane**.

Wildcard Key

The ’*’ (asterisk) character, when supplied as the **key** field, is recognized as the catch-all entry. Such an entry will match any key not previously matched. For instance, if the following entry appeared in the indirect map for **/config**:

```
*             &:/export/config/&
```

this would allow automatic mounts in **/config** of any remote file system whose location could be specified as:

hostname:**/export/config/***hostname*

Variable Substitution

Client specific variables can be used within an **automount** map. For instance, if **$HOST** appeared within a map, **automount** would expand it to its current value for the client’s host name. Supported variables are:

ARCH	Derived from the output of **uname –m**.	The architecture name. For example "sun4"
CPU	The output of **uname –p**.	The processor type. For example "sparc"
HOST	The output of **uname –n**.	The host name. For example "biggles"
OSNAME	The output of **uname –s**.	The OS name. For example "SunOS"
OSREL	The output of **uname –r**.	The OS release name. For example "5.3"
OSVERS	The output of **uname –v**.	The OS version. For example "beta1.0"

If a reference needs to be protected from affixed characters, you can surround the variable name with ’{ }’ (curly braces).

Multiple Mounts

A multiple mount entry takes the form:

key [−*mount-options*] [[*mountpoint*] [−*mount-options*] *location*. . .] . . .

The initial **/**[*mountpoint*] is optional for the first mount and mandatory for all subsequent mounts. The optional *mountpoint* is taken as a pathname relative to the directory named by **key**. If *mountpoint* is omitted in the first occurrence, a *mountpoint* of
/ (root) is implied.

Given an entry in the indirect map for **/src**:

```
beta –ro \
        /                svr1,svr2:/export/src/beta    \
        /1.0             svr1,svr2:/export/src/beta/1.0    \
        /1.0/man         svr1,svr2:/export/src/beta/1.0/man
```

All offsets must exist on the server under **beta**. **automount** will automatically mount **/src/beta**, **/src/beta/1.0**, and **/src/beta/1.0/man**, as needed, from either **svr1** or **svr2**, whichever host is nearest and responds first.

Other File System Types

The automounter assumes NFS mounts as a default file system type. Other file system types can be described using the **fstype** mount option. Other mount options specific to this file system type can be combined with the **fstype** option. The location field must contain information specific to the file system type. If the location field begins with a slash, a colon character must be prepended, for instance, to mount a CD file system:

```
cdrom  -fstype=hsfs,ro      :/dev/sr0
```

or to perform an **autofs** mount:

```
src    -fstype=autofs       auto_src
```

Note: Use this procedure only if you are not using Volume Manager.

Mounts using CacheFS are most useful when applied to an entire map as map defaults. The following entry in the master map describes cached home directory mounts. It assumes the default location of the cache directory, **/cache**.

```
/home  auto_home            -fstype=cachefs,backfstype=nfs
```

See the **NOTES** section for information on option inheritance.

Indirect Maps

An indirect map allows you to specify mappings for the subdirectories you wish to mount under the **directory** indicated on the command line. In an indirect map, each **key** consists of a simple name that refers to one or more file systems that are to be mounted as needed.

Direct Maps

Entries in a direct map are associated directly with **autofs** mount points. Each **key** is the full pathname of an **autofs** mount point. The direct map as a whole is not associated with any single directory.

Included Maps

The contents of another map can be included within a map with an entry of the form

+*mapname*

If *mapname* begins with a slash then it is assumed to be the pathname of a local file. Otherwise the location of the map is determined by the policy of the name service switch according to the entry for the automounter in **/etc/nsswitch.conf**, such as

```
automount: files nis
```

If the name service is **files** then the name is assumed to be that of a local file in **/etc**. If the key being searched for is not found in the included map, the search continues with the next entry.

Special Maps

There are three special maps available: –**hosts**, –**xfn**, and –**null**. The –**hosts** map is used with the **/net** directory and assumes that the map key is the hostname of an NFS server. The **automountd** daemon dynamically constructs a map entry from the server's list of exported file systems. For instance a reference to **/net/hermes/usr** would initiate an automatic mount of all exported file systems from **hermes** that are mountable by the client. References to a directory under **/net/hermes** will refer to the corresponding directory relative to **hermes** root.

The **-xfn** map is used to mount the initial context of the Federated Naming Service (FNS) namespace under the **/xfn** directory. For more information on FNS, see **fns**(5), **fns_initial_context**(5), **fns_policies**(5), and the Federated Naming Service Guide.

The –**null** map, when indicated on the command line, cancels a previous map for the directory indicated. This is most useful in the **/etc/auto_master** for cancelling entries that would otherwise be inherited from the **+auto_master** include entry. To be effective, the –**null** entries must be inserted before the included map entry.

Executable Maps

Local maps that have the execute bit set in their file permissions will be executed by the automounter and provided with a key to be looked up as an argument. The executable map is expected to return the content of an automounter map entry on its stdout or no output if the entry cannot be determined.

Configuration and the auto_master Map

When initiated without arguments, **automount** consults the master map for a list of **autofs** mount points and their maps. It mounts any **autofs** mounts that are not already mounted, and unmounts **autofs** mounts that have been removed from the master map or direct map.

The master map is assumed to be called **auto_master** and its location is determined by the name service switch policy. Normally the master map is located initially as a local file **/etc/auto_master**.

Browsing

The Solaris 2.6 release supports browsability of indirect maps. This allows all of the potential mount points to be visible, whether or not they are mounted. The –**nobrowse** option can be added to any indirect **autofs** map to disable browsing. For example:

```
/net      –hosts         –nosuid,nobrowse
/home     auto_home
```

In this case, any *hostnames* would only be visible in **/net** after they are mounted, but all potential mount points would be visible under **/home**. The –**browse** option enables browsability of **autofs** file systems. This is the default for all indirect maps.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
1	An error occurred.

FILES

/etc/auto_master	master automount map.
/etc/auto_home	map to support automounted home directories.
/etc/nsswitch.conf	the name service switch configuration file.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

automountd(1M), **mount**(1M), **mount_cachefs**(1M), **mount_nfs**(1M), **attributes**(5), **fns**(5), **fns_initial_context**(5), **fns_policies**(5), **nfssec**(5)

NFS Administration Guide

NOTES

Autofs mount points must not be hierarchically related. **automount** does not allow an **autofs** mount point to be created within another **autofs** mount.

Since each direct map entry results in a new **autofs** mount such maps should be kept short.

Entries in both direct and indirect maps can be modified at any time. The new information is used when **automountd** next uses the map entry to do a mount.

New entries added to a master map or direct map will not be useful until the automount command is run to install them as new **autofs** mount points. New entries added to an indirect map may be used immediately.

As of the Solaris 2.6 release, a listing (**ls**) of the **autofs** directory associated with an indirect map shows all potential mountable entries. The attributes associated with the potential mountable entries are temporary. The real file system attributes will only be shown once the file system has been mounted.

Default mount options can be assigned to an entire map when specified as an optional third field in the master map. These options apply only to map entries that have no mount options. Note that map entities with options override the default options, as at this time, the options do not concatenate. The concatenation feature is planned for a future release.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same.

NAME

automountd – autofs mount / unmount daemon

SYNOPSIS

automountd [**–Tvn**] [**–D** *name*=*value*]

DESCRIPTION

automountd is an RPC server that answers file system mount and unmount requests from the **autofs** file system. It uses local files or name service maps to locate file systems to be mounted. These maps are described with the **automount**(1M) command.

The **automountd** daemon is automatically invoked in run level 2.

OPTIONS

–T Trace. Expand each RPC call and display it on the standard output.

–v Verbose. Log status messages to the console.

–n Turn off browsing for all **autofs** mount points. This option overrides the **-browse autofs** map option on the local host.

–D *name*=*value*
Assign *value* to the indicated **automount** map substitution variable. These assignments cannot be used to substitute variables in the master map **auto_master**.

USAGE

See **largefile**(5) for the description of the behavior of **automountd** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/auto_master master map for automounter

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

automount(1M), **attributes**(5), **largefile**(5)

NAME

autopush – configures lists of automatically pushed STREAMS modules

SYNOPSIS

autopush –**f** *filename*
autopush –**g** –**M** *major* –**m** *minor*
autopush –**r** –**M** *major* –**m** *minor*

DESCRIPTION

This command is used to configure the list of modules to be automatically pushed onto the stream when a device is opened. It can also be used to remove a previous setting or get information on a setting.

OPTIONS

–**f** *filename* Set up the **autopush** configuration for each driver according to the information stored in *filename*. An **autopush** file consists of lines of four or more fields, separated by spaces as shown below:

major minor last-minor module1 module2 . . . modulen

The first field is a string that specifies the *major* device name, as listed in the **/kernel/drv** directory. The next two fields are integers that specify the *minor* device number and *last-minor* device number. The fields following represent the names of modules. If *minor* is –1, then all minor devices of a major driver specified by *major* are configured, and the value for *last-minor* is ignored. If *last-minor* is 0, then only a single minor device is configured. To configure a range of minor devices for a particular major, *minor* must be less than *last-minor*.

The last fields of a line in the **autopush** file represent the list of module names. The maximum number of modules that can be automatically pushed on a stream is eight. The modules are pushed in the order they are specified. Comment lines start with a **#** sign.

–**g** Get the current configuration setting of a particular *major* and *minor* device number specified with the –**M** and –**m** options respectively and displays the autopush modules associated with it. It will also return the starting minor device number if the request corresponds to a setting of a range (as described with the –**f** option).

–**M** *major* Specifies the major device number.

–**m** *minor* Specifies the minor device number.

–**r** Remove the previous configuration setting of the particular *major* and *minor* device number specified with the –**M** and –**m** options respectively. If the values of *major* and *minor* correspond to a previously established setting of a range of minor devices, where *minor* matches the first minor device number in the range, the configuration would be removed for the entire range.

EXAMPLES

The following example gets the current configuration settings for the *major* and *minor* device numbers as indicated and displays the **autopush** modules associated with them for the character-special device **/dev/term/a:**

```
example# autopush −g −M 29 −m 0
Major    Minor   Lastminor    Modules
   29        0        1             ldterm ttcompat
```

FILES

/etc/iu.ap

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

bdconfig(1M), **ttymon**(1M), **attributes**(5), **ldterm**(7M), **sad**(7D), **streamio**(7I) **ttcompat**(7M)

STREAMS Programming Guide

NAME

bdconfig – configures the bd (buttons and dials) stream

SYNOPSIS

bdconfig [**startup**] [**off**] [**on**] [*term*] [**status**] [**verbose**]

DESCRIPTION

The **bdconfig** utility is responsible for configuring the autopush facility and defining to the system what serial device to use for the **bd** stream.

OPTIONS

If no options are given, then an interactive mode is assumed. In this mode the current status is presented along with this usage line, and a series of interactive questions asked to determine the user’s desires.

Root privilege is required to change the configuration. The status option does not require root privilege. **bdconfig** can be installed as a setuid root program.

The non-interactive options below can be given in any order.

term Specify to the system the serial device for **bd** use. This option implies the **on** option unless the **off** option is present.

iff Reconfigure the configured term for tty use.

on Reconfigure the configured term for **bd** use. If *term* has not been previously specified, interactive questions are asked to determine the user’s desires.

startup Configure as was last configured before the system went down. This option is used by the startup script, and precludes the use of the **on**, **off**, and *term* options. This option implies non-interactive mode.

status Emit the current configuration in terms of the words used as options: **off**, **on**, **/dev/term/a**, **/dev/term/b**, and so forth. This option implies non interactive mode.

verbose **bdconfig** describes what it finds and what it is doing.

EXIT STATUS

The **bdconfig** utility returns **0** on success, **1** on general error, and **2** on argument error.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdialh

SEE ALSO

autopush(1M), **attributes**(5), **x_buttontest**(6), **x_dialtest**(6), **bd**(7M), **sad**(7D), **streamio**(7I)

NOTES

All **bdconfig** does is configure the AUTOPUSH facility. **bdconfig** does not actually manipulate the serial port or stream in any way. Only the first open of a dismantled stream will see the effects of a previously run **bdconfig**.

The **bdconfig** utility is silent except for error messages unless:

a) invoked with no args: status / usage line emitted
b) interactive modes are invoked as described above
c) the verbose option is used

BUGS

The interface does not support more than one dialbox and one buttonbox, both of which must be on the same serial device.

There should be a library routine to read, parse, and validate records in the **iu.ap** file, so that **bdconfig** could return to the appropriate record in **iu.ap** as the default configuration.

NAME

boot – start the system kernel or a standalone program

SYNOPSIS

SPARC

boot [*OBP names*] [*file*] [–**a**] [–**f**] [*boot-flags*]

b [*device* [*(c, u, p)*]] [*file*] [–**a**] [–**f**] [*boot-flags*]

x86

b [*file*] [–**f**] [*boot-args*]
i

DESCRIPTION

Bootstrapping is the process of loading and executing a standalone program. For the purpose of this discussion, bootstrapping means the process of loading and executing the bootable operating system. Typically, the standalone program is the operating system kernel (see **kernel**(1M)), but any standalone program can be booted instead. As an example, on a SPARC based system, the diagnostic monitor for a machine is a good example of a standalone program other than the operating system that can be booted.

If the standalone is identified as a dynamically-linked executable, boot will load the interpreter (linker / loader) as indicated by the executable format and then transfer control to the interpreter. If the standalone is statically-linked, it will jump directly to the standalone.

Typically, the standalone program is the kernel. Once the program is loaded, it starts the UNIX system, mounts the necessary filesystems (see **vfstab**(4)), and runs **/sbin/init** to bring the system to the "initdefault" state specified in **/etc/inittab** (see **inittab**(4)).

SPARC Bootstrap Procedure

On SPARC based systems, the bootstrap procedure on most machines consists of the following basic phases.

After the machine is turned on, the system firmware (in PROM) executes power-on self-test (POST). The form and scope of these tests depends on the version of the firmware in your system.

After the tests have been completed successfully, the firmware attempts to autoboot if the appropriate flag has been set in the non-volatile storage area used by the firmware. The name of the file to load, and the device to load it from can also be manipulated.

These flags and names can be set using the **eeprom**(1M) command from the shell, or by using PROM commands from the **ok** prompt after the system has been halted.

The second level program is either **ufsboot**, (when booting from a disk) or **inetboot** (when booting across the network). When booting from disk, the bootstrapping process consists of two conceptually distinct phases, primary boot and secondary boot. The PROM assumes that the primary bootblock resides in blocks 1 to 15 of the local disk. When booting over the network, the PROM makes a reverse ARP request and when it receives a reply, the PROM makes a TFTP request to the server that responded and fetches inetboot across the network and executes it. Inetboot also makes another reverse ARP request, then uses the **bootparams** protocol to locate its root filesystem. It then fetches the kernel across the network using the NFS protocol and then executes it.

If the pathname to the standalone is relative (does not begin with a slash), the second level boot will look for the standalone in a platform-dependent location. The first choice is to prepend **/platform/***platform-name* to the name of the standalone. The next choice is to prepend **/platform/***hardware-class-name* to the name of the standalone. See **filesystem**(5). If the pathname is absolute, **boot** will use the specified path. The **boot** program then loads the standalone at the appropriate address, and then transfers control.

x86 Bootstrap Procedure

On x86 based systems, the bootstrapping process consists of two conceptually distinct phases, primary boot and secondary boot. The primary boot is implemented in the BIOS ROM on the system board, and BIOS extensions in ROMs on peripheral boards. It is distinguished by its ability to control the installed peripheral devices and to provide I/O services through software interrupts. It begins the booting process by loading the first physical sector from a floppy disk or hard disk. The primary boot is implemented in x86 real-mode code.

The secondary boot is loaded by the primary boot. It is implemented in 32-bit, paged, protected mode code. It also loads and uses peripheral-specific BIOS extensions written in x86 real-mode code. The secondary boot is called **boot.bin** and is capable of reading and booting from a UFS file system on a hard disk or a CD or via a LAN using the NFS protocol.

The secondary boot is responsible for running the **Configuration Assistant** program which determines the installed devices in the system (possibly with help from the user). The secondary boot then reads the script in **/etc/bootrc**, which controls the booting process. This file contains **boot interpreter** commands, which are defined below, and can be modified to change defaults or to adapt to a specific machine.

The standard **/etc/bootrc** script prompts the user to enter a **b** character to boot with specified options, an **i** character to invoke the interpreter interactively, or any other character to boot the default kernel. Once the kernel is loaded, it starts the operating system, loads the necessary modules, mounts the necessary filesystems (see **vfstab**(4)), and runs **/sbin/init** to bring the system to the ‘‘initdefault’’ state specified in **/etc/inittab** (see **inittab**(4)).

OPTIONS

SPARC

OBP names Specify the open boot prom designations. For example, on Desktop SPARC based systems, the designation **/sbus/esp@0,800000/sd@3,0:a** indicates a SCSI disk (sd) at target 3, lun0 on the SCSI bus, with the esp host adapter plugged into slot 0.

file Name of a standalone program to boot. The default is to boot **/platform/***platform-name***/kernel/unix** or **/platform/***hardware-class-name***/kernel/unix** from the root partition. Some versions of the firmware allow the default filename to be saved in the non-volatile storage area of the system.

–a The boot program interprets this flag to mean **ask me**, and so it prompts for the name of the standalone. The **’–a’** flag is then passed to the standalone program.

–f When booting an Autoclient system, this flag forces the boot program to bypass the client's local cache and read all files over the network from the client's file server. This flag is ignored for all non-Autoclient systems. The **–f** flag is then passed to the standalone program.

boot-flags The boot program passes all *boot-flags* to *file*. They are not interpreted by **boot**. See the **kernel**(1M) and **kadb**(1M) manual pages for information on the options available with the default standalone program, **kernel/unix**.

x86

file Name of a standalone program to boot. The default is to boot **/platform/***platform-name***/kernel/unix** from the root partition, but you can specify another program on the command line.

–f When booting an Autoclient system, this flag forces the boot program to bypass the client's local cache and read all files over the network from the client's file server. This flag is ignored for all non-Autoclient systems. The **–f** flag is then passed to the standalone program.

boot-args The boot program passes all *boot-args* to *file*. They are not interpreted by **boot**. See the **kernel**(1M) and **kadb**(1M) manual pages for information on the options available with the kernel.

x86 BOOT SEQUENCE DETAILS

After a PC-compatible machine is turned on, the system firmware in the BIOS ROM executes a power-on self test (POST), runs BIOS extensions in peripheral board ROMs, and invokes software interrupt INT 19h, Bootstrap. The INT 19h handler typically performs the standard PC-compatible boot, which consists of trying to read the first physical sector from the first diskette drive, or, if that fails, from the first hard disk. The processor then jumps to the first byte of the sector image in memory.

x86 Primary Boot

The first sector on a floppy disk contains the master boot block. The boot block is responsible for loading the image of the boot loader **strap.com**, which then loads the secondary boot, **boot.bin**.

The first sector on a hard disk contains the master boot block, which contains the master boot program and the FDISK table, named for the PC program that maintains it. The master boot finds the active partition in the FDISK table, loads its first sector, and jumps to its first byte in memory. This completes the standard PC-compatible hard disk boot sequence.

An x86 FDISK partition for the Solaris software begins with a one-cylinder boot slice, which contains the partition boot program (**pboot**) in the first sector, the standard Solaris disk label and volume table of contents (VTOC) in the second and third sectors, and the **bootblk** program in the fourth and subsequent sectors. When the FDISK partition for the Solaris software is the active partition, the master boot program (**mboot**) reads the partition boot program in the first sector into memory and jumps to it. It in turn reads the **bootblk** program into memory and jumps to it.

An x86 FDISK Boot FDISK partition for the Solaris software begins with the pboot program, followed immediately by the multi-sector bootstrap program (**strap.com**). When this type of FDISK partition is the active partition, the master boot program reads the partition boot program in the first sector into memory and jumps to it. It in turn reads the **strap.com** program from the first sector into memory and jumps to it.

Regardless of the type of the active partition, if the drive contains multiple FDISK partitions, the user is given the opportunity to reboot another partition.

bootblk or **strap.com** (depending upon the active partition type), reads **boot.bin** from the file system in the Solaris root slice and jumps to its first byte in memory.

x86 Secondary Boot

The secondary boot, **boot.bin**, switches the processor to 32-bit, paged, protected mode, and performs some limited machine initialization. It runs the **Configuration Assistant** program which either auto-boots the system, or presents a list of possible boot devices, depending on the state of the **auto-boot?** variable (see **eeprom**(1M)).

Disk target devices (including CDROM drives) are expected to contain UFS filesystems. Network devices will first issue Reverse Address Resolution Protocol (RARP) requests to discover the machine's IP address and then a **bootparams** RPC to find out which server will provide the root file system. The root file system is then mounted using NFS. After a successful root mount, **boot.bin** invokes a command interpreter, which interprets **/etc/bootrc**.

Secondary Boot Programming Language for x86

The wide range of hardware that must be supported on x86 based systems demands great flexibility in the booting process. This flexibility is achieved in part by making the secondary boot programmable. The secondary boot contains an interpreter that accepts a simple command language similar to those of **sh** and **csh**. The primary differences are that pipelines, loops, standard output, and output redirection are not supported.

x86 Lexical Structure

The boot interpreter splits input lines into words separated by blanks and tabs. The metacharacters are dollar sign (**$**), single-quote (**´**), double-quote ("), number sign (**#**), new-line, and backslash (\). The special meaning of metacharacters can be avoided by preceding them with a backslash. A new-line preceded by a backslash is treated as a blank. A number sign introduces a comment, which continues to the next new-line.

A string enclosed in a pair of single-quote or double-quote characters forms all or part of a single word. White space and new-line characters within a quoted string become part of the word. Characters within a quoted string can be quoted by preceding them with a backslash character; thus a single-quote character can appear in a single-quoted string by preceding it with a backslash. Two backslashes produce a single backslash, and a new-line preceded by a backslash produces a new-line in the string.

x86 Variables

The boot maintains a set of variables, each of which has a string value. The first character of a variable name must be a letter, and subsequent characters can be letters, digits, or underscores. The **set** command creates a variable and / or assigns a value to it, or displays the values of variables. The **unset** command deletes a variable.

Variable substitution is performed when the interpreter encounters a dollar-sign that is not preceded by a backslash. The variable name following the dollar sign is replaced by the value of the variable, and parsing continues at the beginning of the value. Variable substitution is performed in double-quoted strings, but not in single-quoted strings. A variable name can be enclosed in braces to separate it from following characters.

x86 Commands

A command is a sequence of words terminated by a new-line character. The first word is the name of the command and subsequent words are arguments to the command. All commands are built-in commands. Standalone programs are executed with the **run** command.

x86 Conditional Execution of Commands

Commands can be conditionally executed by surrounding them with the **if**, **elseif**, **else**, and **endif** commands:

```
if expr1
   ...
elseif expr2
   ...
elseif expr3
   ...
else
   ...
endif
```

An **if** block may be embedded in other **if** blocks.

x86 Expressions

The **set**, **if**, and **elseif** commands evaluate arithmetic expressions with the syntax and semantics of the C programming language. The **||**, **&&**, **|**, **^**, **&**, **==**, **!=**, **<**, **>**, **<=**, **>=**, **>>**, **<<**, **+**, **–**, *****, **/**, **%**, **~**, and **!** operators are accepted, as are **(**, **)**, and comma. Signed 32-bit integer arithmetic is performed.

Expressions are parsed after the full command line has been formed. Each token in an expression must be a separate argument word, so blanks must separate all tokens on the command line.

Before an arithmetic operation is performed on an operand word, it is converted from a string to a signed 32-bit integer value. After an optional leading sign, a leading 0 produces octal conversion and a leading 0x or 0X produces hexadecimal conversion. Otherwise, decimal conversion is performed. A string that is not a legal integer is converted to zero.

Several built-in functions for string manipulation are provided. Built-in function names begin with a dot. String arguments to these functions are not converted to integers. To cause an operator, for example, **-**, to be treated as a string, it must be preceded by a backslash, and that backslash must be quoted with another backslash. Also be aware that a null string can produce a blank argument, and thus an expression syntax error. For example:

```
if .strneq ( ${usrarg}X , \– , 1 )
```

is the safe way to test whether the variable *usrarg* starts with a –, even if it could be null.

x86 I/O

The boot interpreter takes its input from the system console or from one or more files. The source command causes the interpreter to read a file into memory and begin parsing it. The console command causes the interpreter to take its input from the system console. Reaching EOF causes the interpreter to resume parsing the previous input source. CTRL-D entered at the beginning of console line is treated as EOF.

The **echo** command writes its arguments to the display. The **read** command reads the system console and assigns word values to its argument variables.

x86 Debugging

The **verbose** command turns verbose mode on and off. In verbose mode, the interpreter displays lines from the current source file and displays the command as actually executed after variable substitution.

The **singlestep** command turns singlestep mode on and off. In singlestep mode, the interpreter displays **step ?** before processing the next command, and waits for keyboard input, which is discarded. Processing proceeds when ENTER is pressed. This allows slow execution in verbose mode.

x86 Initialization

When the interpreter is first invoked by the boot, it begins execution of a compiled-in initialization string. This string typically consists of "**source /etc/bootrc\n**" to run the boot script in the root file system.

x86 Communication With Standalone Programs

The boot passes information to standalone programs through arguments to the **run** command. A standalone program can pass information back to the boot by setting a boot interpreter variable using the **var_ops()** boot service function. It can also pass information to the kernel using the **setprop()** boot service function. The **whoami** property is set to the name of the standalone program.

x86 Built-in Commands

console Interpret input from the console until CTRL-D.

echo *arg1* . . . Display the arguments separated by blanks and terminate with a new-line.

echo -n *arg1* . . . Display the arguments separated by blanks, but do not terminate with a new-line.

getprop *propname varname*

Assign the value of property *propname* to the variable *varname*. A property value of length zero produces a null string. If the property does not exist, the variable is not set.

getproplen *propname varname*

Assign the length in hexadecimal of the value of property *propname* to the variable *varname*. Property value lengths include the terminating null. If the property does not exist, the variable is set to 0xFFFFFFFF (-1).

if *expr*
If the expression *expr* is true, execute instructions to the next **elseif**, **else**, or **endif**. If **expr** is false, do not execute the instructions.

elseif *expr*
If the preceding **if** and **elseif** commands all failed, and *expr* is true, execute instructions to the next **elseif**, **else**, or **endif**. Otherwise, do not execute the instructions.

else
If the preceding **if** and **elseif** commands all failed, execute instructions to the next **elseif**, **else**, or **endif**. Otherwise, do not execute the instructions.

endif
Revert to the execution mode of the surrounding block.

help
Display a help screen that contains summaries of all available boot shell commands.

read *name1* . . .
Read a line from the console, break it into words, and assign them as values to the variables *name1*, etc.

readt *time* . . .
Same as **read**, but timeout after *time* seconds.

run *name arg1* . . .
Load and transfer control to the standalone program *name*, passing it *arg1* and further arguments.

set
Display all the current variables and their values.

set *name*
Set the value of the variable name to the null string.

set *name word*
Set the value of the variable *name* to *word*.

set *name expr*
Set the value of the variable name to the value of *expr*. *expr* must consist of more than one word. The value is encoded in unsigned hexadecimal, so that −1 is represented by 0xFFFFFFFF.

setcolor
Set the text mode display attributes. Allowable colors are *black*, *blue*, *green*, *cyan*, *red*, *magenta*, *brown*, *white*, *gray*, *lt_blue*, *lt_green*, *lt_cyan*, *lt_red*, *lt_magenta*, *yellow*, and *hi_white*.

setprop *propname word*
Set the value of the property *propname* to *word*.

singlestep or **singlestep on**
Turn on singlestep mode, in which the interpreter displays **step ?** before each command is processed, and waits for keyboard input. Press ENTER to execute the next command.

singlestep off
Turn off singlestep mode.

source *name*
Read the file *name* into memory and begin to interpret it. At EOF, return to the previous source of input.

unset *name*
Delete the variable name.

verbose or **verbose on**
Turn on verbose mode, which displays lines from source files and commands to be executed.

verbose off Turn off verbose mode.

x86 Built-in Functions

The following built-in functions are accepted within expressions:

.strcmp(*string1*, *string2*)
Returns an integer value that is less than, equal to, or greater than zero, as *string1* is lexicographically less than, equal to, or greater than *string2*.

.strncmp(*string1*, *string2*, *n*)
Returns an integer value that is less than, equal to, or greater than zero, as *string1* is lexicographically less than, equal to, or greater than *string2*. At most, *n* characters are compared.

.streq (*string1*, *string2*)
Returns true if *string1* is equal to *string2*, and false otherwise.

.strneq (*string1*, *string2*, *n*)
Returns true if *string1* is equal to *string2*, and false otherwise. At most, *n* characters are compared.

.strfind (*string*, *addr*, *n*)
Scans *n* locations in memory starting at *addr*, looking for the beginning of *string*. The *string* in memory need not be null-terminated. Returns true if *string* is found, and false otherwise. **.strfind** can be used to search for strings in the ROM BIOS and BIOS extensions that identify different machines and peripheral boards.

FILES

/platform/*platform-name***/ufsboot**
second level program to boot from a disk or CD.

/platform/*platform-name***/kernel/unix**
default program to boot system.

/etc/inittab table in which the "initdefault" state is specified.

/sbin/init program that brings the system to the "initdefault" state.

x86 Only

/etc/bootrc script that controls the booting process.

/platform/*platform-name***/boot/solaris/boot.bin**
second level boot program used on x86 systems in place of **ufsboot**.

/platform/*platform-name***/boot**
directory containing boot-related files.

SEE ALSO

uname(1), **eeprom**(1M), **init**(1M), **installboot**(1M), **kadb**(1M), **kernel**(1M), **shutdown**(1M), **inittab**(4), **vfstab**(4), **filesystem**(5)

System Administration Guide

WARNINGS

The **boot** utility is unable to determine which files can be used as bootable programs. If the booting of a file that is not bootable is requested, the **boot** utility loads it and branches to it. What happens after that is unpredictable.

NOTES

platform-name can be found using the –**i** option of **uname**(1). *hardware-class-name* can be found using the –**m** option of **uname**(1).

x86 Only

Because the ‘‘**-**’’ key on national language keyboards has been moved, an alternate key must be used to supply arguments to the **boot** command on an x86 based system using these keyboards. Use the ‘‘**-**’’ on the numeric keypad. The specific language keyboard and the alternate key to be used in place of the ‘‘**-**’’ during bootup is shown below.

Keyboard	**Substitute Key**
Italy	´
Spain	´
Sweden	+
France	?
Germany	?

For example, **b** –**r** would be typed as **b +r** on Swedish keyboards, although the screen display will show as **b** –**r**.

NAME

bsmconv, bsmunconv – enable / disable the Basic Security Module (BSM) on Solaris

SYNOPSIS

/etc/security/bsmconv [rootdir ...]
/etc/security/bsmunconv [rootdir ...]

DESCRIPTION

The **bsmconv** and **bsmunconv** scripts are used to enable or disable the BSM features on a Solaris system. The optional argument **rootdir** is a list of one or more root directories of diskless clients which have already been configured via the Host Manager, see **admintool**(1M)**.**

To enable or disable BSM on a diskless client, a server, or a stand-alone system, logon as super-user to the system being converted and use the **bsmconv** or **bsmunconv** commands without any options.

To enable or disable BSM on a diskless client from that client's server, logon to the server as super-user and use **bsmconv**, specifying the root directory of each diskless client you wish to affect. For example, the command

myhost# bsmconv /export/root/client1 /export/root/client2

will enable BSM on the two machines named *client1* and *client2.* While the command

myhost# bsmconv

will enable BSM only on the machine called **myhost.** It is no longer necessary to enable BSM on both the server and its diskless clients.

After running **bsmconv** the system can be configured by editing the files in **/etc/security.** Each diskless client has its own copy of configuration files in its root directory. You may wish to edit these files before rebooting each client.

Following the completion of either script, the affected system(s) should be rebooted to allow the auditing subsystem to come up properly initialized.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

admintool(1M) **auditd**(1M), **audit_startup**(1M), **audit.log**(4), **audit_control**(4), **attributes**(5)

NAME

cachefslog – Cache File System logging

SYNOPSIS

cachefslog [–**f** *logfile* | –**h**] *cachefs_mount_point*

DESCRIPTION

The **cachefslog** command displays where CacheFS statistics are being logged. Optionally, it sets where CacheFS statistics are being logged, or it halts logging for a cache specified by *cachefs_mount_point*. The *cachefs_mount_point* argument is a mount point of a cache file system. All file systems cached under the same cache as *cachefs_mount_point* will be logged.

OPTIONS

The following options are supported. You must be super-user to use the –**f** and –**h** options.

–**f** *logfile*	Specify the log file to be used.
–**h**	Halt logging.

OPERANDS

cachefs_mount_point	A mount point of a cache file system.

USAGE

See **largefile**(5) for the description of the behavior of **cachefslog** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

The example below checks if the directory **/home/sam** is being logged:

```
example% cachefslog /home/sam
not logged: /home/sam
```

The example below changes the *logfile* of **/home/sam** to **/var/tmp/samlog**:

```
example# cachefslog -f /var/tmp/samlog /home/sam
/var/tmp/samlog: /home/sam
```

The example below verifies the change of the previous example:

```
example% cachefslog /home/sam
/var/tmp/samlog: /home/sam
```

The example below halts logging for the **/home/sam** directory:

```
example# cachefslog -h /home/sam
not logged: /home/sam
```

EXIT STATUS

The following exit values are returned:

0	success
non-zero	an error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cachefsstat(1M), **cachefswssize**(1M), **cfsadmin**(1M), **attributes**(5), **largefile**(5)

DIAGNOSTICS

Invalid path It is illegal to specify a path within a cache file system.

NAME

cachefspack – pack files and file systems in the cache

SYNOPSIS

cachefspack [**−h**] [**−i** | **−p** | **−u**] [**−f** *packing-list*] [**−U** *cache-directory*] [*file*. . .]

DESCRIPTION

The **cachefspack** utility is used to set up and maintain files in the cache. This utility affords greater control over the cache, ensuring that the specified files will be in the cache whenever possible.

OPTIONS

The following options are supported:

−f *packing-list* Specify a file containing a list of files and directories to be packed. Options within subdirectories and files can also be specified. The format and rules governing *packing-list* are described on the **packingrules**(4) manual page. Directories are packed recursively. Symlinks that match a regular expression on a **LIST** command are followed. Symlinks encountered while recursively processing directories are not followed.

−h Help. Print a brief summary of all the options.

−i View information about the packed files.

−p Pack the file or files specified by *file*. This is the default behavior.

−u Unpack the file or files specified by *file*.

−U *cache-directory* Unpack all files in the specified cache directory.

OPERANDS

The following operands are supported:

file A path name of a file to be packed or unpacked.

USAGE

See **largefile**(5) for the description of the behavior of **cachefspack** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

1. The following example packs the file **projects** in the cache.

   ```
   % cachefspack −p projects
   ```

2. The following example packs the files **projects**, **updates**, and **master_plan** in the cache.

   ```
   % cachefspack −p projects updates master_plan
   ```

3. The following example unpacks the file **projects** from the cache.

   ```
   % cachefspack −u projects
   ```

4. The following example unpacks the files **projects**, **updates**, and **master_plan** from the cache.

   ```
   % cachefspack −u projects updates master_plan
   ```

5. The following example unpacks all files in the cache directory **cache1**.

   ```
   % cachefspack –U /cache/cache1
   ```

6. The following example illustrates the use of a packing list to specify files to be packed in the cache. The contents of **lists.pkg** are as follows:

   ```
   IGNORE SCCS
   BASE /src/junk
   LIST ∗.c
   LIST ∗.h
   ```

   This example will pack all files in the directory **/src/junk** with **.c** and **.h** extensions that do not contained the string SCCS in the file's path name.

   ```
   % cachefspack –f lists.pkg
   ```

EXIT STATUS

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cfsadmin(1M), **mount_cachefs**(1M), **packingrules**(4), **attributes**(5), **largefile**(5)

NAME

cachefsstat – Cache File System statistics

SYNOPSIS

/usr/bin/cachefsstat [**–z**] [*path…*]

DESCRIPTION

The **cachefsstat** command displays statistical information about the cache file system mounted on *path*. The statistical information includes cache hits and misses, consistency checking, and modification operations. If *path* is not specified, all mounted cache file systems are used.

cachefsstat can also be used to reinitialize this information (see **–z** option).

The statistical information has the following format:

<cache hit rate>
<consistency checks>
<modifies>
where:

hit rate The percentage of cache hits over the total number of attempts, followed by the actual numbers of hits and misses.

consistency checks The number of consistency checks performed, followed by the number that passed, and the number that failed.

modifies The number of modify operations, including writes, creates, etc.

OPTIONS

The following option is supported:

–z Zero (reinitialize) statistics. Execute **cachefsstat –z** before executing **cachefsstat** again to gather statistics on the cache performance. This option can only be use by the superuser. The statistics printed reflect those just before the statistics are reinitialized.

USAGE

See **largefile**(5) for the description of the behavior of **cachefsstat** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

```
example% cachefsstat /home/sam
cache hit rate:  73% (1234 hits, 450 misses)
consistency checks:  700 (650 pass, 50 fail)
modifies:  321
```

EXIT STATUS

The following exit values are returned:

0 success

non-zero an error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO **cachefslog**(1M), **cachefswssize**(1M), **cfsadmin**(1M), **attributes**(5), **largefile**(5)

NAME

cachefswssize – determine working set size for cachefs

SYNOPSIS

cachefswssize *logfile*

DESCRIPTION

The **cachefswssize** command displays the workspace size determined from *logfile*. This includes the amount of cache space needed for each filesystem that was mounted under the cache, as well as a total.

USAGE

See **largefile**(5) for the description of the behavior of **cachefswssize** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

```
example% cachefswssize /var/tmp/samlog
/home/sam
                          end size:     10688k
                   high water size:     10704k

/foo
                          end size:       128k
                   high water size:       128k

/usr/dist
                          end size:      1472k
                   high water size:      1472k

total for cache
                      initial size:    110960k
                          end size:     12288k
                   high water size:     12304k
```

EXIT STATUS

The following exit values are returned:

0 success

non-zero an error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cachefslog(1M), **cachefsstat**(1M), **cfsadmin**(1M), **attributes**(5), **largefile**(5)

DIAGNOSTICS

problems were encountered writing log file

There were problems encountered when the kernel was writing the logfile. The most common problem is running out of disk space.

invalid log file
The logfile is not a valid logfile or was created with a newer version of Solaris than the one where **cachefswssize** is running.

NAME

captoinfo – convert a termcap description into a terminfo description

SYNOPSIS

captoinfo [–**1**] [–**v** . . .] [–**V**] [–**w** *width*] *filename* . . .

DESCRIPTION

captoinfo looks in *filename* for **termcap** descriptions. For each one found, an equivalent **terminfo** description is written to standard output, along with any comments found. A description which is expressed as relative to another description (as specified in the **termcap tc = field**) is reduced to the minimum superset before being displayed.

If no *filename* is given, then the environment variable **TERMCAP** is used for the filename or entry. If **TERMCAP** is a full pathname to a file, only the terminal whose name is specified in the environment variable **TERM** is extracted from that file. If the environment variable **TERMCAP** is not set, then the file **/usr/share/lib/termcap** is read.

OPTIONS

–**1** Display the fields one to a line. Otherwise, the fields are printed several to a line, with a maximum width of 60 characters.

–**v** Display tracing information on the standard error as the program runs. Specifying additional –**v** options displays more detailed information.

–**V** Display the version of the program in use on the standard error and then exit.

–**w** *width* Change the output to *width* characters.

FILES

/usr/share/lib/terminfo/?/* compiled terminal description database
/usr/share/lib/termcap

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

infocmp(1M), **curses**(3X), **terminfo**(4), **attributes**(5)

NOTES

captoinfo should be used to convert **termcap** entries to **terminfo** entries because the **termcap** database may not be supplied in future releases.

NAME

catman – create the formatted files for the reference manual

SYNOPSIS

/usr/bin/catman [–**n**] [–**p**] [–**t**] [–**w**] [–**M** *directory*] [–**T** *macro-package*]
[*sections*]

DESCRIPTION

catman creates the preformatted versions of the on-line manual from the **nroff**(1) input files. This feature allows easy distribution of the preformatted manual pages among a group of associated machines (for example, with **rdist**(1)), since it makes the directories of preformatted manual pages self-contained and independent of the unformatted entries.

catman also creates the **windex** database file in the directories specified by the **MANPATH** or the –**M** option. The **windex** database file is a three column list consisting of a keyword, the reference page that the keyword points to, and a line of text that describes the purpose of the utility or interface documented on the reference page. Each keyword is taken from the comma separated list of words on the **NAME** line before the '–' (dash). The reference page that the keyword points to is the first word on the **NAME** line. The text after the – on the **NAME** line is the descriptive text in the third column.

Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, **catman** recreates the **windex** database.

If a manual page is a *shadow* page, that is, it sources another manual page for its contents, a symbolic link is made in the **cat***x* or **fmt***x* directory to the appropriate preformatted manual page.

Shadow files in an unformatted nroff source file are identified by the first line being of the form **.so man***x***/yyy.***x***.**

OPTIONS

–**n** Do not create (or recreate) the **windex** database.

–**p** Print what would be done instead of doing it.

–**t** Create **troff**ed entries in the appropriate **fmt** subdirectories instead of **nroff**ing into the **cat** subdirectories.

–**w** Only create the **windex** database that is used by **whatis**(1) and the **man**(1) –**f** and –**k** options. No manual reformatting is done.

–**M** *directory*
Update manual pages located in the specified *directory*, (**/usr/share/man** by default).

–**T** *macro-package*
Use *macro-package* in place of the standard manual page macros, (**man**(5) by default).

OPERANDS

The following operand is supported:

sections If there is one parameter not starting with a '–', it is taken to be a space separated list of manual sections to be processed by **catman**. If this operand is specified, only the manual sections in the list will be processed. For example

catman 1 2 3

only updates manual sections **1**, **2**, and **3**. If specific sections are not listed, all sections in the **man** directory specified by the environment variable **MANPATH** are processed.

ENVIRONMENT

TROFF The name of the formatter to use when the –**t** flag is given. If not set, **troff**(1) is used.

MANPATH A colon-separated list of directories that are processed by **catman** and **man**(1). Each directory can be followed by a comma-separated list of sections. If set, its value overrides **/usr/share/man** as the default directory search path, and the **man.cf** file as the default section search path. The –**M** and –**s** flags, in turn, override these values.

FILES

/usr/share/man	default manual directory location
/usr/share/man/man∗/∗.∗	raw nroff input files
/usr/share/man/cat∗/∗.∗	preformatted **nroff**ed manual pages
/usr/share/man/fmt∗/∗.∗	preformatted **troff**ed manual pages
/usr/share/man/windex	table of contents and keyword database
/usr/lib/makewhatis	command script to make **windex** database
/usr/share/lib/tmac/an	default macro package

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdoc
CSI	Enabled

SEE ALSO

apropos(1), **man**(1), **nroff**(1), **rdist**(1), **rm**(1), **troff**(1), **whatis**(1), **attributes**(5), **man**(5)

DIAGNOSTICS

man?/xxx.? (.so'ed from man?/yyy.?): No such file or directory

The file outside the parentheses is missing, and is referred to by the file inside them.

target of .so in man?/xxx.? must be relative to /usr/man

catman only allows references to filenames that are relative to the directory **/usr/man**.

opendir:man?: No such file or directory

A harmless warning message indicating that one of the directories **catman** normally looks for is missing.

∗.∗**: No such file or directory**

A harmless warning message indicating **catman** came across an empty directory.

WARNINGS

If a user, who has previously run **catman** to install the **cat**∗ directories, upgrades the operating system, the entire **cat**∗ directory structure should be removed prior to running **catman**. See **rm**(1).

NOTES

If the −**n** option is specified, the **windex** database is not created and the **apropos**, **whatis**, **man** −**f**, and **man** −**k** commands will fail.

If the −**M** option is specified, the directory argument must not contain a ‘,’ (comma), since a comma is used to delineate section numbers. See **man**(1).

NAME

cfsadmin – administer disk space used for caching file systems with the Cache File-System (CacheFS)

SYNOPSIS

cfsadmin –**c** [–**o** *cacheFS-parameters*] *cache_directory*
cfsadmin –**d** [*cache_ID* | **all**] *cache_directory*
cfsadmin –**l** *cache_directory*
cfsadmin –**s** [*mntpt1* . . . | **all**]
cfsadmin –**u** [–**o** *cacheFS-parameters*] *cache_directory*

DESCRIPTION

The **cfsadmin** command provides the following functions:

- cache creation
- deletion of cached file systems
- listing of cache contents and statistics
- resource parameter adjustment when the file system is unmounted.

For each form of the command except –**s**, you must specify a cache directory, that is, the directory under which the cache is actually stored. A path name in the front file system identifies the cache directory. For the **-s** form of the command, you must specify a mount point.

You can specify a cache ID when you mount a file system with CacheFS, or you can let the system generate one for you. The –**l** option includes the cache ID in its listing of information. You must know the cache ID to delete a cached file system.

OPTIONS

–**c** Create a cache under the directory specified by *cache_directory*. This directory must not exist prior to cache creation.

–**d** Remove the file system whose cache ID you specify and release its resources, or remove all file systems in the cache by specifying **all**. After deleting a file system from the cache, you must run the **fsck_cachefs**(1M) command to correct the resource counts for the cache.

–**l** List file systems stored in the specified cache, as well as statistics about them. Each cached file system is listed by cache ID. The statistics document resource utilization and cache resource parameters.

–**s** Request a consistency check on the specified file system (or all cachefs mounted file systems). The –**s** option will only work if the cache file system was mounted with **demandconst** enabled (see **mount_cachefs**(1M)). Each file in the specified cache file system is checked for consistency with its corresponding file in the back file system. Note that the consistency check is performed file by file as files are accessed. If no files are accessed, no checks are performed. Use of this option will not result in a sudden "storm" of consistency checks.

–**u** Update resource parameters of the specified cache directory. Parameter values can only be increased. To decrease the values, you must remove the cache and recreate it. All file systems in the cache directory must be unmounted when you use this option. Changes will take effect the next time you mount any file system in the specified cache directory. The **–u** option with no **–o** option sets all parameters to their default values.

CacheFS Resource Parameters

You can specify the following CacheFS resource parameters as arguments to the **–o** option. Separate multiple parameters with commas.

maxblocks=*n* Maximum amount of storage space that CacheFS can use, expressed as a percentage of the total number of blocks in the front file system. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the space the **maxblocks** parameter allows will be available. The default is **90**.

minblocks=*n* Minimum amount of storage space, expressed as a percentage of the total number of blocks in the front file system, that CacheFS is always allowed to use without limitation by its internal control mechanisms. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the space the **minblocks** parameter attempts to reserve will be available. The default is **0**.

threshblocks=*n* A percentage of the total blocks in the front file system beyond which CacheFS cannot claim resources once its block usage has reached the level specified by **minblocks**. The default is **85**.

maxfiles=*n* Maximum number of files that CacheFS can use, expressed as a percentage of the total number of inodes in the front file system. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the inodes the **maxfiles** parameter allows will be available. The default is **90**.

minfiles=*n* Minimum number of files, expressed as a percentage of the total number of inodes in the front file system, that CacheFS is always allowed to use without limitation by its internal control mechanisms. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the inodes the **minfiles** parameter attempts to reserve will be available. The default is **0**.

threshfiles=*n* A percentage of the total inodes in the front file system beyond which CacheFS cannot claim inodes once its usage has reached the level specified by **minfiles**. The default is **85**.

maxfilesize=*n* Largest file size, expressed in megabytes, that CacheFS is allowed to cache. The default is **3**.

Note: You cannot decrease the block or inode allotment for a cache. To decrease the size of a cache, you must remove it and create it again with different parameters.

OPERANDS

cache_directory The directory under which the cache is actually stored.

mntpt1 The directory where the CacheFS is mounted.

USAGE

See **largefile**(5) for the description of the behavior of **cfsadmin** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

The following example creates a cache directory named **/cache**:

```
example# cfsadmin -c /cache
```

The following example creates a cache named **/cache1** that can claim a maximum of 60 percent of the blocks in the front file system, can use 40 percent of the front file system blocks without interference by CacheFS internal control mechanisms, and has a threshold value of 50 percent. The threshold value indicates that after CacheFS reaches its guaranteed minimum, it cannot claim more space if 50 percent of the blocks in the front file system are already used.

```
example# cfsadmin –c –o maxblocks=60,minblocks=40,
          threshblocks=50 /cache1
```

The following example changes the **maxfilesize** parameter for the cache directory **/cache2** to 2 megabytes:

```
example# cfsadmin –u –o maxfilesize=2 /cache2
```

The following example lists the contents of a cache directory named **/cache3** and provides statistics about resource utilization:

```
example# cfsadmin –l /cache3
```

The following example removes the cached file system with cache ID **23** from the cache directory **/cache3** and frees its resources (the cache ID is part of the information returned by **cfsadmin –l**):

```
example# cfsadmin –d 23 /cache3
```

The following example removes all cached file systems from the cache directory **/cache3**:

```
example# cfsadmin –d all /cache3
```

The following example checks for consistency all file systems mounted with **demandconst** enabled. No errors will be reported if no **demandconst** file systems were found.

```
example# cfsadmin –s all
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cachefslog(1M), **cachefsstat**(1M), **cachefswssize**(1M), **fsck_cachefs**(1M), **mount_cachefs**(1M), **attributes**(5), **largefile**(5)

NAME

cg14config – configure the cgfourteen device

SYNOPSIS

/usr/platform/*platform-name***/sbin/cg14config** [**–d** *device*] [**–r** *resolution*]
[**–g** *gammavalue*] [**–G** *gammafile*] [**–u** *degammavalue*] [**–U** *degammafile*]

DESCRIPTION

cg14config sets up state on the selected cgfourteen device. *platform-name* can be found using the **–i** option of **uname**(1).

cg14config is supported only on Desktop SPARCsystems with SX graphics option.

The interface, output, and command location are uncommitted and subject to change in future releases.

OPTIONS

–d *device* Use *device* as the cgfourteen device to configure. Default is **/dev/fb**.

–r *resolution* Use *resolution* as the desired screen resolution. Resolution is specified in terms of screen width and height (in pixels), and vertical refresh (in hz). Available resolutions are:

1024x768@60
1024x768@66
1024x768@70
1152x900@66
1152x900@76
1280x1024@66
1280x1024@76
1600x1280@66
1920x1080@72

The default is the value read from the monitor sense codes. Note that some or all of the resolutions above may not be supported by any given monitor. If a programmed resolution is outside of the range of allowable values for a monitor, unpredictable results can occur, including damage to the monitor. Thus, care should be taken when programming the resolution. See *Openboot Command Reference* for a description of how to reset the console device to the default value if it becomes unusable from programming an unsupported resolution.

The **–r** option is not available when the window system is running.

–g *gammavalue* Each entry of the gamma lookup table will be loaded with entry^(1 / *gammavalue*). The gamma lookup table has 256 entries. Default *gammavalue* is **2.2**.

–G *filename* Initialize the gamma lookup table with the contents of *filename*. The format of *filename* is 256 triplets (red green blue) of non-negative integers separated by NEWLINE characters. The integers must be in the range 0 to 1023, inclusive.

–u *degammavalue*
Each entry of the degamma lookup table will be loaded with

entry^(*degammavalue*). The degamma lookup table has 256 entries. Default *degammavalue* is **2.2**.

–U *filename* Initialize the degamma lookup table with the contents of *filename*. The format of *filename* is 256 entries of non-negative integers separated by NEWLINE characters. The integers must be in the range 0 to 255, inclusive.

EXIT STATUS

cg14config returns **0** on success and a positive integer on failure.

1 Selected device is not a cgfourteen device.

2 Requested action failed.

3 Unsupported resolution.

4 Gamma or degamma value out of range.

FILES

/platform/*platform-name***/kernel/drv/cgfourteen**
cgfourteen device driver

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm

SEE ALSO

uname(1), **init**(1M), **mmap**(2), **attributes**(5)

Platform Notes: SPARCstation 10SX System Configuration Guide

Openboot Command Reference

NAME

chown – change owner

SYNOPSIS

/usr/ucb/chown [**–f**] [**–R**] *owner* [*group*] *filename* . . .

DESCRIPTION

chown changes the owner of the *filename*s to *owner*. The owner may be either a decimal user ID (UID) or a login name found in the password file. An optional *group* may also be specified. The group may be either a decimal group ID (GID) or a group name found in the GID file.

Only the super-user of the machine where the file is physically located can change owner, in order to simplify accounting procedures.

OPTIONS

–f Do not report errors.

–R Recursively descend into directories setting the ownership of all files in each directory encountered. When symbolic links are encountered, their ownership is changed, but they are not traversed.

FILES

/etc/passwd password file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

chgrp(1), **chown**(2), **group**(4), **passwd**(4), **attributes**(5)

NAME

chroot – change root directory for a command

SYNOPSIS

/usr/sbin/chroot *newroot command*

DESCRIPTION

chroot causes *command* to be executed relative to *newroot*. The meaning of any initial slashes (**/**) in the path names is changed to *newroot* for *command* and any of its child processes. Upon execution, the initial working directory is *newroot*.

Notice that redirecting the output of *command* to a file:

```
chroot newroot command >x
```

will create the file **x** relative to the original root of *command*, not the new one.

The new root path name is always relative to the current root: even if a **chroot** is currently in effect, the *newroot* argument is relative to the current root of the running process.

This command can be run only by the super-user.

RETURN VALUES

The exit status of **chroot** is the return value of *command.*

EXAMPLE

chroot provides an easy way to extract **tar** files written with absolute filenames to a different location.

```
example# cp /usr/sbin/static/tar /tmp
example# dd if=/dev/nrst0 | chroot /tmp tar xvf -
```

Note that **tar** is statically linked, so you do not have to copy any shared libraries to the *newroot* filesystem.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cd(1), **chroot**(2), **attributes**(5)

NOTES

Exercise extreme caution when referencing device files in the new root file system.

NAME

clear_locks – clear locks held on behalf of an NFS client

SYNOPSIS

/usr/sbin/clear_locks [**–s**] *hostname*

DESCRIPTION

The **clear_locks** command removes all file, record, and share locks created by the *hostname* and held on the current host, regardless of which process created or owns the locks.

This command can be run only by the super-user.

This command should only be used to repair the rare case of a client crashing and failing to clear held locks. Clearing locks held by an active client may cause applications to fail in an unexpected manner.

OPTIONS

–s Remove all locks created by the current machine and held by the server *hostname*.

OPERANDS

hostname name of host server

EXIT STATUS

0	Successful operation.
1	If not root.
2	Usage error.
3	If unable to contact server (RPC).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fcntl(2), **attributes**(5)

NAME

clri, dcopy – clear inode

SYNOPSIS

clri [**–F** *FSType*] [**–V**] *special i-number*

dcopy [**–F** *FSType*] [**–V**] *special i-number*

DESCRIPTION

clri writes zeros on the inodes with the decimal *i-number* on the filesystem stored on *special*. After **clri**, any blocks in the affected file will show up as missing in an **fsck**(1M) of *special*.

Read and write permission is required on the specified file system device. The inode becomes allocatable.

The primary purpose of this routine is to remove a file which for some reason appears in no directory. If it is used to zap an inode which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated inode, so the whole cycle is likely to be repeated again and again.

dcopy is a symbolic link to **clri**.

OPTIONS

–F *FSType*　Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from **/etc/vfstab** by matching *special* with an entry in the table, or by consulting **/etc/default/fs**.

–V　Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from **/etc/vfstab**. This option should be used to verify and validate the command line.

USAGE

See **largefile**(5) for the description of the behavior of **clri** and **dcopy** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/default/fs	default local file system type
/etc/vfstab	list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fsck(1M), **vfstab**(4), **attributes**(5), **largefile**(5)

NOTES

This command may not be supported for all *FSTypes*.

NAME

conv_lp – convert LP configuration

SYNOPSIS

conv_lp [**–d** *dir*] [**–f** *file*]

DESCRIPTION

conv_lp reads LP printer configuration information from a directory and converts it to an output file for use with print client software.

OPTIONS

The following options are supported:

–d *dir* The root (‘ / ’) directory from which LP configuration information is read. The default is root (‘ / ’).

–f *file* The output file to which **conv_lp** writes the converted LP configuration information. The default is **/etc/printers.conf**.

EXAMPLES

The following example converts LP configuration information from directory root (‘ / ’) to file **/etc/printers.conf**.

```
example% conv_lp
```

The following example converts LP configuration information from directory **/export/root/client** to file **/export/root/client/etc/printers.conf**.

```
example% conv_lp -d /export/root/client -f /export/root/client/etc/printers.conf
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

FILES

/etc/printers.conf System printer configuration database.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

lpset(1M), **printers.conf**(4), **attributes**(5)

NAME

conv_lpd – convert LPD configuration

SYNOPSIS

conv_lpd [**–c printers** | **–c printcap**] [**–n**] *file*

DESCRIPTION

conv_lpd converts LPD printer configuration information from *file* to a **printers.conf** or a **printcap** file (see **printers.conf**(4)). *file* specifies the name of the input file, and can be either in **printers.conf** or **printcap** format. If *file* is in **printers.conf** format, it converts it to a **printcap** file. If *file* is in **printcap** format, it converts it to a **printers.conf** file.

OPTIONS

The following options are supported:

–c printers | **–c printcap**
Specifies the type of output file produced by the conversion. **–c printers** converts to a **printers.conf** file. **–c printcap** converts to a **printcap** file.
–c printers is the default.

–n Preserves the namelist during the conversion.

OPERANDS

The following operands are supported:

file The file to be converted.

EXAMPLES

The following example converts a **printcap** file to a **printers.conf** file.

```
example% conv_lpd /etc/printcap
```

The following example converts a **printcap** file to a **printers.conf** file and preserves the namelist.

```
example% conv_lpd -c printers -n /etc/printcap
```

The following example converts a **printers.conf** file to a **printcap** file and preserves the namelist.

```
example% conv_lpd -c printcap -n /etc/printers.conf
```

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

FILES

/etc/printers.conf	System printer configuration database.
/etc/printcap	SunOS 4.*x* printer capability database.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

lpset(1M), **printers.conf**(4), **attributes**(5)

NAME

crash – examine system images

SYNOPSIS

/usr/sbin/crash [–**d** *dumpfile*] [–**n** *namelist*] [–**w** *output-file*]

DESCRIPTION

The **crash** command is used to examine the system memory image of a running or a crashed system by formatting and printing control structures, tables, and other information. Command line arguments to **crash** are *dumpfile*, *namelist*, and *output-file*.

OPTIONS

–**d** *dumpfile* Specify *dumpfile* as the file containing the system memory image. The default *dumpfile* is **/dev/mem**. The system image can also be the pathname of a dump file generated by the **savecore**(1M) utility.

–**n** *namelist* Specify the text file *namelist* which contains the symbol table information needed for symbolic access to the system memory image to be examined. The default *namelist* is **/dev/ksyms**. Note: It is recommended that **crash** dumps be analyzed on a machine having the same kernel architecture as the machine from which the dump was taken.

–**w** *output-file* When the **crash** command is invoked, a session is initiated. The output from a **crash** session is directed to *output-file*. The default *output-file* is the standard output.

USAGE

Input during a **crash** session is of the form:

function [*argument*…]

where *function* is one of the **crash** functions described in the **Functions** subsection of this manual page, and *arguments* are qualifying data that indicate which items of the system image are to be printed.

The default for process-related items is the current process for a running system or the process that was running at the time of the crash for a crashed system. Similarly, the default for thread-related items is the current thread for a running system or the thread that was running at the time of the crash for a crash system. If the contents of a table are being dumped, the default is all active table entries.

Function Options

The following function options are available to **crash** functions wherever they are semantically valid. Valid function options are shown in **Functions**.

–**e** Display every entry in a table.

–**f** Display the full structure.

–**p** Interpret all address arguments in the command line as physical addresses. If the addresses specified are not physical addresses, results are inconsistent.

–**s** *process* Specify a process slot other than the default.

–**w** *filename* Redirect the output of a function to *filename*.

Output from **crash** functions may be piped to another program in the following way:

function [*argument*. . .] **!** *shell_command*

The redirection option –**w** cannot be used with this feature.

Depending on the context of the function, numeric arguments are assumed to be in a specific radix. Counts are assumed to be decimal. Addresses are always hexadecimal. Table address arguments larger than the size of the function table are interpreted as hexadecimal addresses; those smaller are assumed to be decimal slots in the table. Default bases on all arguments may be overridden. The C conventions for designating the bases of numbers are recognized. A number that is usually interpreted as decimal is interpreted as hexadecimal if it is preceded by **0x** and as octal if it is preceded by **0**. Decimal override is designated by **0d**, and binary by **0b**.

Aliases for functions may be any uniquely identifiable initial substring of the function name. Traditional aliases of one letter, such as **b** for **buffer**, remain valid.

Many functions accept different forms of entry for the same argument. Requests for table information accept a table entry number, a physical address, a virtual address, a symbol, a range, or an expression. A range of slot numbers may be specified in the form *a*–*b* where *a* and *b* are decimal numbers. An expression consists of two operands and an operator. An operand may be an address, a symbol, or a number; the operator may be **+**, –, *, /, **&**, or **|**. An operand that is a number should be preceded by a radix prefix if it is not a decimal number (**0** for octal, **0x** for hexadecimal, **0b** for binary). The expression must be enclosed in parentheses. Other functions accept any of these argument forms that are meaningful.

Two abbreviated arguments to **crash** functions are used throughout. Both accept data entered in several forms. They may be expanded into the following:

table_entry = *slot number* | *address* | *symbol* | *range* | *expression*

start_addr = *address* | *symbol* | *expression*

Functions

? [–**w** *filename*]
List available functions.

!*command* Escape to the shell and execute *command*.

base [–**w** *filename*] *number*. . .
Print *number* in binary, octal, decimal, and hexadecimal. A number in a radix other than decimal should be preceded by a prefix that indicates its radix as follows: **0x**, hexadecimal; **0**, octal; and **0b**, binary.

buffer [–**w** *filename*] [–*format*] *bufferslot*

buffer [–**w** *filename*] [–*format*] [–**p**] *start_addr*
Alias: **b**
Print the contents of a buffer in the designated format. The following *format* designations are recognized: –**b**, byte; –**c**, character; –**d**, decimal; –**x**, hexadecimal; –**o**, octal; and, –**i**, inode. If no format is given, the previous format is used. The default format at the beginning of a **crash** session is hexadecimal.

bufhdr [−**f**] [−**w** *filename*] [[−**p**] *table_entry*. . .]
Alias: **buf**
Print system buffer headers.

callout [−**l**] [−**w** *filename*]
Alias: **c**
Print the **callout** table. If the −**l** option is specified, the contents of the locks pertaining to the **callout** structure are also displayed.

class [−**w** *filename*] [*table_entry*. . .]
Print information about process scheduler classes.

help [−**w** *filename*] *function*. . .
Print a description of the named function, including syntax and aliases.

kmastat [−**w** *filename*]
Print kernel memory allocator statistics.

kmausers [−**e**] [−**f**] [−**w** *filename*] [*cachename*. . .]
Print the information about the medium and large users of the kernel memory allocator that have current memory allocations. The output consists of one entry for each unique stack trace specifying the total amount of memory and number of allocations that was made with that stack trace.

This function is only available if the kernel has the **KMF_AUDIT** flag set in **kmem_flags**. (See **NOTES** below.)

If one or more cache names (for example, **kmem_alloc_256**) are specified, the scan of memory usage is restricted to those caches. By default all caches are included.

If the −**e** option is used, the small users of the allocator are included. The small users are allocations that total less than 1024 bytes of memory or for which there are less than 10 allocations with the same stack trace.

If the −**f** option is used, the stack traces are printed for each individual allocation.

lck [−**e**] [−**w** *filename*] [[−**p**] *lock_addr*. . .]
Alias: **l**
Print record locking information. If the −**e** option is used or lock address arguments are given, the record lock list is printed. If no argument is entered, information on locks relative to UFS inodes is printed.

mblk [−**e**] [−**f**] [−**w** *filename*] [[−**p**] *table_entry*. . .]
Print allocated streams message block and data block headers.

mount [−**f**] [−**w** *filename*] [[−**p**] *table_entry*. . .]
Alias: **m**, **vfs**
Print information about mounted filename systems.

nm [−**w** *filename*] *symbol*. . .
Print value and type for the given symbol.

od [−**p**] [−**w** *filename*] [−*format*] [−*mode*] [−**s** *process*] *start_addr* [*count*]

Alias: **rd**

Print *count* values starting at *start_addr* in one of the following formats: character (−**c**), decimal (−**d**), hexadecimal (−**x**), octal (−**o**), ASCII (−**a**), or hexadecimal / character (−**h**), and one of the following modes: long (−**l**), short (−**t**), or byte (−**b**). The default mode for character and ASCII formats is byte; the default mode for decimal, hexadecimal, and octal formats is long. The format −**h** prints both hexadecimal and character representations of the addresses dumped; no mode needs to be specified. When format or mode is omitted, the previous value is used. At the start of a **crash** session, the format is hexadecimal and the mode is long. If no count is entered, **1** is assumed.

proc [−**e**] [−**f**] [−**l**] [−**w** *filename*] [[−**p**] [−**a**] *table_entry*. . . | *#procid*. . .]

proc [−**e**] [−**f**] [−**l**] [−**w** *filename*] [−**r**]

Alias: **p**

Print the process table. Process table information may be specified in two ways. First, any mixture of table entries and process IDs may be entered. Each process ID must be preceded by a **#**. Alternatively, process table information for runnable processes may be specified with the runnable option −**r**. If the −**l** option is specified, all relevant locking information is displayed.

snode [−**e**] [−**f**] [−**l**] [−**w** *filename*] [[−**p**] *table_entry*. . .]

Print information about open special filenames. If the −**l** option is specified, all relevant locking information is also displayed.

strstat [−**w** *filename*]

Print STREAMS statistics.

tsdptbl [−**w** *filename*] [*table_entry*. . .]

Print the time-sharing dispatcher parameter table. See **ts_dptbl**(4).

uinode [−**d**] [−**e**] [−**f**] [−**l**] [−**r**] [−**w** *filename*] [[−**p**] *table_entry*. . .]

Alias: **ui**

Print the UFS inode table. The −**d** option will list the address and i-number of all UFS inodes in use and on the free list. If the −**l** option is specified, all relevant locking information is also displayed. The −**r** option will display all free UFS inodes.

var [−**w** *filename*]

Alias: **v**

Print the tunable system parameters.

vfs [−**e**] [−**w** *filename*] [[−**p**] *address*. . .]

Alias: **m**, **mount**

Print information about mounted filename systems.

vfssw [−**f**] [−**w** *filename*] [[−**p**] *table_entry*. . .]

Alias: **fs**

Print information about configured filename system types.

vnode [−**w** *filename*] [−**l**] [−**p**] *vnode_addr*. . .
Print information about vnodes.

vtop [−**w** *filename*] [−**s** *process*] *start_addr*. . .
Print the physical address translation of the virtual address *start_addr*.

Large File Behavior

See **largefile**(5) for the description of the behavior of **crash** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

The following exit values are returned:

0 Successful completion.
1 An error has occurred.

FILES

/dev/mem	system image of currently running system
/dev/ksyms	system namelist

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

savecore(1M), **soconfig**(1M), **rt_dptbl**(4), **ts_dptbl**(4), **attributes**(5), **largefile**(5)

NOTES

Kernel core dumps should be examined on the same platform on which they were created.

The following steps are necessary in order to set **KMF_AUDIT** in **kmem_flags** (which is needed for the **kmausers** and **mblkusers** commands):

```
ok boot kadb -d
kadb: kernel/unix
kadb[0]: startup:b
kadb[0]: :c
kadb[0]: startup:b
kadb[0]: :c
kadb[0]: kmem_flags/W1
kadb[0]: :c
```

NAME

cron – clock daemon

SYNOPSIS

/usr/sbin/cron

DESCRIPTION

The **cron** command starts a process that executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in **crontab** files in the directory **/var/spool/cron/crontabs**. Users can submit their own **crontab** file using the **crontab**(1) command. Commands which are to be executed only once may be submitted using the **at**(1) command.

cron only examines **crontab** or **at** command files during its own process initialization phase and when the **crontab** or **at** command is run. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

Since **cron** never exits, it should be executed only once. This is done routinely through **/etc/rc2.d/S75cron** at system boot time. The file **/etc/cron.d/FIFO** is used (among other things) as a lock file to prevent the execution of more than one instance of **cron**.

cron captures the output of the job's **stdout** and **stderr** streams, and, if it is non-empty, mails the output to the user. If the job does not produce output, no mail is sent to the user (unless the job is an **at**(1) job and the **–m** option was specified when the job was submitted).

Setting cron Defaults

To keep a log of all actions taken by **cron**, **CRONLOG=YES** (by default) must be specified in the **/etc/default/cron** file. If **CRONLOG=NO** is specified, no logging is done. Keeping the log is a user configurable option since **cron** usually creates huge log files.

The **PATH** for **user** cron jobs can be set using **PATH=** in **/etc/default/cron**. The **PATH** for **root** cron jobs can be set using **SUPATH=** in **/etc/default/cron**. The security implications of setting **PATH** and **SUPATH** should be carefully considered.

Example **/etc/default/cron** file:

```
CRONLOG=YES
PATH=/usr/bin:/usr/ucb:
```

This example enables logging and sets the default **PATH** used by non-root jobs to **/usr/bin:/usr/ucb:**. Root jobs will continue to use **/usr/sbin:/usr/bin**.

/etc/cron.d/logchecker is a script that checks to see if the log file has exceeded the system ulimit. If so, the log file is moved to **/var/cron/olog**.

FILES

/etc/cron.d	main **cron** directory
/etc/cron.d/FIFO	used as a lock file
/etc/default/cron	contains cron default settings
/var/cron/log	**cron** history information
/var/spool/cron	spool area
/etc/cron.d/logchecker	moves log file to **/var/cron/olog** if log file exceeds system ulimit.
/etc/cron.d/queuedefs	queue description file for **at**, **batch**, and **cron**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

at(1), **crontab**(1), **sh**(1), **queuedefs**(4), **attributes**(5)

DIAGNOSTICS

A history of all actions taken by **cron** is stored in **/var/cron/log** and (possibly) **/var/cron/olog**.

NAME

dd – convert and copy a file

SYNOPSIS

/usr/bin/dd [*operand=value* ...]

DESCRIPTION

dd copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block sizes may be specified to take advantage of raw physical I/O. Sizes are specified in bytes; a number may end with **k**, **b**, or **w** to specify multiplication by 1024, 512, or 2, respectively. Numbers may also be separated by **x** to indicate multiplication.

dd will read the input one block at a time, using the specified input block size; it then will process the block of data actually returned, which could be smaller than the requested block size. **dd** will apply any conversions that have been specified and write the resulting data to the output in blocks of the specified output block size.

cbs is used only if **ascii**, **asciib**, **unblock**, **ebcdic**, **ebcdicb**, **ibm**, **ibmb**, or **block** conversion is specified. In the first two cases, **cbs** characters are copied into the conversion buffer, any specified character mapping is done, trailing blanks are trimmed, and a NEWLINE is added before sending the line to output. In the last three cases, characters up to NEWLINE are read into the conversion buffer and blanks are added to make up an output record of size **cbs**. ASCII files are presumed to contain NEWLINE characters. If **cbs** is unspecified or **0**, the **ascii**, **asciib**, **ebcdic**, **ebcdicb**, **ibm**, and **ibmb** options convert the character set without changing the input file's block structure; the **unblock** and **block** options become a simple file copy.

After completion, **dd** reports the number of whole and partial input and output blocks.

OPERANDS

The following operands are supported:

if=*file* Specify the input path; standard input is the default.

of=*file* Specify the output path; standard output is the default. If the **seek=***expr* conversion is not also specified, the output file will be truncated before the copy begins, unless **conv=notrunc** is specified. If **seek=***expr* is specified, but **conv=notrunc** is not, the effect of the copy will be to preserve the blocks in the output file over which **dd** seeks, but no other portion of the output file will be preserved. (If the size of the seek plus the size of the input file is less than the previous size of the output file, the output file will be shortened by the copy.)

ibs=*n* Specify the input block size in *n* bytes (default is **512**).

obs=*n* Specify the output block size in *n* bytes (default is **512**).

bs=*n* Set both input and output block sizes to *n* bytes, superseding **ibs=** and **obs=**. If no conversion other than **sync**, **noerror**, and **notrunc** is specified, each input block will be copied to the output as a single block without aggregating short blocks.

cbs=*n* Specify the conversion block size for **block** and **unblock** in bytes by *n* (default is **0**). If **cbs=** is omitted or given a value of **0**, using **block** or

unblock produces unspecified results.

This option is used only if ASCII or EBCDIC conversion is specified. For the **ascii** and **asciib** operands, the input is handled as described for the **unblock** operand except that characters are converted to ASCII before the trailing SPACE characters are deleted. For the **ebcdic**, **ebcdicb**, **ibm**, and **ibmb** operands, the input is handled as described for the **block** operand except that the characters are converted to EBCDIC or IBM EBCDIC after the trailing SPACE characters are added.

files=*n* Copy and concatenate *n* input files before terminating (makes sense only where input is a magnetic tape or similar device).

skip=*n* Skip *n* input blocks (using the specified input block size) before starting to copy. On seekable files, the implementation will read the blocks or seek past them; on non-seekable files, the blocks will be read and the data will be discarded.

iseek=*n* Seek *n* blocks from beginning of input file before copying (appropriate for disk files, where **skip** can be incredibly slow).

oseek=*n* Seek *n* blocks from beginning of output file before copying.

seek=*n* Skip *n* blocks (using the specified output block size) from beginning of output file before copying. On non-seekable files, existing blocks will be read and space from the current end-of-file to the specified offset, if any, filled with null bytes; on seekable files, the implementation will seek to the specified offset or read the blocks as described for non-seekable files.

count=*n* Copy only *n* input blocks.

conv=*value*[**,***value*. . .]

Where *value*s are comma-separated symbols from the following list:

ascii Convert EBCDIC to ASCII.

asciib Convert EBCDIC to ASCII using BSD-compatible character translations.

ebcdic Convert ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, set up a pipeline with **dd conv=unblock** beforehand.

ebcdicb Convert ASCII to EBCDIC using BSD-compatible character translations. If converting fixed-length ASCII records without NEWLINEs, set up a pipeline with **dd conv=unblock** beforehand.

ibm Slightly different map of ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, set up a pipeline with **dd conv=unblock** beforehand.

ibmb Slightly different map of ASCII to EBCDIC using BSD-compatible character translations. If converting fixed-length ASCII records without NEWLINEs, set up a pipeline with **dd**

conv=unblock beforehand.

The **ascii** (or **asciib**), **ebcdic** (or **ebcdicb**), and **ibm** (or **ibmb**) values are mutually exclusive.

block Treat the input as a sequence of NEWLINE-terminated or EOF-terminated variable-length records independent of the input block boundaries. Each record is converted to a record with a fixed length specified by the conversion block size. Any NEWLINE character is removed from the input line; SPACE characters are appended to lines that are shorter than their conversion block size to fill the block. Lines that are longer than the conversion block size are truncated to the largest number of characters that will fit into that size; the number of truncated lines is reported.

unblock Convert fixed-length records to variable length. Read a number of bytes equal to the conversion block size (or the number of bytes remaining in the input, if less than the conversion block size), delete all trailing SPACE characters, and append a NEWLINE character.

The **block** and **unblock** values are mutually exclusive.

lcase Map upper-case characters specified by the **LC_CTYPE** keyword **tolower** to the corresponding lower-case character. Characters for which no mapping is specified will not be modified by this conversion.

ucase Map lower-case characters specified by the **LC_CTYPE** keyword **toupper** to the corresponding upper-case character. Characters for which no mapping is specified will not be modified by this conversion.

The **lcase** and **ucase** symbols are mutually exclusive.

swab Swap every pair of input bytes. If the current input record is an odd number of bytes, the last byte in the input record is ignored.

noerror Do not stop processing on an input error. When an input error occurs, a diagnostic message will be written on standard error, followed by the current input and output block counts in the same format as used at completion. If the **sync** conversion is specified, the missing input will be replaced with null bytes and processed normally; otherwise, the input block will be omitted from the output.

notrunc Do not truncate the output file. Preserve blocks in the output file not explicitly written by this invocation of **dd**. (See also the preceding **of=***file* operand.)

sync Pad every input block to the size of the **ibs=** buffer, appending null bytes. (If either **block** or **unblock** is also specified, append SPACE characters, rather than null bytes.)

If operands other than **conv=** are specified more than once, the last specified **operand=***value* will be used.

For the **bs=**, **cbs=**, **ibs=**, and **obs=** operands, the application must supply an expression specifying a size in bytes. The expression, *expr*, can be:

1. a positive decimal number
2. a positive decimal number followed by **k**, specifying multiplication by 1024
3. a positive decimal number followed by **b**, specifying multiplication by 512
4. two or more positive decimal numbers (with or without **k** or **b**) separated by **x**, specifying the product of the indicated values.

All of the operands will be processed before any input is read.

USAGE

See **largefile**(5) for the description of the behavior of **dd** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

1. The following command:

```
example% dd if=/dev/rmt/0h  of=/dev/rmt/1h
```

copies from tape drive 0 to tape drive 1, using a common historical device naming convention.

2. The following command:

```
example% dd ibs=10  skip=1
```

strips the first 10 bytes from standard input.

3. This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the ASCII file **x**:

```
example% dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase
```

4. Use **conv=sync** when writing to a tape as in the following example:

```
example% tar cvf - . | compress | dd obs=1024k of=/dev/rmt/0 conv=sync
```

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **dd**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 The input file was copied successfully.

>0 An error occurred.

If an input error is detected and the **noerror** conversion has not been specified, any partial output block will be written to the output file, a diagnostic message will be written, and the copy operation will be discontinued. If some other error is detected, a diagnostic message will be written and the copy operation will be discontinued.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cp(1), **sed**(1), **tr**(1), **attributes**(5), **environ**(5), **largefile**(5)

DIAGNOSTICS

f+p records in(out) numbers of full and partial blocks read(written)

NOTES

Do not use **dd** to copy files between file systems having different block sizes.

Using a blocked device to copy a file will result in extra nulls being added to the file to pad the final block to the block boundary.

When **dd** reads from a pipe, using the **ibs=X** and **obs=Y** operands, the output will always be blocked in chunks of size Y. When **bs=Z** is used, the output blocks will be whatever was available to be read from the pipe at the time.

When using **dd** to copy files to a tape device, the file size must be a multiple of the device sector size (for example, 512 Kbyte). To copy files of arbitrary size to a tape device, use **tar**(1) or **cpio**(1).

For **SIGINT**, **dd** will write status information to standard error before exiting. It will take the standard action for all other signals.

NAME

deallocate – device deallocation

SYNOPSIS

deallocate [**–s**] *device*

deallocate [**–s**] [**–F**] *device*

deallocate [**–s**] **–I**

DESCRIPTION

deallocate deallocates a *device* allocated to the evoking user. *device* can be a device defined in **device_allocate**(4) or one of the device special files associated with the device. It resets the ownership and the permission on all device special files associated with *device,* disabling the user's access to that device. This option can be used by the super user to remove access to the device by another user.

When deallocation or forced deallocation is performed, the appropiate device cleaning program is executed, based on the contents of **device_allocate**(4). These cleaning programs are normally stored in **/etc/security/lib.**

OPTIONS

device Deallocate the device associated with the device special file specified by *device***.**

–s Silent. Suppress any diagnostic output.

–F *device* Force deallocation of the device associated with the file specified by *device*. Only the super user is permitted to use this option.

–I Force deallocation of all allocatable devices. Only the super user is permitted to use this option. This option should only be used at system initialization.

DIAGNOSTICS

deallocate returns an nonzero exit status in the event of an error.

FILES

/etc/security/device_allocate
/etc/security/device_maps
/etc/security/dev/*
/etc/security/lib/*

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

allocate(1M), **bsmconv**(1M), **device_allocate**(4), **device_maps**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

devattr – lists device attributes

SYNOPSIS

devattr [–**v**] *device* [*attribute* ...]

DESCRIPTION

devattr displays the values for a device's attributes. The display can be presented in two formats. Used without the –**v** option, only the attribute values are shown. Used with the –**v** option, the attributes are shown in an *attribute***=***value* format. When no attributes are given on the command line, all attributes for the specified device are displayed in alphabetical order by attribute name. If attributes are given on the command line, only those attributes are shown, displayed in command line order.

OPTIONS

–**v** Specifies verbose format. Attribute values are displayed in an *attribute***=***value* format.

device Defines the device whose attributes should be displayed. Can be the pathname of the device or the device alias.

attribute Defines which attribute, or attributes, should be shown. Default is to show all attributes for a device. See the **putdev**(1M) manual page for a complete listing and description of available attributes.

ERRORS

The command will exit with one of the following values:

0 = successful completion of the task.

1 = command syntax incorrect, invalid option used, or internal error occurred.

2 = device table could not be opened for reading.

3 = requested device could not be found in the device table.

4 = requested attribute not defined for specified device.

FILES

/etc/device.tab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

getdev(1M), **putdev**(1M), **attributes**(5)

NAME

devconfig – configure device attributes

SYNOPSIS

devconfig

DESCRIPTION

devconfig is an interactive editor for device driver hardware configuration files and the OpenWindows configuration file.

Devices that are not self-identifying require that correct information be supplied in the hardware configuration file in order for the device to be recognized. Devconfig is designed to ease the task of maintaining correct device information in the configuration files.

Prototype configuration information files stored in **/usr/lib/devconfig** are used to check user input to ensure that the values provided for each attribute are of the correct type and fall within legal bounds. See **device.cfinfo**(4) for a description of the format of configuration information files. The location for the **cfinfo** files can be set by the **DEVCONFIGHOME** environment variable.

After making changes to a hardware configuration file that has a driver associated with it, an attempt is made to reload the driver to verify the attributes. Some drivers may not be unloadable. In this case, a system reboot must be initiated before the new attributes can take effect. If necessary, **devconfig** also updates the OpenWindows configuration file, **OWconfig** (see the *OpenWindows Desktop Reference Manual*). **devconfig** makes a backup copy of a modified file in a **.bak** file. In addition, the first version of **OWconfig** is saved in **OWconfig.save**. This is because the original version of **OWconfig** contains helpful prototype information that may be referred to in case **OWconfig** needs to be edited manually.

If the default location for configuration files is not writable (as is the case during installation) **devconfig** writes the updated files in the same location relative to the directory **/tmp/root**. No attempt is made to reload the driver in this case.

devconfig is controlled by a simple menu system. The Up / Down arrow keys move the cursor to different items in a menu. The Left / Right arrow keys move the cursor to different items in a field. The Enter key selects an item. (Note that the Enter key may be labeled Return on some keyboards.) See the online help for more guidance.

Operation

devconfig first displays a list of configured devices in the system. Selecting a configured device allows you to view its attributes or unconfigure it. Self-identifying devices can not be unconfigured by **devconfig**.

When you add a new device, **devconfig** displays the supported device categories. After choosing a device category, **devconfig** displays the devices supported in that category. Self-identifying devices cannot be added with **devconfig** and they are not displayed in the list of the devices. After you have selected the device to be added, **devconfig** displays the list of the device attributes.
Once you have chosen the proper values for the attributes and applied them by using the Apply button, the device is added to the list of configured devices. You may cancel an operation by using the Cancel button.

FILES

/kernel/drv/*.conf	hardware configuration files
/usr/lib/devconfig/*.cfinfo	configuration information files
/usr/openwin/server/etc/OWconfig	network OpenWindows configuration file
/etc/openwin/server/etc/OWconfig	local OpenWindows configuration file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWo86u

SEE ALSO

drvconfig(1M), **prtconf**(1M), **device.cfinfo**(4), **attributes**(5)

OpenWindows Desktop Reference Manual

NAME

devfree – release devices from exclusive use

SYNOPSIS

devfree *key* [*device* ...]

DESCRIPTION

devfree releases devices from exclusive use. Exclusive use is requested with the command **devreserv**.

When **devfree** is invoked with only the *key* argument, it releases all devices that have been reserved for that *key*. When called with *key* and *device* arguments, **devfree** releases the specified devices that have been reserved with that *key*.

OPTIONS

key Designates the unique key on which the device was reserved.

device Defines device that this command will release from exclusive use. Can be the pathname of the device or the device alias.

ERRORS

The command will exit with one of the following values:

0 Successful completion of the task.

1 Command syntax incorrect, invalid option used, or internal error occurred.

2 Device table or device reservation table could not be opened for reading.

3 Reservation release could not be completely fulfilled because one or more of the devices was not reserved or was not reserved on the specified key.

FILES

/etc/device.tab
/etc/devlkfile

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

devreserv(1M), **attributes**(5)

NOTES

The commands **devreserv** and **devfree** are used to manage the availability of devices on a system. These commands do not place any constraints on the access to the device. They serve only as a centralized bookkeeping point for those who wish to use them. Processes that do not use **devreserv** may concurrently use a device with a process that has reserved that device.

NAME

devinfo – print device specific information

SYNOPSIS

/usr/sbin/devinfo **–i** *device*

/usr/sbin/devinfo **–p** *device*

DESCRIPTION

The **devinfo** command is used to print device specific information about disk devices on standard out. The command can only be used by the superuser.

OPTIONS

–i Prints the following device information:

- Device name
- Software version (not supported and prints as **0**)
- Drive id number (not supported and prints as **0**)
- Device blocks per cylinder
- Device bytes per block
- Number of device partitions with a block size greater than zero

–p Prints the following device partition information:

- Device name
- Device major and minor numbers (in hexadecimal)
- Partition start block
- Number of blocks allocated to the partition
- Partition flag
- Partition tag

This command is used by various other commands to obtain device specific information for the making of file systems and determining partition information. If the device cannot be opened, an error message is reported.

OPERANDS

device Device name.

EXIT STATUS

0 Successful operation.

2 Operation failed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

prtvtoc(1M), **attributes**(5)

NAME

devlinks – adds /dev entries for miscellaneous devices and pseudo-devices

SYNOPSIS

/usr/sbin/devlinks [**–d**] [**–r** *rootdir*] [**–t** *table-file*]

DESCRIPTION

devlinks creates symbolic links from the **/dev** directory tree to the actual block- and character-special device nodes under the **/devices** directory tree. The links are created according to specifications found in the *table-file* (by default **/etc/devlink.tab**).

devlinks is called each time the system is reconfiguration-booted, and can only be run after **drvconfig**(1M) is run, since **drvconfig**(1M) builds the kernel data structures and the **/devices** tree.

The *table-file* (normally **/etc/devlink.tab**) is an ASCII file, with one line per record. Comment lines, which must contain a hash character ('**#**') as their first character, are allowed. Each entry must contain at least two fields, but may contain three fields. Fields are separated by single TAB characters.

The fields are:

devfs-spec Specification of devinfo nodes that will have links created for them. This specification consists of one or more keyword-value pairs, where the keyword is separated from the value by an equal-sign ('**=**'), and keyword-value pairs are separated from one another by semicolons.

The possible keywords are:

type The devinfo device type. Possible values are specified in **ddi_create_minor_node**(9F)**.**

name The name of the node. This is the portion of the **/devices** tree entry name that occurs before the first '**@**' or '**:**' character.

addr[*n*] The address portion of a node name. This is the portion of a node name that occurs between the '**@**' and the '**:**' characters. It is possible that a node may have a name without an address part, which is the case for many of the pseudo-device nodes. If a number is given after the *addr* it specifies a match of a particular comma-separated subfield of the address field: *addr1* matches the first subfield, *addr2* matches the second, and so on. *addr0* is the same as *addr* and matches the whole field.

minor[*n*] The minor portion of a node name – the portion of the name after the '**:**'. As with *addr* above, a number after the *minor* keyword specifies a subfield to match.

Of these four specifications, only the *type* specification must always be present.

name Specification of the **/dev** links that correspond to the devinfo nodes. This field allows **devlinks** to determine matching **/dev** names for the **/devices** nodes it has found. The specification of this field uses escape-sequences to allow portions of the **/devices** name to be included in the **/dev** name, *or* to allow a counter to be used in creating node names. If a counter is used to create a name, the portion of the name before the counter must be specified absolutely, and all names in the **/dev/**-subdirectory that match (up to and including the counter) are considered to be subdevices of the same device. This means that they should all point to the same directory, **name** and **address** under the **/devices/**-tree

The possible escape-sequences are:

\D Substitute the **device-name** (name) portion of the corresponding devinfo node-name.

\A*n* Substitute the *n*th component of the **address** component of the corresponding devinfo node name. Sub-components are separated by commas, and sub-component **0** is the whole **address** component.

\M*n* Substitute the *n*th sub-component of the **minor** component of the corresponding devinfo node name. Sub-components are separated by commas, and sub-component **0** is the whole **minor** component.

\N*n* Substitute the value of a 'counter' starting at *n*. There can be only one counter for each dev-spec, and counter-values will be selected so they are as low as possible while not colliding with already-existing link names.

In a dev-spec the **counter** sequence should not be followed by a digit, either explicitly or as a result of another escape-sequence expansion. If this occurs, it would not be possible to correctly match already-existing links to their counter entries, since it would not be possible to unambiguously parse the already-existing **/dev**-name.

extra-dev-link Optional specification of an extra **/dev** link that points to the initial **/dev** link (specified in field 2). This field may contain a **counter** escape-sequence (as described for the *dev-spec* field) but may not contain any of the other escape-sequences. It provides a way to specify an alias of a particular **/dev** name.

OPTIONS

–d Debugging mode – print out all **devinfo** nodes found, and indicate what links would be created, but do not do anything.

–r *rootdir* Use *rootdir* as the root of the **/dev** and **/devices** directories under which the device nodes and links are created. Changing the root directory does not change the location of the **/etc/devlink.tab** default table, nor is the root directory applied to the filename supplied to the **–t** option.

–t *table-file* Set the table file used by **devlinks** to specify the links that must be created. If this option is not given, **/etc/devlink.tab** is used. This option gives a way to instruct **devlinks** just to perform a particular piece of work, since just the links-types that **devlinks** is supposed to create can be specified in a command-file and fed to **devlinks**.

ERRORS

If **devlinks** finds an error in a line of the *table-file* it prints a warning message on its standard output and goes on to the next line in the *table-file* without performing any of the actions specified by the erroneous rule.

If it cannot create a link for some filesystem-related reason it prints an error-message and continues with the current rule.

If it cannot read necessary data it prints an error message and continues with the next *table-file* line.

EXAMPLES

Example **/etc/devlink.tab** fields are:

```
type=pseudo;name=win        win\M0
type=ddi_display      framebuffer/\M0      fb\N0
```

The first example states that all devices of type **pseudo** with a name component of **win** will be linked to **/dev/win***x*, where *x* is the minor-component of the *devinfo-name* (this is always a single-digit number for the **win** driver).

The second example states that all devinfo nodes of type **ddi_display** will be linked to entries under the **/dev/framebuffer** directory, with names identical to the entire minor component of the **/devices** name. In addition an extra link will be created pointing from **/dev/fb***n* to the entry under **/dev/framebuffer**. This entry will use a counter to end the name.

FILES

/dev	entries for the miscellaneous devices for general use
/devices	device nodes
/etc/devlink.tab	the default rule-file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ddi_create_minor_node(9F), **disks**(1M), **drvconfig**(1M), **ports**(1M), **tapes**(1M), **attributes**(5)

BUGS

It is very easy to construct mutually-contradictory link specifications, or specifications that can never be matched. The program does not check for these conditions.

NAME

devnm – device name

SYNOPSIS

/usr/sbin/devnm *name* [*name*…]

DESCRIPTION

The **devnm** command identifies the special file associated with the mounted file system where the argument *name* resides. One or more *name* can be specified.

EXAMPLE

The command:

```
/usr/sbin/devnm /usr
```

produces:

```
/dev/dsk/c0t3d0s6 /usr
```

if **/usr** is mounted on **/dev/dsk/c0t3d0s6**.

FILES

/dev/dsk/*
/etc/mnttab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mnttab(4), **attributes**(5)

NAME

devreserv – reserve devices for exclusive use

SYNOPSIS

devreserv [*key* [*devicelist* ...]]

DESCRIPTION

devreserv reserves devices for exclusive use. When the device is no longer required, use **devfree** to release it.

devreserv reserves at most one device per *devicelist*. Each list is searched in linear order until the first available device is found. If a device cannot be reserved from each list, the entire reservation fails.

When **devreserv** is invoked without arguments, it lists the devices that are currently reserved and shows to which key it was reserved. When **devreserv** is invoked with only the *key* argument, it lists the devices that are currently reserved to that key.

OPTIONS

key Designates a unique key on which the device will be reserved. The key must be a positive integer.

devicelist Defines a list of devices that **devreserv** will search to find an available device. (The list must be formatted as a single argument to the shell.)

EXAMPLES

To reserve a floppy disk and a cartridge tape:

```
$ key=$$
$ echo "The current Process ID is equal to: $key"
The Current Process ID is equal to: 10658
$ devreserv $key diskette1 ctape1
```

To list all devices currently reserved:

```
$ devreserv
disk1       2423
diskette1   10658
ctape1      10658
```

To list all devices currently reserved to a particular key:

```
$ devreserv $key
diskette1
ctape1
```

ERRORS

The command will exit with one of the following values:

0 Successful completion of the task.

1 Command syntax incorrect, invalid option used, or internal error occurred.

2 Device table or device reservation table could not be opened for reading.

3 Device reservation request could not be fulfilled.

FILES

/etc/device.tab
/etc/devlkfile

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

devfree(1M), **attributes**(5)

NOTES

The commands **devreserv** and **devfree** are used to manage the availability of devices on a system. Their use is on a participatory basis and they do not place any constraints on the actual access to the device. They serve as a centralized bookkeeping point for those who wish to use them. To summarize, devices which have been reserved cannot be used by processes which utilize the device reservation functions until the reservation has been canceled. However, processes that do not use device reservation may use a device that has been reserved since such a process would not have checked for its reservation status.

NAME

df – report number of free disk blocks and files

SYNOPSIS

/usr/bin/df [–**F** *FSType*] [–**abegklntV**] [–**o** *FSType-specific_options*]
[*directory* | *block_device* | *resource* . . .]

/usr/xpg4/bin/df [–**F** *FSType*] [–**abegklnPtV**] [–**o** *FSType-specific_options*]
[*directory* | *block_device* | *resource* . . .]

DESCRIPTION

The **df** command displays the amount of disk space occupied by mounted or unmounted file systems, directories, or mounted resources, the amount of used and available space, and how much of the file system's total capacity has been used.

directory represents a valid directory name. If *directory* is specified, **df** reports on the file system that contains *directory*. *block_device* represents a block special device (for example, **/dev/dsk/c1d0s7**); if *block_device* is specified, the corresponding file system need not be mounted. *resource* is an NFS resource name.

Used without operands or options, **df** reports on all mounted file systems.

OPTIONS

The following options are supported for both **/usr/bin/df** and **/usr/xpg4/bin/df**:

–**a** Report on all filesystems including ones whose entries in **/etc/mnttab** (see **mnttab**(4)) have the **ignore** option set.

–**b** Print the total number of kilobytes free.

–**e** Print only the number of files free.

–**F** *FSType* Specify the *FSType* on which to operate. This is only needed if the file system is unmounted. The *FSType* should be specified here or be determinable from **/etc/vfstab** (see **vfstab**(4)) have the by matching the *directory*, *block_device*, or *resource* with an entry in the table, or by consulting **/etc/default/fs**. See **default_fs**(4).

–**g** Print the entire **statvfs**(2) structure. This option is used only for mounted file systems. It cannot be used with the –**o** option. This option will override the –**b**, –**e**, –**k**, –**n**, –**P**, and –**t** options.

–**k** Print the allocation in kbytes. The output consists of one line of information for each specified file system. This information includes the file system name, the total space allocated in the file system, the amount of space allocated to existing files, the total amount of space available for the creation of new files by unpriviledged users, and the percentage of normally available space that is currently allocated to all files on the file system. This option will override the –**b**, –**e**, –**n**, and –**t** options.

–**l** Report on local file systems only. This option is used only for mounted file systems. It cannot be used with the –**o** option.

–n Print only the *FSType* name. Invoked with no operands, this option prints a list of mounted file system types. This option is used only for mounted file systems. It cannot be used with the **–o** option.

–o *FSType-specific_options*
Specify *FSType-specific* options. These options are comma-separated, with no intervening spaces. See the manual page for the *FSType-specific* command for details.

–t Print full listings with totals. This option will override the **–b**, **–e**, and **–n** options.

–V Echo the complete set of file system specific command lines, but do not execute them. The command line is generated by using the options and operands provided by the user and adding to them information derived from **/etc/mnttab**, **/etc/vfstab**, or **/etc/default/fs**. This option may be used to verify and validate the command line.

/usr/xpg4/bin/df

The following option is supported for **/usr/xpg4/bin/df** only:

–P Same as **–k** except in 512-byte units.

OPERANDS

The following operands are supported:

directory represents a valid directory name. **df** reports on the file system that contains *directory*.

block_device represents a block special device (for example, **/dev/dsk/c1d0s7**); the corresponding file system need not be mounted.

resource represents an NFS resource name.

USAGE

See **largefile**(5) for the description of the behavior of **df** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

The following example writes portable information about the **/usr** file system:

```
example% /usr/xpg4/bin/df –P /usr
```

Assuming that **/usr/src** is part of the **/usr** file system, the following will do the same as the previous example:

```
example% /usr/xpg4/bin/df –P /usr/src
```

ENVIRONMENT

SYSV3 This variable is used to: override the default behavior of **df**, provide compatibility with INTERACTIVE UNIX System and SCO UNIX installation scripts, and thus should not be used in new scripts. (It is provided for compatibility only.)

When set, any header which normally displays “files” will now display “nodes”. See **environ**(5) for descriptions of the following environment variables that affect the execution of **df**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

FILES

/dev/dsk/* disk devices

/etc/default/fs default local file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: **LOCAL=ufs**

LOCAL: The default partition for a command if no FSType is specified.

/etc/mnttab mount table

/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

/usr/bin/df

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

/usr/xpg4/bin/df

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4

SEE ALSO

find(1), **mount**(1M), **statvfs**(2), **default_fs**(4), **mnttab**(4), **vfstab**(4), **attributes**(5), **environ**(5), **largefile**(5), **xpg4**(5)

Manual pages for the *FSType*-specific modules of **df**.

NOTES

The –**F** option is intended for use with unmounted file systems.

This command may not be supported for all *FSTypes*.

NAME

dfmounts – display mounted resource information

SYNOPSIS

dfmounts [**–F** *FSType*] [**–h**] [**–o** *specific_options*] [*restriction* ...]

DESCRIPTION

dfmounts shows the local resources shared through a distributed file system *FSType* along with a list of clients that have the resource mounted. If *restriction* is not specified, **dfmounts** shows file systems that are currently shared on any NFS server. *specific_options* as well as the availability and semantics of *restriction* are specific to particular distributed file system types.

If **dfmounts** is entered without arguments, all remote resources currently mounted on the local system are displayed, regardless of file system type.

dfmounts Output

The output of **dfmounts** consists of an optional header line (suppressed with the **–h** flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server pathname clients ...

where:

resource Specifies the resource name that must be given to the **mount**(1M) command.

server Specifies the system from which the resource was mounted.

pathname Specifies the pathname that must be given to the **share**(1M) command.

clients Is a comma-separated list of systems that have mounted the resource. Clients are listed in the form *domain.*, *domain.system*, or *system*, depending on the file system type.

A field may be null. Each null field is indicated by a hyphen (–) unless the remainder of the fields on the line are also null; in which case, the hyphen may be omitted.

Fields with whitespace are enclosed in quotation marks (" ").

OPTIONS

–F *FSType* Specify filesystem type. Defaults to the first entry in **/etc/dfs/fstypes**.

Note: currently the only valid *FSType* is **nfs**.

–h Suppress header line in output.

–o *specific_options* Specify options specific to the filesystem provided by the **–F** option.

Note: currently no options are supported.

FILES

/etc/dfs/fstypes file system types

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

dfshares(1M), **mount**(1M), **share**(1M), **unshare**(1M), **attributes**(5)

NAME

dfmounts_nfs – display mounted NFS resource information

SYNOPSIS

dfmounts [–**F nfs**] [–**h**] [*server* ...]

DESCRIPTION

dfmounts shows the local resources shared through NFS, along with the list of clients that have mounted the resource. The –**F** flag may be omitted if NFS is the only file system type listed in the file **/etc/dfs/fstypes**.

dfmounts without options, displays all remote resources mounted on the local system, regardless of file system type.

The output of **dfmounts** consists of an optional header line (suppressed with the –**h** flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server pathname clients ...

where

resource Does not apply to NFS. Printed as a hyphen (-).

server Specifies the system from which the resource was mounted.

pathname Specifies the pathname that must be given to the **share**(1M) command.

clients Is a comma-separated list of systems that have mounted the resource.

OPTIONS

–**F nfs** Specifies the **nfs-**FSType.

–**h** Suppress header line in output.

server Displays information about the resources mounted from each server, where *server* can be any system on the network. If no server is specified, the *server* is assumed to be the local system.

FILES

/etc/dfs/fstypes

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mount(1M), **share**(1M), **unshare**(1M), **attributes**(5)

NAME

dfshares – list available resources from remote or local systems

SYNOPSIS

dfshares [**–F** *FSType*] [**–h**] [**–o** *specific_options*] [*server* . . .]

DESCRIPTION

dfshares provides information about resources available to the host through a distributed file system of type *FSType*. *specific_options* as well as the semantics of *server* are specific to particular distributed file systems.

If **dfshares** is entered without arguments, all resources currently shared on the local system are displayed, regardless of file system type.

The output of **dfshares** consists of an optional header line (suppressed with the **–h** flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server access transport

where

resource Specifies the resource name that must be given to the **mount**(1M) command.

server Specifies the name of the system that is making the resource available.

access Specifies the access permissions granted to the client systems, either **ro** (for read-only) or **rw** (for read / write). If **dfshares** cannot determine access permissions, a hyphen (–) is displayed.

transport Specifies the transport provider over which the resource is shared.

A field may be null. Each null field is indicated by a hyphen (–) unless the remainder of the fields on the line are also null; in which case, the hyphen may be omitted.

OPTIONS

–F *FSType* Specify filesystem type. Defaults to the first entry in **/etc/dfs/fstypes**.

–h Suppress header line in output.

–o *specific_options* Specify options specific to the filesystem provided by the **–F** option.

FILES

/etc/dfs/fstypes

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

dfmounts(1M), **mount**(1M), **share**(1M), **unshare**(1M), **attributes**(5)

NAME

dfshares_nfs – list available NFS resources from remote systems

SYNOPSIS

dfshares [–**F nfs**] [–**h**] [*server* ...]

DESCRIPTION

dfshares provides information about resources available to the host through NFS. The –**F** flag may be omitted if NFS is the first file system type listed in the file **/etc/dfs/fstypes**.

The query may be restricted to the output of resources available from one or more servers.

dfshares without arguments displays all resources shared on the local system, regardless of file system type.

Specifying *server* displays information about the resources shared by each server. *Server* can be any system on the network. If no server is specified, then *server* is assumed to be the local system.

The output of **dfshares** consists of an optional header line (suppressed with the –**h** flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server access transport

where

resource Specifies the resource name that must be given to the **mount**(1M) command.

server Specifies the system that is making the resource available.

access Specifies the access permissions granted to the client systems; however, **dfshares** cannot determine this information for an NFS resource and populates the field with a hyphen (-).

transport Specifies the transport provider over which the *resource* is shared; however, **dfshares** cannot determine this information for an NFS resource and populates the field with a hyphen (-).

A field may be null. Each null field is indicated by a hyphen (-) unless the remainder of the fields on the line are also null; in which case, the hyphen may be omitted.

OPTIONS

–**F nfs** Specify the NFS file system type

–**h** Suppress header line in output.

FILES

/etc/dfs/fstypes

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO **mount**(1M), **share**(1M), **unshare**(1M), **attributes**(5)

NAME

df_ufs – report free disk space on ufs file systems

SYNOPSIS

df –**F ufs** [*generic_options*] [−**o i**] [*directory* | *special*]

DESCRIPTION

df displays the amount of disk space occupied by **ufs** file systems, the amount of used and available space, and how much of the file system's total capacity has been used.

Note that the amount of space reported as used and available is less than the amount of space in the file system; this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this may be adjusted using **tunefs**(1M). When all the space on the file system except for this reserve is in use, only the super-user can allocate new files and data blocks to existing files. When the file system is overallocated in this way, **df** may report that the file system is more than 100% utilized.

If neither *directory* nor *special* is specified, **df** displays information for all mounted ufs file systems.

OPTIONS

generic_options Options supported by the generic **df** command. See **df**(1M) for a description of these options.

−**o** Specify **ufs** file system specific options. The available option is:

i Report the number of used and free inodes. This option may not be used with *generic_options*.

FILES

/etc/mnttab list of file systems currently mounted

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu, SUNWxcu4

SEE ALSO

df(1M), **tunefs**(1M), **fs_ufs**(4), **mnttab**(4), **attributes**(5)

NOTES

df calculates its results differently for mounted and unmounted file systems. For unmounted systems, the numbers reflect the 10% reservation mentioned above; this reservation is not reflected in **df** output for mounted file systems. For this reason, the available space reported by the generic command may differ from the available space reported by this module.

NAME

dhcpagent – daemon for client Dynamic Host Configuration Protocol (DHCP)

SYNOPSIS

dhcpagent [–**d***n*] [–**f**] [–**l***n*]

DESCRIPTION

dhcpagent implements the client half of the DHCP protocol for workstations with a local disk.

The DHCP protocol, among other things, permits a client to establish an endpoint for communication with a network by delivering an IP address of one or more of the client's network interfaces, and a "lease" on that address. The lease specifies the interval that the address remains valid; it may be infinite or of fixed duration. If it appears that the client wishes to continue using the IP address after its expiration, the DHCP protocol must negotiate an extension. For this reason, the DHCP client code must run as a daemon, only terminating when the client powers down.

Communication with the **dhcpagent** daemon is effected through the agency of **ifconfig**(1M) in much the same way that the **init**(1M) daemon is controlled by **telinit**(1M). **dhcpagent** may be invoked as a user process (albeit one requiring root privileges), but this is not necessary, as **ifconfig**(1M) will start it implicitly.

When started, **dhcpagent** enters a passive state while awaiting instructions to be passed to it by **ifconfig**(1M). When a command is received to configure an interface, the DHCP protocol is started. If successful, the interface is configured and brought up. The configuration received is stored in a file named *interface***.dhc** located under the directory **/etc/dhcp**. The agent will then sleep until it needs to renew the lease, which will happen well before the lease expires. Upon wakeup, if the interface is found to be down or to have a different IP address, **dhcpagent** considers that the interface is no longer under its control and will drop it from future consideration, until a further explicit request arrives from **ifconfig**(1M). If the lease cannot be renewed, **dhcpagent** will take the interface down when it expires, as required by the DHCP protocol. The user should consult *Dynamic Host Configuration Protocol* (RFC 2131) for details.

The DHCP protocol also acts as a mechanism to configure other information needed by the client (for example, name domain, addresses of routers, and so forth). The agent does not directly configure the workstation with this information but instead acts as a database which may be interrogated by other programs, and in particular by **dhcpinfo**(1). This approach is more flexible; it may eventually allow third party software access to the data through a published API, and allows system administrators more control over client configuration by editing startup scripts to permit various aspects of the client and its software to be customized in a specific order.

On clients with a single interface, this is quite straightforward. Clients with multiple interfaces may present difficulties, as there exists the possibility that some information arriving on different interfaces may need to be merged, or indeed that it may be inconsistent. Furthermore, the configuration of the interfaces is asynchronous, so requests may arrive while some or all of the interfaces are still unconfigured. The agent resolves these problems by permitting one interface, referred to as the *primary*, to be designated as special in that, in the absence of any other information, it is the only interface to be regarded as authoritative for global parameters. If a request for the value of the parameter is received which is not contained in the repository for the primary interface, the request fails. **dhcpinfo**(1) allows this behavior to be overridden by insisting that the global data sought be associated with a particular interface. (See **dhcpinfo**(1) for details.)

Messages

dhcpagent writes informational and error messages in four categories: errors, warnings, log, and debug. Errors are severe, usually unrecoverable, events due to resource exhaustion and other unexpected failure of system calls. An error is also generated if the client's lease on an IP address is in imminent danger of expiring. Warnings are less severe, and in most cases describe unusual or incorrect datagrams received from servers, or requests for service that cannot be provided. Informatory messages simply provide a human readable transcription of (correct) actions performed by the agent. Debug messages, if **dhcpagent** was built to generate them, may be generated at various levels of verbosity from **0** (not at all) through **9**, as controlled by the **−d** option. They are chiefly of benefit to persons having access to source code.

The disposition of messages is (by default) as follows: warning, log, and debug messages are discarded; errors are written to **/dev/console** and are sent to the system logger **syslog**(3) at priority **LOG_ERR** and with a facility identifier, **LOG_DAEMON**. If warnings have been enabled, they also are written to the system console and **syslog** with the same facility, but at priority **LOG_WARNING**. The creation and disposition of messages is controlled by the −**d**, −**f**, and −**l** command line flags (see **OPTIONS**) and the environment variable **DHCPLOG**. When present, **DHCPLOG** should name a file to which messages are sent in preference to the system console. Note that until the root file system is mounted read-write, no ordinary file can be used for this purpose.

OPTIONS

−**d***n* Set debug level to *n*. If debug is turned on, log messages are also enabled.

−**f** Run in the foreground instead of as a daemon process.

−**l***n* Enable warning (*n* > 0) and log (*n* > 1) messages. If *n* is not explicitly given, the value **1** is assumed, that is, warnings are turned on.

FILES

dhcpagent writes the configurations received into files in the directory **/etc/dhcp**.

*interface***.dhc** Contains the configuration for *interface*. The mere existence of this file does not imply that the configuration is correct, since the lease may have expired.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

dhcpinfo(1), **ifconfig**(1M), **init**(1M), **syslog**(3), **attributes**(5)

Droms, R., *Dynamic Host Configuration Protocol*, RFC 2131, Bucknell University, March 1997.

DIAGNOSTICS

Receipt of **SIGUSR1** signals **dhcpagent** to dump to **stdout** the contents of its scheduling table and the status of each of the interfaces under its control.

NAME

dhcpconfig – DHCP service configuration utility

SYNOPSIS

/usr/sbin/dhcpconfig

DESCRIPTION

General Description

dhcpconfig is a Korn shell (ksh) front-end to the DHCP table administration utilities **dhtadm**(1M) and **pntadm**(1M). It enables and configures the DHCP server service on the machine on which it is run. **dhcpconfig** displays the following menu:

```
***          DHCP Configuration          ***

Would you like to:

    1) Configure DHCP Service
    2) Configure BOOTP Relay Agent
    3) Unconfigure DHCP or Relay Service
    4) Exit

Choice:
```

After selecting one of the menu choices at the **Choice** line, you will be prompted to answer a series of questions concerning your choice, with recommended defaults. The menu choices are explained in more detail below:

1) Configure DHCP service
This configures the DHCP service, including setting startup options, such as **OFFER** timeout, **dhcptab** rescan interval, and enabling BOOTP compatibility mode, as well as bootstrapping **dhcptab** configuration data and producing the appropriate **dhcp network** tables.

2) Configure BOOTP Relay Agent
In this mode, no DHCP service databases are required. You are prompted for a list of BOOTP and / or DHCP servers to which the relay agent is to forward BOOTP / DHCP requests.

3) Unconfigure DHCP or Relay Service
This option restores the DHCP service to an uninitialized state. This option should be used with extreme caution, since the DHCP tables for the BOOTP / DHCP service are removed. This is particularly the case if the resource type you are using is **nisplus**, since other DHCP servers may be using this information.

Note that **dhcpconfig** can be run over and over again. Parameters are merged with existing parameters. Thus **dhcpconfig** can be used to synchronize the **dhcptab** configuration table with the server machine's settings.

How DHCP Tables Are Bootstrapped

dhcpconfig scans various configuration files on your Solaris machine for information it can use to populate the **dhcptab** configuration table. The following table lists the information and source used for this information:

Information	*Source*
Timezone	System date, timezone settings
DNS parameters	**nsswitch.conf**, **/etc/resolv.conf**
NIS parameters	system domainname, **nsswitch.conf**, NIS
NIS+ parameters	system domainname, **nsswitch.conf**, NIS+
Default router	system routing tables, user prompt.
Subnetmask	network interface, netmasks table in nameservice
broadcast address	network interface, user prompt.

If you have not set these parameters on your server machine, you should do so before running **dhcpconfig**. Otherwise, you will need to rerun **dhcpconfig** to pick up any changes and merge them with your **dhcptab** configuration table.

Serving BOOTP Clients

If you would like to configure the DHCP service to serve BOOTP clients, you will need to add the appropriate DHCP daemon startup options, as well as allocate IP addresses for your BOOTP clients.

Run **dhcpconfig** and select menu choice **1) Configure DHCP Service**. Descend into the "DHCP server daemon option setup" section, answering "Yes" when prompted for enabling BOOTP compatibility mode.

You will next be prompted for whether or not you would like the DHCP server to automatically allocate BOOTP-only IP addresses. If you answer "Yes", be sure to enter the "Select Networks For BOOTP / DHCP Support" section, and add additional IP addresses to the appropriate **dhcp network** tables. You will later be prompted whether you would like some (or all) of these addresses reserved for BOOTP clients. BOOTP IP addresses for automatic allocation are treated separately from DHCP addresses to prevent competition between BOOTP and DHCP clients for the same pool of addresses.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWdhcsu

SEE ALSO

dhtadm(1M), **in.dhcpd**(1M), **pntadm**(1M), **dhcp**(4), **dhcp_network**(4), **dhcptab**(4), **nsswitch.conf**(4), **resolv.conf**(4), **attributes**(5)

NAME

dhtadm – DHCP configuration table management utility

SYNOPSIS

dhtadm –**C** [–**r** *resource*] [–**p** *path*]
dhtadm –**A** –**s** *symbol_name* –**d** *definition* [–**r** *resource*] [–**p** *path*]
dhtadm –**A** –**m** *macro_name* –**d** *definition* [–**r** *resource*] [–**p** *path*]
dhtadm –**M** –**s** *symbol_name* –**d** *definition* [–**r** *resource*] [–**p** *path*]
dhtadm –**M** –**m** *macro_name* –**n** *new_name* [–**r** *resource*] [–**p** *path*]
dhtadm –**M** –**s** *symbol_name* –**d** *definition* [–**r** *resource*] [–**p** *path*]
dhtadm –**M** –**m** *macro_name* –**d** *definition* [–**r** *resource*] [–**p** *path*]
dhtadm –**M** –**m** *macro_name* –**e** *symbol=value* [–**r** *resource*] [–**p** *path*]
dhtadm –**D** –**s** *symbol_name* [–**r** *resource*] [–**p** *path*]
dhtadm –**D** –**m** *macro_name* [–**r** *resource*] [–**p** *path*]
dhtadm –**P** [–**r** *resource*] [–**p** *path*]
dhtadm –**R** [–**r** *resource*] [–**p** *path*]

DESCRIPTION

The **dhtadm** command manages the DHCP service configuration table, **dhcptab**. (For a description of the table format, see **dhcptab**(4).) One of the following option flags must be specified: –**C**, –**A**, –**M**, –**D**, or –**R**.

Depending on the resource type (–**r** option), the user must have the proper file permissions or NIS+ credentials.

OPTIONS

–**A** Add a symbol or macro definition to the **dhcptab** table. Required sub-options are:

Option	Argument	Description
–**d**	*definition*	Macro or symbol definition
–**m**	*macro_name*	Name of macro to be added
–**s**	*symbol_name*	Name of symbol to be added

–**C** Create the DHCP service configuration table, **dhcptab**.

–**D** Delete a symbol or macro definition. Required sub-options are:

Options	Argument	Description
–**m**	*macro_name*	Macro to delete
–**s**	*symbol_name*	Symbol to delete

–M Modify an existing symbol or macro definition. Required sub-options are:

Option	Argument	Description
–d	*definition*	Macro (–**m**) or symbol (–**s**) definition
–e	*symbol=value*	Edit symbol / value pair within a macro. To add a symbol which does not have an associated value, enter: "*symbol*=**_NULL_VALUE_**" To delete a symbol definition from a macro, enter: "*symbol*="
–m	*macro_name*	Macro; legal companions: one of –**n**, –**d**, or –**e**
–n	*new_name*	New macro name
–s	*symbol_name*	Symbol; legal companions: –**d**

–p *path* Override the **/etc/default/dhcp** configuration value for resource path. The resource path for the **files** resource is an absolute UNIX pathname, and a fully specified **nisplus** directory (including the tailing period) for the NIS+ resource. See **dhcp**(4) for more details.

–P Display the **dhcptab** table.

–r *resource* Override the **/etc/default/dhcp** configuration value for resource type. Currently supported resource types are **files** or **nisplus**. See **dhcp**(4).

–R Remove the **dhcptab** table.

EXAMPLES

1) The following command creates the DHCP service configuration table, **dhcptab**:

```
# dhtadm –C
```

2) The following command adds a **Vendor** option symbol definition for a new symbol called **MySym** to the **dhcptab** table in the **files** resource in the **/var/mydhcp** directory:

```
# dhtadm –A –s MySym –d ’Vendor=SUNW.PCW.LAN,20,IP,1,0’ \
                         –r files –p /var/mydhcp
```

3) The following command adds the **aruba** macro definition to the **dhcptab** table. Note that symbol / value pairs are bracketed with colons (**:**).

```
# dhtadm –A –m aruba –d ’:Timeserv=10.0.0.10 10.0.0.11:DNSserv=10.0.0.1:’
```

4) The following command modifies the **Locale** macro definition, setting the value of the **UTCOffst** symbol to 18000 seconds. Note that any macro definition which includes the definition of the **Locale** macro will inherit this change.

```
# dhtadm –M –m Locale –e ’UTCOffst=18000’
```

5) The following command deletes the **Timeserv** symbol from the **aruba** macro. Note that any macro definition which includes the definition of the **aruba** macro will inherit

this change.

```
# dhtadm –M –m aruba –e 'Timeserv='
```

6) The following command adds the **Hostname** symbol to the **aruba** macro. Note that the **Hostname** symbol takes no value, and thus requires the special value **_NULL_VALUE_**. Note also that any macro definition which includes the definition of the **aruba** macro will inherit this change.

```
# dhtadm –M –m aruba –e 'Hostname=_NULL_VALUE_'
```

7) The following command renames the **Locale** macro to **MyLocale**. Note that any **Include** statements in macro definitions which include the **Locale** macro will also need to be changed.

```
# dhtadm –M –m Locale –n MyLocale
```

8) The following command deletes the **MySym** symbol definition. Note that any macro definitions which use **MySym** will need to be modified.

```
# dhtadm –D –s MySym
```

9) The following command removes the **dhcptab** table in the **nisplus** directory specified.

```
# dhtadm –R –r nisplus –p Test.Nis.Plus.
```

EXIT STATUS

0	Successful completion.
1	Object already exists.
2	Object does not exist.
3	Non-critical error.
4	Critical error.

FILES

/var/dhcp/dhcptab	file or NIS+ table
/etc/default/dhcp	DHCP service configuration file
/etc/inet/hosts	file or NIS+ table

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu

SEE ALSO

dhcpconfig(1M), **dhcp**(4), **dhcp_network**(4), **dhcptab**(4), **hosts**(4), **attributes**(5)

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 1533, Lachman Technology, Inc., Bucknell University, October 1993.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 1541, Bucknell University, October 1993.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

NAME

disks – creates /dev entries for hard disks attached to the system

SYNOPSIS

/usr/sbin/disks [**–r** *rootdir*]

DESCRIPTION

disks creates symbolic links in the **/dev/dsk** and **/dev/rdsk** directories pointing to the actual disk device special files under the **/devices** directory tree. It performs the following steps:

1. **disks** searches the kernel device tree to see what hard disks are attached to the system. It notes the **/devices** pathnames for the slices on the drive and determines the physical component of the corresponding **/dev/dsk** or **/dev/rdsk** name.
2. The **/dev/dsk** and **/dev/rdsk** directories are checked for disk entries – that is, symbolic links with names of the form **c***N*[**t***N*]**d***N***s***N*, or **c***N*[**t***N*]**d**N**p***N*, where *N* represents a decimal number. **c***N* is the logical controller number, an arbitrary number assigned by this program to designate a particular disk controller. The first controller found on the first occasion this program is run on a system, is assigned number **0**. **t***N* is the bus-address number of a subsidiary controller attached to a peripheral bus such as SCSI or IPI (the **target** number for SCSI, and the **facility** number for IPI controllers). **d***N* is the number of the disk attached to the controller. **s***N* is the *slice* number on the disk. **p***N* is the FDISK partition number used by **fdisk**(1M). (x86 Only)
3. If only some of the disk entries are found in **/dev/dsk** for a disk that has been found under the **/devices** directory tree, disks creates the missing symbolic links. If none of the entries for a particular disk are found in **/dev/dsk**, **disks** checks to see if any entries exist for other disks attached to the same controller, and if so, creates new entries using the same controller number as used for other disks on the same controller. If no other **/dev/dsk** entries are found for slices of disks belonging to the same physical controller as the current disk, **disks** assigns the lowest-unused controller number and creates entries for the disk slices using this newly-assigned controller number.

disks is run automatically each time a reconfiguration-boot is performed or when **add_drv**(1M) is executed. When invoking **disks**(1M) manually, first run **drvconfig**(1M) to ensure **/devices** is consistent with the current device configuration.

Notice to Driver Writers

disks considers all devices with a node type of **DDI_NT_BLOCK**, **DDI_NT_BLOCK_CHAN, DDI_NT_CD**, **DDI_NT_BLOCK_WWN** or **DDI_NT_CD_CHAN** to be disk devices. **disks**(1M) requires the minor name of disk devices obey the following format conventions.

The minor name for block interfaces consists of a single lowercase ASCII character, **a** through **u**. The minor name for character (raw) interfaces consists of a single lowercase ASCII character, **a** through **u**, followed by **,raw**.

disks translates **a** through **p** to **s0** through **s15**, while it translates **q** through **u** to **p0** through **p4**. SPARC drivers should only use the first 8 slices: **a** through **h**, while x86 drivers can use **a** through **u**, with **q** through **u** corresponding to **fdisk**(1M) partitions. **q** represents the entire disk, while **r**, **s**, **t**, and **u** represent up to 4 additional partitions.

To prevent **disks** from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using a node type: **DDI_NT_BLOCK**, **DDI_NT_BLOCK_CHAN, DDI_NT_CD**, or **DDI_NT_CD_CHAN** when calling **ddi_create_minor_node**(9F).

OPTIONS

–r *rootdir* Causes **disks** to presume that the **/dev/dsk**, **/dev/rdsk** and **/devices** directory trees are found under *rootdir*, not directly under **/**.

ERRORS

If disks finds entries of a particular logical controller linked to different physical controllers, it prints an error message and exits without making any changes to the **/dev** directory, since it cannot determine which of the two alternative logical-to-physical mappings is correct. The links should be manually corrected or removed before another reconfiguration-boot is performed.

EXAMPLES

The following example demonstrates creating the block and character minor devices from within the **xkdisk** driver's **attach**(9E) function.

```
#include    <sys/dkio.h>

/*
 * Create the minor number by combining the instance number
 * with the slice number.
 */
#define MINOR_NUM(i, s)                              ((i) << 4 | (s))

int
xkdiskattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
   int instance, slice;
   char name[8];

          /* other stuff in attach... */

   instance = ddi_get_instance(dip);
   for (slice = 0; slice < V_NUMPAR; slice++) {
      /*
       * create block device interface
       */
      sprintf(name, "%c", slice + 'a');
      ddi_create_minor_node(dip, name, S_IFBLK,
         MINOR_NUM(instance, slice), DDI_NT_BLOCK_CHAN, 0);

      /*
       * create the raw (character) device interface
       */
      sprintf(name,"%c,raw", slice + 'a');
```

```
            ddi_create_minor_node(dip, name, S_IFCHR,
              MINOR_NUM(instance, slice), DDI_NT_BLOCK_CHAN, 0);
        }
    }
```

Installing the **xkdisk** disk driver on a SPARCstation 20, with the driver controlling a SCSI disk (target 3 attached to an **esp**(7D) SCSI HBA) and performing a reconfiguration-boot (causing disks to be run) creates the following special files in **/devices**.

```
# ls -l /devices/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/

brw-r-----   1 root sys   32, 16 Aug 29 00:02 xkdisk@3,0:a
crw-r-----   1 root sys   32, 16 Aug 29 00:02 xkdisk@3,0:a,raw
brw-r-----   1 root sys   32, 17 Aug 29 00:02 xkdisk@3,0:b
crw-r-----   1 root sys   32, 17 Aug 29 00:02 xkdisk@3,0:b,raw
brw-r-----   1 root sys   32, 18 Aug 29 00:02 xkdisk@3,0:c
crw-r-----   1 root sys   32, 18 Aug 29 00:02 xkdisk@3,0:c,raw
brw-r-----   1 root sys   32, 19 Aug 29 00:02 xkdisk@3,0:d
crw-r-----   1 root sys   32, 19 Aug 29 00:02 xkdisk@3,0:d,raw
brw-r-----   1 root sys   32, 20 Aug 29 00:02 xkdisk@3,0:e
crw-r-----   1 root sys   32, 20 Aug 29 00:02 xkdisk@3,0:e,raw
brw-r-----   1 root sys   32, 21 Aug 29 00:02 xkdisk@3,0:f
crw-r-----   1 root sys   32, 21 Aug 29 00:02 xkdisk@3,0:f,raw
brw-r-----   1 root sys   32, 22 Aug 29 00:02 xkdisk@3,0:g
crw-r-----   1 root sys   32, 22 Aug 29 00:02 xkdisk@3,0:g,raw
brw-r-----   1 root sys   32, 23 Aug 29 00:02 xkdisk@3,0:h
crw-r-----   1 root sys   32, 23 Aug 29 00:02 xkdisk@3,0:h,raw
```

/dev/dsk will contain the disk entries to the block device nodes in **/devices**

```
# ls -l /dev/dsk
/dev/dsk/c0t3d0s0 -> ../../devices/[...]/xkdisk@3,0:a
/dev/dsk/c0t3d0s1 -> ../../devices/[...]/xkdisk@3,0:b
/dev/dsk/c0t3d0s2 -> ../../devices/[...]/xkdisk@3,0:c
/dev/dsk/c0t3d0s3 -> ../../devices/[...]/xkdisk@3,0:d
/dev/dsk/c0t3d0s4 -> ../../devices/[...]/xkdisk@3,0:e
/dev/dsk/c0t3d0s5 -> ../../devices/[...]/xkdisk@3,0:f
/dev/dsk/c0t3d0s6 -> ../../devices/[...]/xkdisk@3,0:g
/dev/dsk/c0t3d0s7 -> ../../devices/[...]/xkdisk@3,0:h
```

and **/dev/rdsk** will contain the disk entries for the character device nodes in **/devices**

```
# ls -l /dev/rdsk
/dev/rdsk/c0t3d0s0 -> ../../devices/[...]/xkdisk@3,0:a,raw
/dev/rdsk/c0t3d0s1 -> ../../devices/[...]/xkdisk@3,0:b,raw
/dev/rdsk/c0t3d0s2 -> ../../devices/[...]/xkdisk@3,0:c,raw
/dev/rdsk/c0t3d0s3 -> ../../devices/[...]/xkdisk@3,0:d,raw
```

```
/dev/rdsk/c0t3d0s4 -> ../../devices/[...]/xkdisk@3,0:e,raw
/dev/rdsk/c0t3d0s5 -> ../../devices/[...]/xkdisk@3,0:f,raw
/dev/rdsk/c0t3d0s6 -> ../../devices/[...]/xkdisk@3,0:g,raw
/dev/rdsk/c0t3d0s7 -> ../../devices/[...]/xkdisk@3,0:h,raw
```

FILES

/dev/dsk/∗	disk entries (block device interface)
/dev/rdsk/∗	disk entries (character device interface)
/devices/∗	device special files (minor device nodes)

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

add_drv(1M), **devlinks**(1M), **drvconfig**(1M), **fdisk**(1M), **ports**(1M), **tapes**(1M), **attributes**(5), **dkio**(7I), **esp**(7D), **attach**(9E), **ddi_create_minor_node**(9F)

Writing Device Drivers

BUGS

disks silently ignores malformed minor device names.

NAME

diskscan – perform surface analysis

SYNOPSIS

diskscan [–**W**] [–**n**] [–**y**] *raw_device*

DESCRIPTION

diskscan is used by the system administrator to perform surface analysis on a portion of a hard disk. The disk portion may be a raw partition or slice; it is identified using its raw device name. By default, the specified portion of the disk is read (non-destructive) and errors reported on standard error. In addition, a progress report is printed on standard out. The list of bad blocks should be saved in a file and later fed into **addbadsec**(1M), which will remap them.

–**W** Causes **diskscan** to perform write and read surface analysis. This type of surface analysis is destructive and should be invoked with caution.

–**n** Causes **diskscan** to suppress linefeeds when printing progress information on standard out.

–**y** Causes **diskscan** to suppress the warning regarding destruction of existing data that is issued when –**W** is used.

FILES

The raw device should be **/dev/rdsk/c?t?d?[ps]?**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWcsu

SEE ALSO

fdisk(1M), **fmthard**(1M), **addbadsec**(1M), **attributes**(5)

NAME

dispadmin – process scheduler administration

SYNOPSIS

dispadmin –l
dispadmin –c *class* **–g [–r** *res***]**
dispadmin –c *class* **–s** *file*

DESCRIPTION

The **dispadmin** command displays or changes process scheduler parameters while the system is running.

dispadmin does limited checking on the values supplied in *file* to verify that they are within their required bounds. The checking, however, does not attempt to analyze the effect that the new values have on the performance of the system. Inappropriate values can have a negative effect on system performance. (See *System Administration Guide*.)

OPTIONS

–l Lists the scheduler classes currently configured in the system.

–c *class* Specifies the class whose parameters are to be displayed or changed. Valid *class* values are: **RT** for the real-time class, **TS** for the time-sharing class, and **IA** for the inter-active class. The time-sharing and inter-active classes share the same scheduler, so changes to the scheduling parameters of one will change those of the other.

–g Gets the parameters for the specified class and writes them to the standard output. Parameters for the real-time class are described in **rt_dptbl**(4). Parameters for the time-sharing and inter-active classes are described in **ts_dptbl**(4).

–r *res* When using the **–g** option you may also use the **–r** option to specify a resolution to be used for outputting the time quantum values. If no resolution is specified, time quantum values are in milliseconds. If *res* is specified it must be a positive integer between 1 and 1000000000 inclusive, and the resolution used is the reciprocal of *res* in seconds. For example, a *res* value of 10 yields time quantum values expressed in tenths of a second; a *res* value of 1000000 yields time quantum values expressed in microseconds. If the time quantum cannot be expressed as an integer in the specified resolution, it is rounded up to the next integral multiple of the specified resolution.

–s *file* Sets scheduler parameters for the specified class using the values in *file*. These values overwrite the current values in memory—they become the parameters that control scheduling of processes in the specified class. The values in *file* must be in the format output by the **–g** option. Moreover, the values must describe a table that is the same size (has same number of priority levels) as the table being overwritten.
Super-user privileges are required in order to use the **–s** option.

Note: The **–g** and **–s** options are mutually exclusive: you may not retrieve the table at the same time you are overwriting it.

EXAMPLES

The following command retrieves the current scheduler parameters for the real-time class from kernel memory and writes them to the standard output. Time quantum values are in microseconds.

```
dispadmin −c RT −g −r 1000000
```

The following command overwrites the current scheduler parameters for the real-time class with the values specified in **rt.config**.

```
dispadmin −c RT −s rt.config
```

The following command retrieves the current scheduler parameters for the time-sharing class from kernel memory and writes them to the standard output. Time quantum values are in nanoseconds.

```
dispadmin −c TS −g −r 1000000000
```

The following command overwrites the current scheduler parameters for the time-sharing class with the values specified in **ts.config**.

```
dispadmin −c TS −s ts.config
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

priocntl(1), **priocntl**(2), **rt_dptbl**(4), **ts_dptbl**(4), **attributes**(5)

System Administration Guide
System Interface Guide

DIAGNOSTICS

dispadmin prints an appropriate diagnostic message if it fails to overwrite the current scheduler parameters due to lack of required permissions or a problem with the specified input file.

NAME

dmesg – collect system diagnostic messages to form error log

SYNOPSIS

/usr/bin/dmesg [–]
/usr/sbin/dmesg [–]

DESCRIPTION

Note: **dmesg** is made obsolete by **syslogd**(1M) for maintenance of the system error log.

dmesg looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed or logged by the system when errors occur. If the '–' flag is given, then **dmesg** computes (incrementally) the new messages since the last time it was run and places these on the standard output.

FILES

/var/adm/msgbuf scratch file for memory of '–' option

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

SEE ALSO

syslogd(1M), **attributes**(5)

NAME

dmi_cmd − DMI command line interface utility

SYNOPSIS

dmi_cmd −**AL** −**c** *compId* −**g** *groupId* [−**dp**] [−**a** *attrId*] [−**m** *max-count*]
[−**r** *req-mode*] [−**s** *hostname*]

dmi_cmd −**CD** −**c** *compId* [−**s** *hostname*]

dmi_cmd −**CI** *mif-file* [−**s** *hostname*]

dmi_cmd −**CL** [−**dp**] [−**c** *compId*] [−**m** *max-count*] [−**r** *req-mode*] [−**s** *hostname*]

dmi_cmd −**GD** −**c** *compId* −**g** *groupId* [−**s** *hostname*]

dmi_cmd −**GI** *schema-file* −**c** *compId* [−**s** *hostname*]

dmi_cmd −**GL** −**c** *compId* −**g** *groupId* [−**dp**] [−**m** *max-count*] [−**r** *req-mode*]
[−**s** *hostname*]

dmi_cmd −**GM** −**c** *compId* [−**m** *max-count*] [−**s** *hostname*]

dmi_cmd −**h**

dmi_cmd −**ND** −**c** *compId* −**l** *language-string* [−**s** *hostname*]

dmi_cmd −**NI** *schema-file* −**c** *compId* [−**s** *hostname*]

dmi_cmd −**NL** −**c** *compId* [−**s** *hostname*]

dmi_cmd −**V** [−**s** *hostname*]

dmi_cmd −**W** *config-file* [−**s** *hostname*]

dmi_cmd −**X** [−**s** *hostname*]

DESCRIPTION

The **dmi_cmd** utility provides the ability to:

- Obtain version information about the DMI Service Provider
- Set the configuration to describe the language required by the management application
- Obtain configuration information describing the current language in use for the session
- Install components into the database
- List components in a system to determine what is installed
- Delete an existing component from the database
- Install group schemas to an existing component in the database
- List class names for all groups in a component
- List the groups within a component
- Delete a group from a component
- Install a language schema for an existing component in the database
- List the set of language mappings installed for a specified component
- Delete a specific language mapping for a component

- List the properties for one or more attributes in a group

OPTIONS

The following options are supported:

–a *attrId*	Specify an attribute by its ID (positive integer). The default value is **0**.
–AL	List the attributes for the specified component.
–c *compId*	Specify a component by its ID (positive integer). The default value is **0**.
–CD	Delete the specified component.
–CI *mif-file*	Install the component described in the *mif-file.*
–CL	List component information.
–d	Display descriptions.
–g *groupId*	Specify a group by its ID (positive integer). The default value is **0**.
–GD	Delete a group for the specified component.
–GI *schema-file*	Install the group schema specified in *schema-file.*
–GL	List the groups for the specified component.
–GM	List the class names for the specified component.
–h	Help. Print the command line usage.
–l *language-string*	Specify a language mapping.
–m *max-count*	Specify the maximum number of components to display.
–ND	Delete a language mapping for the specified component.
–NI *schema-file*	Install the language schema specified in *schema-file.*
–NL	List the language mappings for a specified component.
–p	Display the pragma string.
–r *req-mode*	Specify the request mode. The valid values are: **1** DMI_UNIQUE - access the specified item (or table row). **2** DMI_FIRST - access the first item. **3** DMI_NEXT - access the next item. The default request mode is **1** DMI_UNIQUE.
–s *hostname*	Specify the host machine on which **dmispd** is running. The default host is the local host.
–V	Version. Prints version information about the DMI Service Provider.
–W *config-file*	Set the configuration specified in *config-file* to **dmispd**.
–X	Retrieve configuration information describing the current language in use.

EXIT STATUS

The following error values are returned:

0 Successful completion.

–1 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi

SEE ALSO

dmiget(1M), **dmispd**(1M), **attributes**(5)

NAME

dmiget – DMI command line retrieval utility

SYNOPSIS

dmiget –c *compId* [**–a** *attrId*] [**–g** *groupId*] [**–s** *hostname*]

dmiget –h

DESCRIPTION

The **dmiget** utility retrieves the table information of a specific component in the DMI Service Provider.

OPTIONS

The following options are supported:

–**a** *attrId* Display the attribute information for the component specified with the –**c** argument.

–**c** *compId* Display all the table information for the specified component.

–**g** *groupId* Display all the attribute information in the group specified with *groupId* for the component specified with the –**c** argument

–**h** Help. Print the command line usage.

–**s** *hostname* Specify the host machine on which **dmispd** is running. The default host is the local host.

EXIT STATUS

The following error values are returned:

0 Successful completion.

–**1** An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi

SEE ALSO

dmi_cmd(1M), **dmispd**(1M), **attributes**(5)

NAME

dminfo – report information about a device entry in a device maps file

SYNOPSIS

dminfo [–**v**] [–**a**] [–**f** *pathname*]

dminfo [–**v**] [–**a**] [–**f** *pathname*] –**n** *dev–name* . . .

dminfo [–**v**] [–**a**] [–**f** *pathname*] –**d** *dev–path* . . .

dminfo [–**v**] [–**a**] [–**f** *pathname*] –**t** *dev–type* . . .

dminfo [–**v**] [–**f** *pathname*] –**u** *dm–entry*

DESCRIPTION

dminfo reports and updates information about the **device_maps**(4) file.

OPTIONS

–**v** Verbose. Print the requested entry or entries, one line per entry, on the standard output. If no entries are specified, all are printed.

–**a** Succeed if any of the requested entries are found. If used with –**v**, all entries that match the requested case(s) are printed.

–**f** *pathname* Use a device_maps file with *pathname* instead of **/etc/security/device_maps**.

–**n** *dev–name* Search by *dev–name*. Search **device_maps**(4) for a *device_name* field matching *dev–name*. This option cannot be used with –**d**, –**t** or –**u**.

–**d** *dev–path* Search by *dev–path*. Search **device_maps**(4) for a device special pathname in the *device_list* field matching the *dev–path* argument. This option cannot be used with –**n**, –**t** or –**u**.

–**t** *dev–type* Search by *dev–type*. Search **device_maps**(4) for a *device_type* field matching the given *dev–type*. This option cannot be used with –**d**, –**n** or –**u**.

–**u** *dm–entry* Update the **device_maps**(4) file. This option is provided to add entries to the **device_maps**(4) file. The *dm–entry* must be a complete **device_maps**(4) file entry. The *dm–entry* has fields, as in the **device_maps** file. It uses the colon (**:**) as a field separator, and white space as the *device_list* subfield separators. The *dm–entry* is not made if any fields are missing, or if the *dm–entry* would be a duplicate. The default device maps file can be updated only by the super user.

DIAGNOSTICS

dminfo returns an exit code of 0 if successful, 1 if the request failed, and 2 if the invocation syntax was incorrect.

FILES

/etc/security/device_maps

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

bsmconv(1M), **device_maps**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

dmispd – Sun Solstice Enterprise DMI Service Provider

SYNOPSIS

/usr/lib/dmi/dmispd [**–h**] [**–c** *config-dir*] [**–d** *debug-level*]

DESCRIPTION

The DMI Service Provider, **dmispd**, is the core of the DMI solution. Management applications and Component instrumentations communicate with each other through the Service Provider. The Service Provider coordinates and arbitrates requests from the management application to the specified component instrumentations. The Service Provider handles runtime management of the Component Interface (CI) and the Management Interface (MI), including component installation, registration at the MI and CI level, request serialization and synchronization, event handling for CI, and general flow control and housekeeping.

OPTIONS

The following options are supported:

–c *config-dir* Specify the full path of the directory containing the **dmispd.conf** configuration file. The default directory is **/etc/dmi/conf**.

–d *debug-level* Debug. Levels from 0 to 5 are supported, giving various levels of debug information. The default is 0, meaning no debug information is given.

If this option is omitted, then **dmispd** is run as a daemon process.

–h Help. Print the command line usage.

EXIT STATUS

The following error values are returned:

0 Successful completion.

1 An error occurred.

FILES

/etc/dmi/conf/dmispd.conf DMI Service Provider configuration file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi

SEE ALSO

snmpXdmid(1M), **attributes**(5)

NAME

domainname – set or display name of the current domain

SYNOPSIS

domainname [*name-of-domain*]

DESCRIPTION

Without an argument, **domainname** displays the name of the current domain, which typically encompasses a group of **hosts** or **passwd** entries under the same administration.

The **domainname** command is used by various components of Solaris to resolve names for types such as **passwd**, **hosts** and **aliases**. By default, various naming services such as NIS, NIS+, the Internet Domain Name Service (DNS) and **sendmail**(1M) use this **domainname** to resolve names. The **domainname** is normally a valid Internet domain name.

The **domainname** for various naming services can also be set by other means. For example, **ypinit** can be used to specify a different domain for all NIS calls. The file **/etc/resolv.conf** can be used to specify a different domain for DNS lookups. For **sendmail**, the **domainname** can be specified through the *sendmail_vars* entry in the **/etc/nsswitch.conf** file, or through the **/etc/mail/sendmail.cf** file.

Only the super-user can set the name of the domain by specifying the new **domainname** as an argument. The domain name of the machine is usually set during boot-time through the **domainname** command in the **/etc/init.d/inetinit** file. If the new domain name is not saved in the **/etc/defaultdomain** file, the machine will revert back to the old domain after rebooting.

FILES

/etc/defaultdomain
/etc/init.d/inetinit
/etc/mail/sendmail.cf
/etc/nsswitch.conf
/etc/resolv.conf

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

nis+(1), **nischown**(1), **nispasswd**(1), **hostconfig**(1M), **named**(1M), **nisaddcred**(1M), **sendmail**(1M), **ypinit**(1M), **sys-unconfig**(1M), **aliases**(4), **hosts**(4), **nsswitch.conf**(4), **passwd**(4), **attributes**(5)

NAME

drvconfig – configure the /devices directory

SYNOPSIS

drvconfig [–**bn**] [–**a** *alias_name*] [–**c** *class_name*] [–**i** *drivername*]
[–**m** *major_num*] [–**r** *rootdir*]

DESCRIPTION

The default operation of **drvconfig** is to create the **/devices** directory tree that describes, in the filesystem namespace, the hardware layout of a particular machine. Hardware devices present on the machine and powered on as well as pseudo-drivers are represented under **/devices**. Normally this command is run automatically after a new driver has been installed (with **add_drv**(1M)) and the system has been rebooted.

/etc/minor_perm file

drvconfig reads the **/etc/minor_perm** file to obtain permission information and applies the permissions only to nodes that it has just created. It does not change permissions on already existing nodes. The format of the **/etc/minor_perm** file is as follows:

*name***:***minor_name permissions owner group*

minor_name may be the actual name of the minor node, or contain shell metacharacters to represent several minor nodes (see **sh**(1)).

For example:

```
sd:* 0640 root sys
zs:[a-z],cu 0600 uucp uucp
mm:kmem 0640 root bin
```

The first line sets all devices exported by the **sd** node to **0640** permissions, owned by **root**, with group **sys**. In the second line, devices such as **a,cu and z,cu** exported by the **zs** driver are set to **0600** permission, owned by **uucp**, with group **uucp**. In the third line the **kmem** device exported by the **mm** driver is set to **0640** permission, owned by **root**, with group **bin**.

OPTIONS

The following options may be of use to system administrators and driver developers:

–**i** *drivername* Only configure the devices for the named driver.

The following options are used by the implementation of **add_drv**(1M) and **rem_drv**(1M), and may not be supported in future versions of Solaris:

–**b** Add a new major number to name binding into the kernel's internal **name_to_major** tables. This option is not normally used directly, but is used by other utilities such as **add_drv**(1M). Use of the –**b** option requires that –**i** and –**m** be used also. No **/devices** entries are created.

–**n** Do not try to load and attach any drivers, or if the –**i** option is given, do not try to attach the driver named *drivername*.

–**a** *alias_name* Add the name *alias_name* to the list of aliases that this driver is known by. This option, if used, must be used with the –**m** *major_num*, the –**b** and the –**i** *drivername* options.

–**c** *class_name* The driver being added to the system exports the class *class_name.* This option is not normally used directly, but is used by other utilities. It is only effective when used with the **–b** option.

–m *major_num* Specify the major number *major_num* for this driver to add to the kernel's **name_to_major** binding tables.

–r *rootdir* Build the device tree under the directory specified by *rootdir* instead of the default **/devices** directory.

EXIT STATUS

0 Successful completion.

non-zero An error occurred.

FILES

/devices	device nodes directory
/etc/minor_perm	minor mode permissions
/etc/name_to_major	major number binding
/etc/driver_classes	driver class binding file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

sh(1), **add_drv**(1M), **devlinks**(1M), **disks**(1M), **modinfo**(1M), **modload**(1M), **modunload**(1M), **ports**(1M), **rem_drv**(1M), **tapes**(1M), **path_to_inst**(4), **attributes**(5)

NOTES

This document does not constitute an API. **/etc/minor_perm**, **/etc/name_to_major**, **/etc/driver_classes**, and **/devices** may not exist or may have different contents or interpretations in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.

NAME

du – summarize disk usage

SYNOPSIS

/usr/bin/du [**–adkr**] [**–s** | **–o**] [*file* . . .]
/usr/xpg4/bin/du [**–a** | **–s**] [**–krx**] [*file* . . .]

DESCRIPTION

The **du** utility writes to standard output the size of the file space allocated to, and the size of the file space allocated to each subdirectory of, the file hierarchy rooted in each of the specified files. The size of the file space allocated to a file of type directory is defined as the sum total of space allocated to all files in the file hierarchy rooted in the directory plus the space allocated to the directory itself.

Files with multiple links will be counted and written for only one entry. The directory entry that is selected in the report is unspecified. By default, file sizes are written in 512-byte units, rounded up to the next 512-byte unit.

/usr/xpg4/bin/du

When **du** cannot obtain file attributes or read directories (see **stat**(2)), it will report an error condition and the final exit status will be affected.

OPTIONS

The following options are supported for **/usr/bin/du** and **/usr/xpg4/bin/du**:

–a In addition to the default output, report the size of each file not of type directory in the file hierarchy rooted in the specified file. Regardless of the presence of the **–a** option, non-directories give as *file* operands will always be listed.

–k Write the files sizes in units of 1024 bytes, rather than the default 512-byte units.

–s Instead of the default output, report only the total sum for each of the specified files.

/usr/bin/du

The following options are supported for **/usr/bin/du** only:

–d Do not cross filesystem boundaries. For example, **du –d /** reports usage only on the root partition.

–L Process symbolic links by using the file or directory which the symbolic link references, rather than the link itself.

–o Do not add child directories’ usage to a parent’s total. Without this option, the usage listed for a particular directory is the space taken by the files in that directory, as well as the files in all directories beneath it. This option does nothing if **–s** is used.

–r Generate messages about directories that cannot be read, files that cannot be opened, and so forth, rather than being silent (the default).

/usr/xpg4/bin/du

The following options are supported for **/usr/xpg4/bin/du** only:

–r By default, generate messages about directories that cannot be read, files that cannot be opened, and so forth.

–x When evaluating file sizes, evaluate only those files that have the same device as the file specified by the *file* operand.

OPERANDS

The following operand is supported:

file The path name of a file whose size is to be written. If no *file* is specified, the current directory is used.

OUTPUT

The output from **du** consists of the amount of the space allocated to a file and the name of the file.

USAGE

See **largefile**(5) for the description of the behavior of **du** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **du**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

/usr/bin/du

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

/usr/xpg4/bin/du

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4
CSI	enabled

SEE ALSO

ls(1), **stat**(2), **attributes**(5), **environ**(5), **largefile**(5), **xpg4**(5)

System Administration Guide

NOTES

A file with two or more links is counted only once. If, however, there are links between files in different directories where the directories are on separate branches of the file system hierarchy, **du** will count the excess files more than once.

Files containing holes will result in an incorrect block count.

NAME

edquota – edit user quotas for ufs file system

SYNOPSIS

edquota [**–p** *proto_user*] *username* . . .
edquota –t

DESCRIPTION

edquota is a quota editor. One or more users may be specified on the command line. For each user a temporary file is created with an ASCII representation of the current disk quotas for that user for each mounted ufs file system that has a **quotas** file, and an editor is then invoked on the file. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, **edquota** reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is **vi**(1) unless the **EDITOR** environment variable specifies otherwise.

Only the super-user may edit quotas. In order for quotas to be established on a file system, the root directory of the file system must contain a file, owned by root, called **quotas**. (See **quotaon**(1M).)

proto_user and *username* can be numeric, corresponding to the UID of a user. Unassigned UIDs may be specified; unassigned names may not. In this way, default quotas can be established for users who are later assigned a UID.

If no options are specified, the temporary file created will have one or more lines of the form

> fs *mount_point* blocks (soft =*number*, hard =*number*) inodes (soft =*number*, hard =*number*)
>
> Where a block is considered to be a 1024 byte (1K) block.

The *number* fields may be modified to reflect desired values.

OPTIONS

–p Duplicate the quotas of the *proto_user* specified for each *username* specified. This is the normal mechanism used to initialize quotas for groups of users.

–t Edit the soft time limits for each file system. If the time limits are zero, the default time limits in **/usr/include/sys/fs/ufs_quota.h** are used. The temporary file created will have one or more lines of the form

> fs *mount_point* blocks time limit = *number tmunit*, files time limit = *number tmunit*

tmunit may be one of ‘‘month’’, ‘‘week’’, ‘‘day’’, ‘‘hour’’, ‘‘min’’ or ‘‘sec’’; characters appended to these keywords are ignored, so you may write ‘‘months’’ or ‘‘minutes’’ if you prefer. The *number* and *tmunit* fields may be modified to set desired values. Time limits are printed in the greatest possible time unit such that the value is greater than or equal to one. If ‘‘default’’ is printed after the *tmunit*, this indicates that the value shown is zero (the default).

USAGE

See **largefile**(5) for the description of the behavior of **edquota** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

quotas	quota file at the file system root
/etc/mnttab	table of mounted file systems

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

vi(1), **quota**(1M), **quotacheck**(1M), **quotaon**(1M), **repquota**(1M), **attributes**(5), **largefile**(5), **quotactl**(7I)

NOTES

quotacheck(1M) *must* be invoked when setting initial quota limits for users; if not, the quota limit remains 0 and no changes made with **edquota** will take effect.

Users with a UID greater than 67108864 cannot be given quotas.

NAME

eeprom – EEPROM display and load utility

SYNOPSIS

/usr/platform/*platform-name***/sbin/eeprom** [–] [–**f** *device*] [*parameter* [=*value*] . . .]

DESCRIPTION

eeprom displays or changes the values of parameters in the EEPROM. On x86 machines, EEPROM storage is simulated using a file residing in the platform specific boot area. It processes parameters in the order given. When processing a *parameter* accompanied by a *value*, **eeprom** makes the indicated alteration to the EEPROM; otherwise it displays the *parameter*'s value. When given no parameter specifiers, **eeprom** displays the values of all EEPROM parameters. A '**–**' (hyphen) flag specifies that parameters and values are to be read from the standard input (one *parameter* or *parameter***=***value* per line).

Only the super-user may alter the EEPROM contents.

eeprom verifies the EEPROM checksums and complains if they are incorrect.

platform-name is the name of the platform implementation and can be found using the –**i** option of **uname**(1).

OPTIONS

–**f** *device* Use *device* as the EEPROM device.

x86 OPTIONS

–**I** Initialize boot properties on x86. Only **init**(1M) run-level initialization scripts should use this option.

NVRAM CONFIGURATION PARAMETERS

Not all OpenBoot systems support all parameters. Defaults may vary depending on the system and the PROM revision.

auto-boot? If **true**, boot automatically after power-on or reset. Defaults to **true**.

ansi-terminal? Configuration variable used to control the behavior of the terminal emulator. The value **false** makes the terminal emulator stop interpreting ANSI escape sequences, instead just echoing them to the output device. Default value: **true**.

boot-command Command executed if **auto-boot?** is **true**. Default value is **boot**.

boot-device Device from which to boot. *boot-device* may contain 0 or more device specifiers separated by spaces. Each device specifier may be either a prom device alias or a prom device path. The boot prom will attempt to open each successive device specifier in the list beginning with the first device specifier. The first device specifier which opens successfully will be used as the device to boot from. Defaults to **disk net**.

boot-file File to boot (an empty string lets the secondary booter choose default). Defaults to empty string.

boot-from Boot device and file (OpenBoot PROM version 1.*x* only). Defaults to **vmunix**.

boot-from-diag Diagnostic boot device and file (OpenBoot PROM version 1.*x* only). Defaults to **le()unix**.

diag-device Diagnostic boot source device. Defaults to **net**.

diag-file File from which to boot in diagnostic mode. Defaults to empty string.

diag-level Diagnostics level. Values include **off**, **min**, **max** and **menus**. There may be additional platform-specific values. When set to **off**, POST is not called. If POST is called, the value is made available as an argument to, and is interpreted by POST. The default value is **platform-dependent**.

diag-switch? If **true**, run in diagnostic mode. Defaults to **true**.

fcode-debug? If **true**, include name parameter for plug-in device FCodes. Defaults to **false**.

hardware-revision System version information.

input-device Input device used at power-on (usually **keyboard**, **ttya**, or **ttyb**). Defaults to **keyboard**.

keyboard-click? If **true** enable keyboard click. Defaults to **false**.

keymap Keymap for custom keyboard.

last-hardware-update System update information.

load-base Default load address for client programs. Default value is **16384**.

local-mac-address? If true, network drivers use their own MAC address, not system’s. Defaults to **false**.

mfg-mode Manufacturing mode argument for POST. Possible values include **off** or **chamber**. The value is passed as an argument to POST. Default value: **off**.

mfg-switch? If true, repeat system self-tests until interrupted with STOP-A. Defaults to **false**.

nvramrc Contents of NVRAMRC. Defaults to empty.

oem-banner Custom OEM banner (enabled by setting **oem-banner?** to **true**). Defaults to empty string.

oem-banner? If **true**, use custom OEM banner. Defaults to **false**.

oem-logo Byte array custom OEM logo (enabled by setting **oem-logo?** to **true**). Displayed in hexadecimal.

oem-logo? If **true**, use custom OEM logo (else, use Sun logo). Defaults to **false**.

output-device Output device used at power-on (usually **screen**, **ttya**, or **ttyb**). Defaults to **screen**.

sbus-probe-list Which SBus slots are probed and in what order. Defaults to

0123.

screen-#columns Number of on-screen columns (characters / line). Defaults to **80**.

screen-#rows Number of on-screen rows (lines). Defaults to **34**.

scsi-initiator-id SCSI bus address of host adapter, range 0-7. Defaults to **7**.

sd-targets Map SCSI disk units (OpenBoot PROM version 1.*x* only). Defaults to **31204567**, which means that unit 0 maps to target **3**, unit 1 maps to target **1**, and so on.

security-#badlogins Number of incorrect security password attempts.

security-mode Firmware security level (options: **none**, **command**, or **full**). If set to **command** or **full**, system will prompt for PROM security password. Defaults to **none**.

security-password Firmware security password (never displayed). Can be set only when **security-mode** is set to **command** or **full**.

```
example# eeprom security-password=
Changing PROM password:
New password:
Retype new password:
```

selftest-#megs Metabytes of RAM to test. Ignored if **diag-switch?** is **true**. Defaults to **1**.

skip-vme-loopback? If **true**, POST does not do VMEbus loopback tests. Defaults to **false**.

st-targets Map SCSI tape units (OpenBoot PROM version 1.*x* only). Defaults to **45670123**, which means that unit 0 maps to target **4**, unit 1 maps to target **5**, and so on.

sunmon-compat? If **true**, display Restricted Monitor prompt (>). Defaults to **false**.

testarea One-byte scratch field, available for read / write test. Defaults to **0**.

tpe-link-test? Enable 10baseT link test for built-in twisted pair Ethernet. Defaults to **true**.

ttya-mode TTYA (baud rate, #bits, parity, #stop, handshake). Defaults to **9600,8,n,1,–**.

Fields, in left-to-right order, are:

baud rate:	110, 300, 1200, 4800, 9600 . . .
data bits:	5, 6, 7, 8
parity:	n(none), e(even), o(odd), m(mark), s(space)
stop bits:	1, 1.5, 2

handshake:	–(none), h(hardware:rts / cts), s(software:xon / xoff)

ttyb-mode TTYB (baud rate, #bits, parity, #stop, handshake). Defaults to **9600,8,n,1,–**.

Fields, in left-to-right order, are:

baud rate:	110, 300, 1200, 4800, 9600 . . .
data bits:	5, 6, 7, 8
stop bits:	1, 1.5, 2
parity:	n(none), e(even), o(odd), m(mark), s(space)
handshake:	–(none), h(hardware:rts / cts), s(software:xon / xoff)

ttya-ignore-cd If **true**, operating system ignores carrier-detect on TTYA. Defaults to **true**.

ttyb-ignore-cd If **true**, operating system ignores carrier-detect on TTYA. Defaults to **true**.

ttya-rts-dtr-off If **true**, operating system does not assert DTR and RTS on TTYA. Defaults to **false**.

ttyb-rts-dtr-off If **true**, operating system does not assert DTR and RTS on TTYB. Defaults to **false**.

use-nvramrc? If **true**, execute commands in NVRAMRC during system start-up. Defaults to **false**.

version2? If **true**, hybrid (1.*x* / 2.*x*) PROM comes up in version 2.*x*. Defaults to **true**.

watchdog-reboot? If **true**, reboot after watchdog reset. Defaults to **false**.

EXAMPLES

The following example demonstrates the method for changing from one to two the number of megabytes of RAM that the system will test.

```
example# eeprom selftest-#megs
selftest-#megs=1

example# eeprom selftest-#megs=2

example# eeprom selftest-#megs
selftest-#megs=2
```

The following example demonstrates the method for setting the **auto-boot?** parameter to **true**.

```
example# eeprom auto-boot?=true
```

When the **eeprom** command is executed in user mode, the parameters with a trailing question mark (?) need to be enclosed in double quotation marks (" ") to prevent the shell from interpreting the question mark. Preceding the question mark with an escape character (\) will also prevent the shell from interpreting the question mark.

```
example% eeprom "auto-boot?"=true
```

FILES

/dev/openprom device file

/usr/platform/*platform-name***/sbin/eeprom**
platform-specific version of **eeprom**. To obtain *platform-name*, use **uname –i**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

passwd(1), **init**(1M), **sh**(1), **uname**(1), **attributes**(5)

OpenBoot 3.x Command Reference Manual ONC+ Developer's Guide

NAME

fdetach – detach a name from a STREAMS-based file descriptor

SYNOPSIS

fdetach *path*

DESCRIPTION

The **fdetach** command detaches a STREAMS-based file descriptor from a name in the file system. *path* is the path name of the object in the file system name space, which was previously attached (see **fattach**(3C)). The user must be the owner of the file or a user with the appropriate privileges. All subsequent operations on *path* will operate on the underlying file system entry and not on the STREAMS file. The permissions and status of the entry are restored to the state they were in before the STREAMS file was attached to the entry.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fattach(3C), **fdetach**(3C), **attributes**(5), **streamio**(7I)

STREAMS Programming Guide

NAME

fdisk – create or modify fixed disk partition table

SYNOPSIS

fdisk [–o *offset* **] [–s** *size* **] [–P** *fill_patt* **] [–S** *geom_file* **]**
[–w | r | d | n | I | B | t | T | g | G | R] [–F *fdisk_file* **] [[–v] –W {***fdisk_file* **| –}]**
[–h] [–b *masterboot* **]**
[–A *id* **:** *act* **:** *bhead* **:** *bsect* **:** *bcyl* **:** *ehead* **:** *esect* **:** *ecyl* **:** *rsect* **:** *numsect* **]**
[–D *id* **:** *act* **:** *bhead* **:** *bsect* **:** *bcyl* **:** *ehead* **:** *esect* **:** *ecyl* **:** *rsect* **:** *numsect* **]** *rdevice*

DESCRIPTION

This command is used to create and modify the partition table, and to install the master boot (x86 only) record that is put in the first sector of the fixed disk. This table is used by the first-stage bootstrap (or firmware) to identify parts of the disk reserved for different operating systems, and to identify the partition containing the second-stage bootstrap (the *active* Solaris partition). The *rdevice* argument must be used to specify the raw device associated with the fixed disk, for example, **/dev/rdsk/c0t0d0p0**.

The program can operate in three different modes. The first is interactive mode. In interactive mode, the program displays the partition table as it exists on the disk, and then presents a menu allowing the user to modify the table. The menu, questions, warnings, and error messages are intended to be self-explanatory.

In interactive mode, if there is no partition table on the disk, the user is given the options of creating a default partitioning or specifying the initial table values. The default partitioning allocates the entire disk for the Solaris system and makes the Solaris system partition active. In either case, when the initial table is created, **fdisk** also writes out the first-stage bootstrap (x86 only) code along with the partition table.

The second mode of operation is used for automated entry addition, entry deletion, or replacement of the entire fdisk table. This mode can add or delete an entry described on the command line. In this mode the entire **fdisk** table can be read in from a file replacing the original table. **fdisk** can also be used to create this file. There is a command line option that will cause **fdisk** to replace any **fdisk** table with the default of the whole disk for the Solaris system.

The third mode of operation is used for disk diagnostics. In this mode, a section of the disk can be filled with a user specified pattern, and mode sections of the disk can also be read or written.

Menu Options

The menu options for interactive mode given by the **fdisk** program are:

Create a partition

This option allows the user to create a new partition. The maximum number of partitions is 4. The program will ask for the type of the partition (SOLARIS, MS-DOS, UNIX, or other). It will then ask for the size of the partition as a percentage of the disk.
The user may also enter the letter **c** at this point, in which case the program will ask for the starting cylinder number and size of the partition in cylinders. If a **c** is not entered, the program will determine the starting cylinder number where the partition will fit. In either case, if the partition would overlap an

existing partition or will not fit, a message is displayed and the program returns to the original menu.

Change Active (Boot from) partition
This option allows the user to specify the partition where the first-stage bootstrap will look for the second-stage bootstrap, otherwise known as the *active* partition.

Delete a partition
This option allows the user to delete a previously created partition. Note that this will destroy all data in that partition.

Use the following options to include your modifications to the partition table at this time or to cancel the session without modifying the table:

Exit This option writes the new version of the table created during this session with *fdisk* out to the fixed disk, and exits the program.

Cancel This option exits without modifying the partition table.

OPTIONS

The following options apply to **fdisk**:

–S *geom_file* Set the label geometry to the content of the *geom_file*. The *geom_file* contains one specification line. Each line is delimited by a new-line character (**\n**). If the first character of a line is an asterisk (*), the line is treated as a comment. Each line is composed of entries that are position-dependent, are separated by white space, and have the following format:

PCYL NCYL ACYL BCYL NHEADS NSECTORS SECTSIZ

where the entries have the following values:

PCYL This is the number of physical cylinders for the drive.

NCYL This is the number of usable cylinders for the drive.

ACYL This is the number of alt cylinders for the drive.

BCYL This is the number of offset cylinders for the drive (should be zero).

NHEADS The number of heads for this drive.

NSECTORS The number of sectors per track.

SECTSIZ The size in bytes of a sector.

–g Get the label geometry for disk and display on stdout (see the **–S** option for the format).

–G Get the physical geometry for disk and display on stdout (see the **–S** option for the format).

–n Don't update **fdisk** table unless explicitly specified by another option. If no other options are used, **–n** will only write the master boot record to the disk. In addition, note that **fdisk** will not come up in interactive mode if the **–n** option is specified.

–I Forgo device checks. This is used to generate a file image of what would go on a disk without using the device. Note that you must use –**S** with this option (see above).

–B Default to one Solaris partition that uses the whole disk.

–t Adjust incorrect slice table entries so that they will not cross partition table boundaries.

–T Remove incorrect slice table entries that span partition table boundaries.

–b *master_boot* (x86 only) Specify the file *master_boot* as the master boot program. The default master boot program is **/usr/platform/***platform-name***/lib/fs/ufs/mboot**. The platform name can be found using the –**i** option of **uname**(1).

–R Treat disk as read-only. This is for testing purposes.

–W *fdisk_file* Create an **fdisk** file *fdisk_file* from disk table. This can be used with the –**F** option below.

–W – Output the disk table to **stdout**.

–v Output the HBA (virtual) geometry demensions. This option must be used in conjunction with the –**W** flag. This option will work for platforms which support virtual geometry. (x86 only)

–A *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
Add a partition as described by the argument (see the –**F** option below for the format).

–D *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
Delete a partition as described by the argument (see the –**F** option below for the format). Note that the argument must be an exact match or the entry will not be deleted!

–F *fdisk_file* Use fdisk file *fdisk_file* to initialize table. The *fdisk_file* contains up to four specification lines. Each line is delimited by a new-line character (**\n**). If the first character of a line is an asterisk (∗), the line is treated as a comment. Each line is composed of entries that are position-dependent, are separated by ‘‘white space’’ or colons, and have the following format:

id act bhead bsect bcyl ehead esect ecyl rsect numsect

where the entries have the following values:

id This is the type of partition and the correct numeric values may be found in **fdisk.h**.

act This is the active partition flag; **0** means not active and **128** means active.

bhead This is the head where the partition starts. If this is set to **0**, **fdisk** will correctly fill this in from other information.

bsect This is the sector where the partition starts. If this is set to **0**, **fdisk** will correctly fill this in from other information.

bcyl This is the cylinder where the partition starts. If this is set to **0**, **fdisk** will correctly fill this in from other information.

ehead This is the head where the partition ends. If this is set to **0**, **fdisk** will correctly fill this in from other information.

esect This is the sector where the partition ends. If this is set to **0**, **fdisk** will correctly fill this in from other information.

ecyl This is the cylinder where the partition ends. If this is set to **0**, **fdisk** will correctly fill this in from other information.

rsect The relative sector from the beginning of the disk where the partition starts. This must be specified and can be used by **fdisk** to fill in other fields.

numsect The size in sectors of this disk partition. This must be specified and can be used by **fdisk** to fill in other fields.

–**r** Read from disk and write to stdout. See –**o** and –**s**, which specify the starting point and size of the operation.

–**w** Write to disk and read from stdin. See –**o** and –**s**, which specify the starting point and size of the operation.

–**P** *fill_patt* Fill disk with pattern *fill_patt*. *fill_patt* can be decimal or hex and is used as number for constant long word pattern. If *fill_patt* is **#**, then pattern is block # for each block. Pattern is put in each block as long words and fills each block (see –**o** and –**s**).

–**o** *Offset* Block offset from start of disk. This option is used for –**P**, –**r**, and –**w**. Zero is assumed when this option is not used.

–**s** *size* Number of blocks to perform operation on (see –**o**).

–**d** Turn on verbose *debug* mode. This will cause **fdisk** to print its state on stderr as it is used. The output from this option should not be used with –**F**.

–**h** Issue verbose message; message will list all options and supply an explanation for each.

FILES

/dev/rdsk/c0t0d0p0 Raw device associated with the fixed disk.

x86 Only

/usr/platform/*platform-name***/lib/fs/ufs/mboot** Default master boot program.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86, PowerPC Edition
Availability	SUNWcsu

SEE ALSO

uname(1), **fmthard**(1M), **prtvtoc**(1M)**, attributes**(5)

DIAGNOSTICS

Most messages will be self-explanatory. The following may appear immediately after starting the program:

Fdisk: **cannot open** <**device**>

This indicates that the device name argument is not valid.

Fdisk: **unable to get device parameters for device** <**device**>

This indicates a problem with the configuration of the fixed disk, or an error in the fixed disk driver.

Fdisk: **error reading partition table**

This indicates that some error occurred when trying initially to read the fixed disk. This could be a problem with the fixed disk controller or driver, or with the configuration of the fixed disk.

Fdisk: **error writing boot record**

This indicates that some error occurred when trying to write the new partition table out to the fixed disk. This could be a problem with the fixed disk controller, the disk itself, the driver, or the configuration of the fixed disk.

NAME

ff – list file names and statistics for a file system

SYNOPSIS

ff [–**F** *FSType*] [–**V**] [*generic_options*] [–**o** *specific_options*] *special* . . .

DESCRIPTION

ff prints the pathnames and inode numbers of files in the file system which resides on the special device *special*. Other information about the files may be printed using options described below. Selection criteria may be used to instruct **ff** to only print information for certain files. If no selection criteria are specified, information for all files considered will be printed (the default); the –**i** option may be used to limit files to those whose inodes are specified.

Output is sorted in ascending inode number order. The default line produced by **ff** is:

> *path-name i-number*

The maximum information the command will provide is:

> *path-name i-number size uid*

OPTIONS

–**F** Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from **/etc/vfstab** by matching the *special* with an entry in the table, or by consulting **/etc/default/fs**.

–**V** Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from **/etc/vfstab**. This option may be used to verify and validate the command line.

generic_options Options that are supported by most FSType-specific modules of the command. The following options are available:

- –**I** Do not print the i-node number after each path name.
- –**l** Generate a supplementary list of all path names for multiply-linked files.
- –**p** *prefix* The specified *prefix* will be added to each generated path name. The default is ‘.’ (dot).
- –**s** Print the file size, in bytes, after each path name.
- –**u** Print the owner’s login name after each path name.
- –**a** –*n* Select if the file has been accessed in *n* days.
- –**m** –*n* Select if the file has been written or created in *n* days.
- –**c** –*n* Select if file’s status has been changed in *n* days.
- –**n** *file* Select if the file has been modified more recently than the argument *file*.

−**i** *i-node-list* Generate names for only those i-nodes specified in *i-node-list*. *i-node-list* is a list of numbers separated by commas (with no intervening spaces).

−**o** Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

OPERANDS

special A special device.

USAGE

See **largefile**(5) for the description of the behavior of **ff** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/default/fs default local file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: **LOCAL=ufs**

LOCAL The default partition for a command if no FSType is specified.

/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

find(1), **ncheck**(1M), **stat**(2), **vfstab**(4), **attributes**(5), **largefile**(5)
Manual pages for the *FSType*-specific modules of **ff**.

NOTES

This command may not be supported for all *FSTypes*.

The −**a**, −**m**, and −**c** flags examine the **st_atime**, **st_mtime**, and **st_ctime** fields of the **stat** structure respectively. (See **stat**(2).)

NAME

ffbconfig – configure the FFB Graphics Accelerator

SYNOPSIS

/usr/sbin/ffbconfig [**–dev** *device-filename*]
[**–res** *video-mode* [**now** | **try**] [**noconfirm** | **nocheck**]]
[**–file machine** | **system**]
[**–deflinear** **true** | **false**]
[**–defoverlay** **true** | **false**]
[**–linearorder** **first** | **last**]
[**–overlayorder** **first** | **last**]
[**–maxwids** *n*] [**–propt**] [**–prconf**] [**–defaults**]

/usr/sbin/ffbconfig [**–propt**] [**–prconf**]

/usr/sbin/ffbconfig [**–help**] [**–res** ?]

DESCRIPTION

ffbconfig configures the FFB Graphics Accelerator and some of the X11 window system defaults for FFB.

The first form of **ffbconfig** shown in the synopsis above stores the specified options in the OWconfig file. These options will be used to initialize the FFB device the next time the window system is run on that device. Updating options in the OWconfig file provides persistence of these options across window system sessions and system reboots.

The second and third forms which invoke only the –**prconf**, –**propt**, –**help**, and –**res** ? options do not update the OWconfig file. Additionally, for the third form all other options are ignored.

Options may be specified for only one FFB device at a time. Specifying options for multiple FFB devices requires multiple invocations of **ffbconfig**.

Only FFB-specific options can be specified through **ffbconfig**. The normal window system options for specifying default depth, default visual class and so forth are still specified as device modifiers on the openwin command line (see the Xsun(1) manual page in the Openwindows Reference Manual).

The user can also specify the OWconfig file that is to be updated. By default, the machine-specific file in the /etc/openwin directory tree is updated. The –**file** option can be used to specify an alternate file to use. For example, the system-global OWconfig file in the /usr/openwin directory tree can be updated instead.

Both of these standard OWconfig files can only be written by root. Consequently, the **ffbconfig** program, which is owned by the root user, always runs with setuid root permission.

OPTIONS

–**dev** *device-filename*
Specifies the FFB special file. The default is **/dev/fbs/ffb0**.

–**file** **machine** | **system**
Specifies which OWconfig file to update. If **machine**, the machine-specific OWconfig file in the /etc/openwin directory tree is used. If **system**, the global OWconfig file in the /usr/openwin directory tree is used. If the file does not

exist, it is created.

–res *video-mode* [**now** | **try** [**noconfirm** | **nocheck**]]

Specifies the video mode used to drive the monitor connected to the specified FFB device.

The format of these built-in video modes is:

widthx**height**x**rate**

where **width** is the screen width in pixels, **height** is the screen height in pixels, and **rate** is the vertical frequency of the screen refresh. The **s** suffix of 960x680x112s and 960x680x108s means that these are stereo video modes. The **i** suffix of 640x480x60i and 768x575x50i designates interlaced video timing. If absent, non-interlaced timing will be used. As a convenience, –**res** also accepts formats with '@' (at sign) in front of the refresh rate instead of x. For example: 1280x1024@76. The list of valid video-modes is show below. This list can also be obtained by running **ffbconfig** with the –**res** ? option (the third form shown in the command synopsis above).

1024x768x60	
1024x768x70	
1024x768x77	
1024x800x84	
1152x900x66	
1152x900x76	
1280x1024x67	
1280x1024x76	
960x680x112s	(Stereo)
960x680x108s	(Stereo)
640x480x60i	(Interlaced)
768x575x50i	(Interlaced)

Symbolic names

For convenience, some of the above video modes have symbolic names defined for them. Instead of the form **width** x **height** x **rate**, one of these names may be supplied as the argument to –**res**. The meaning of the symbolic name **none** is that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

Name	Corresponding Video Mode
svga	1024x768x60
1152	1152x900x76
1280	1280x1024x76
stereo	960x680x112s
ntsc	640x480x60i
pal	768x575x50i
none	(see text above)

The –**res** option also accepts additional, optional arguments immediately following the video mode specification. Any or all of these may be present.

now If present, not only will the video mode be updated in the OWconfig file, but the FFB device will be immediately programmed to display this video mode. (This is useful for changing the video mode before starting the window system).

Note – It is inadvisable to use this suboption with **ffbconfig** while the configured device is being used (e.g. while running the window system); unpredictable results may occur. To run **ffbconfig** with the now suboption, first bring the window system down. If the now suboption is used within a window system session, the video mode will be changed immediately, but the width and height of the affected screen won't change until the window system is exited and reentered again. In addition, the system may not recognize changes in stereo mode. Consequently, this usage is strongly discouraged.

noconfirm Using the –**res** option, the user could potentially put the system into an usable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of **ffbconfig** is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The noconfirm option instructs **ffbconfig** to bypass this confirmation and to program the requested video mode anyway. This option is useful when **ffbconfig** is being run from a shell script.

nocheck If present, the normal error checking based on the monitor sense code (described above) will be suspended. The video mode specified by the user will be accepted regardless of whether it is appropriate for the currently attached monitor. (This option is useful if a different monitor is to be connected to the FFB device). *Use of this option implies noconfirm well*.

try If present, the specified video mode will be programmed on a trial basis. The user will be asked to confirm the video mode by typing 'y' within 10 seconds. Or the user may terminate the trial before 10 seconds are up by typing any character. Any character other than 'y' or carriage return is considered a no and the previous video mode will be restored and **ffbconfig** will not change the video mode in the OWconfig file (other options specified will still take effect). If a carriage return is typed, the user is prompted for a yes or no answer on whether to keep the new video mode. This option implies the now suboption (see the warning note on the now suboption).

FFB possesses two types of visuals: linear and nonlinear. Linear visuals are
gamma corrected and nonlinear visuals are not. There are two visuals that have

both linear and nonlinear versions: 24-bit TrueColor and 8-bit StaticGray.

If true, the default visual is set to the linear visual that satisfies other specified default visual selection options (specifically, the Xsun(1) defdepth and defclass options described in the OpenWindows Reference Manual).

If false, or if there is no linear visual that satisfies the other default visual selection options, the non-linear visual specified by these other options will be chosen to be the default.

This option cannot be used when the –**defoverlay** option is present, because FFB doesn't possess a linear overlay visual.

–**defoverlay true** | **false**

The FFB provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the FFB visuals. This is called the overlay visual. Windows created in this visual will not damage windows created in other visuals. The converse, however, is not true. Windows created in other visuals will damage overlay windows. This visual has (256 – maxwids) number of opaque color values (refer to the –**maxwids** option).

If the value of this option is true, the overlay visual will be made the default visual.

If false, the nonoverlay visual that satisfies the other default visual selection options, such as defdepth and defclass, will be chosen as the default visual. See the Xsun(1) manual page in the OpenWindows Reference Manual.

Whenever –**defoverlay true** is used, the default depth and class chosen on the openwin command line must be 8-bit PseudoColor. If not, a warning message will be printed and the –**defoverlay** option will be treated as false.

This option cannot be used when the –**deflinear** option is present, because FFB doesn't possess a linear overlay visual.

–**linearorder first** | **last**

If true, linear visuals will come before their non-linear counterparts on the X11 screen visual list for the FFB screen. If false, the nonlinear visuals will come before the linear ones.

–**overlayorder** first | last

If true, the depth 8 PseudoColor Overlay visual will come before the non-overlay visual on the X11 screen visual list for the FFB screen. If false, the non-overlay visual will come before the overlay one.

–**maxwids** *n*

Specifies the maximum number of FFB X channel pixel values that are reserved for use as window IDs (WIDs). The remainder of the pixel values in overlay colormaps are used for normal X11 opaque color pixels.

The reserved WIDs are allocated on a first-come first-serve basis by 3D graphics windows (such as XGL), MBX windows, and windows that have a non-default visual.

The X channel codes 0 to (255 – n) will be opaque color pixels. The X channel codes (255 – n + 1) to 255 will be reserved for use as WIDs. Legal values: 1, 2, 4, 8, 16, 32.

–defaults
Resets all option values to their default values.

–propt Prints the current values of all FFB options in the OWconfig file specified by the –**file** option for the device specified by the –**dev** option. Prints the values of options as they will be in the OWconfig file after the call to **ffbconfig** completes. This is a typical display:

```
--- OpenWindows Configuration for /dev/fbs/ffb0 ---
OWconfig: machine
Video Mode: NONE
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
Allocated WIDs: 32
```

–prconf
Prints the FFB hardware configuration. This is a typical display:

```
--- Hardware Configuration for /dev/fbs/ffb0 ---
Type: double-buffered FFB with Z-buffer
Board: rev x
FBC: version x
DAC: Brooktree 9068, version x
3DRAM: Mitsubishi 1308, version x
Monitor Sense ID: 4  (Sun 19" monitor)
Monitor possible resolutions: 1280x1024x67, 1280x1024x76
        1152x900x66, 1152x900x76, 960x680x112s, 960x680x108s
        1024x800x84, 1024x768x77
Current resolution setting: 1280x1024x76
```

–help Prints a list of the **ffbconfig** command line options, along with a brief explanation of each.

DEFAULTS

For a given invocation of **ffbconfig** command line if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value.

When the window system is run, if an FFB option has never been specified via **ffbconfig**, a default value is used. The option defaults are as follows:

Option	Default
–dev	/dev/fbs/ffb0
–file	machine
–res	none
–deflinear	false
–defoverlay	false
–linearorder	last
–overlayorder	last
–maxwids	32

The default for the **–res** option of none means that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

Note – This provides compatibility for users who are used to specifying the device resolution through the PROM. On some devices (e.g. GX) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default FFB video mode.

EXAMPLES

The following example switches the monitor type to the maximum resolution of 1280 × 1024 at 76 Hz:

```
example% /usr/sbin/ffbconfig –res 1280x1024x76
```

FILES

/dev/fbs/ffb0	device special file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWffbcf

SEE ALSO

mmap(2), **attributes**(5), **fbio**(7I), **ffb**(7D)

NAME

ff_ufs – list file names and statistics for a ufs file system

SYNOPSIS

ff –F ufs [*generic_options*] [**–o a,m,s**] *special*. . .

DESCRIPTION

ff prints the pathnames and inode numbers of files in the file system which resides on the special device *special*. **ff** is described in **ff**(1M); ufs-specific options are described below.

OPTIONS

–o Specify **ufs** file system specific options. The options available are:

- **a** Print the '.' and '. .' directory entries.
- **m** Print mode information. This option must be specified in conjunction with the **–i** *i-node-list* option (see **ff**(1M)).
- **s** Print only special files and files with set-user-ID mode.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWcsu

SEE ALSO

find(1), **ff**(1M), **ncheck**(1M), **attributes**(5)

NAME

firmware – bootable firmware programs and firmware commands

DESCRIPTION

Between the time most computers are turned on and the boot program is loaded to bootstrap the machine, the computer is in an operating state known as the firmware state. In the firmware state, a small program in non-volatile memory is running on the machine, and the user can perform certain system operations usually unavailable from single- or multi-user operating states.

There are two basic kinds of firmware operations:

> Running firmware commands. These commands might include commands for displaying the Equipped Device Table, performing a system memory dump, displaying the firmware version, creating a floppy key, etc. These commands are executed by the firmware program.
>
> Running bootable programs. These include the operating system and other bootable programs (such as a program to fill the Equipped Device Table). These programs are located in the **/stand** file system. When a bootable program is requested from firmware, the firmware program loads and executes the program, passing control of the system to the bootable program.

Some firmware programs, allow you to request the configuration of a new bootable operating system from firmware by specifying the name of a configuration file (usually **/stand/system**) as the name of the program to boot; see **system**(4).

See the hardware guide that accompanies your computer for descriptions of the firmware commands and programs available with your machine.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC

SEE ALSO

system(4), **attributes**(5)

Writing FCode 2.x Programs

OpenBoot 2.x Command Reference Manual

WARNINGS

The firmware program typically does not know if a requested program is bootable or not; requesting a program that is not bootable from firmware can lead to unpredictable results.

NAME

fmthard – populate VTOC on hard disks

SPARC SYNOPSIS

fmthard –d *data* | **–n** *volume_name* | **–s** *datafile* [**–i**] **/dev/rdsk/c?**[**t?**]**d?s2**

x86 SYNOPSIS

fmthard –d *data* | **–n** *volume_name* | **–s** *datafile* [**–i**][**–p** *pboot*][**–b** *bootblk*]
/dev/rdsk/c?[**t?**]**d?s2**

DESCRIPTION

The **fmthard** command updates the VTOC (Volume Table of Contents) on hard disks and, on x86 systems, adds boot information to the Solaris **fdisk** partition. One or more of the options **–s** *datafile*, **–d** *data*, or **–n** *volume_name*, must be used to request modifications to the disk label. To print disk label contents, see **prtvtoc**(1M). The **/dev/rdsk/c***?*[**t***?*]**d***?***s2** file must be the character special file of the device where the new VTOC is to be installed. On x86 systems, **fdisk**(1M) must be run on the drive before **fmthard**.

If you are using an x86 system, note that the term ‘‘partition’’ in this page refers to *slices* within the x86 **fdisk** partition on x86 machines. Do not confuse the partitions created by **fmthard** with the partitions created by **fdisk**.

OPTIONS

The following options apply to **fmthard**:

–i This option allows the command to create the desired VTOC table, but prints the information to standard output instead of modifying the VTOC on the disk.

–d *data* The *data* argument of this option is a string representing the information for a particular partition in the current VTOC. The string must be of the format *part:tag:flag:start:size* where *part* is the partition number, *tag* is the ID TAG of the partition, *flag* is the set of permission flags, *start* is the starting sector number of the partition, and *size* is the number of sectors in the partition. See the description of the *datafile* below for more information on these fields.

–n *volume_name* This option is used to give the disk a *volume_name* up to 8 characters long.

–s *datafile* This option is used to populate the VTOC according to a *datafile* created by the user. If the *datafile* is "–", **fmthard** reads from standard input. The *datafile* format is described below. This option causes all of the disk partition timestamp fields to be set to zero.

Every VTOC generated by **fmthard** will also have partition 2, by convention, that corresponds to the whole disk. If the input in *datafile* does not specify an entry for partition 2, a default partition 2 entry will be created automatically in VTOC with the tag **V_BACKUP** and size equal to the full size of the disk.

The *datafile* contains one specification line for each partition, starting with partition 0. Each line is delimited by a new-line character (**\n**). If the first character of a line is an asterisk (∗), the line is treated as a comment.

Each line is composed of entries that are position-dependent, separated by "white space" and having the following format:

partition tag flag starting_sector size_in_sectors

where the entries have the following values.

partition The partition number: **0-15** decimal or **0x0-0xf** hexadecimal.

tag The partition tag: a two-digit hex number. The following are reserved codes: **0x00** (**V_UNASSIGNED**), **0x01** (**V_BOOT**), **0x02** (**V_ROOT**), **0x03** (**V_SWAP**), **0x04** (**V_USR**), **0x05** (**V_BACKUP**), **0x06** (**V_STAND**), **0x07** (**V_VAR**) and **0x08** (**V_HOME**).

flag The flag allows a partition to be flagged as unmountable or read only, the masks being: **V_UNMNT 0x01**, and **V_RONLY 0x10**. For mountable partitions use **0x00**.

starting sector The sector number (decimal) on which the partition starts.

size in sectors The number (decimal) of sectors occupied by the partition.

Note that you can save the output of a **prtvtoc** command to a file, edit the file, and use it as the *datafile* argument to the **–s** option.

x86 Options

The functionality provided by the following two x86 options is also provided by **installboot**(1M). Because the functionality described here may be removed in future versions of **fmthard**, you should use **installboot** to install boot records. The following options currently apply to **fmthard**:

–p *pboot* This option allows the user to override the default partition boot file, **/usr/platform/***platform-name***/lib/fs/ufs/pboot**. The partition boot file is platform dependent, where *platform-name* can be determined using the **–i** option to **uname**(1).

–b *bootblk* This option allows the user to override the default *bootblk* file, **/usr/platform/***platform-name***/lib/fs/ufs/bootblk**. The boot block file is platform dependent, where *platform-name* can be determined using the **–i** option to **uname**(1).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

uname(1), **format**(1M), **prtvtoc**(1M), **attributes**(5)

x86 Only

fdisk(1M), **installboot**(1M)

NOTES

Special care should be exercised when overwriting an existing VTOC, as incorrect entries could result in current data being inaccessible. As a precaution, save the old VTOC.

fmthard cannot write a disk label on an unlabeled disk. Use **format**(1M) for this purpose.

NAME

fncheck – check for consistency between FNS data and NIS+ data

SYNOPSIS

fncheck [**-r**] [**-s**] [**-u**] [**-t** *type*] [*domain_name*]

DESCRIPTION

fncheck is used for checking for inconsistencies between FNS username or hostname contexts and the contents of the corresponding NIS+ *passwd.org_dir* or *hosts.org_dir* tables, respectively, in the NIS+ domain *domain_name.* If *domain_name* is omitted, the domain name of the current machine is used. By default (in the absense of the **-r** and **-s** options), the following inconsistencies are displayed:

- items that appear only in the FNS context but do not appear in the NIS+ table,
- items that appear only in the NIS+ table but do not appear in the FNS context.

OPTIONS

-r Display only items that appear in the FNS context but do not appear in the corresponding NIS+ table.

-s Display items that appear in the NIS+ table but do not appear in the corresponding FNS context.

-u Update the FNS context based on information in the corresponding NIS+ table. If the **-r** option is used, items that appear only in the FNS context are removed from the FNS context. If the **-s** option is used, items that appear only in the NIS+ table are added to the FNS context. If neither **-r** or **-s** are specified, items are added and removed from the FNS context to make it consistent with the corresponding NIS+ table.

-t *type* Specify the type of context to check. *type* can be either **hostname** or **username.** If this option is omitted, both hostname and username contexts are checked. If *type* is **hostname,** the FNS hostname context is checked against the NIS+ *hosts.org_dir* table. If *type* is **username,** the FNS username context is checked against the NIS+ *passwd.org_dir* table.

USAGE

Although **fncheck** can be used to add users and hosts to the username and hostname contexts as new users and hosts are added to NIS+, that is not its intended purpose. **fncheck** is an expensive operation because it makes complete comparisons of the NIS+ table and the corresponding FNS context. When a user or host is added or removed from NIS+ using **admintool** (see **admintool**(1M)), it automatically updates the appropriate FNS contexts.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

admintool(1M), **fncreate**(1M), **fndestroy**(1M), **nis**(1), **attributes**(5), **fns**(5), **fns_policies**(5)

NAME

fncopy – copy FNS contexts, possibly from one naming service to another naming service

SYNOPSIS

fncopy [–**f** *filename*] [–**i** *old-naming-service*] [–**o** *new-naming-service*]
old-fns-context new-fns-context

DESCRIPTION

fncopy copies recursively the FNS context, *old-fns-context*, and attributes to a new FNS context, *new-fns-context*. If –**i** and –**o** options are specified with the respective naming service, the *old-fns-context* with be resolved using *old-naming-service* as the underlying naming service, and *new-fns-context* will be created using *new-naming-service* as the underlying naming service. In the absence of –**i** and –**o** options, the default naming service will be used (see **fnselect**(1M)).

When the –**f** option is used, *filename* names a file containing a list of contexts in the *old-fns-context* that should be copied to the *new-fns-context*.

If the FNS context *new-fns-context* already exists in the target naming service, *new-naming-service*, this command will copy only the contexts and bindings that do not exist in the target naming service. This command will not over-write any of the existing FNS contexts in the target naming service.

This command follows links and copies FNS contexts and binding to the *new-fns-context* namespace.

OPTIONS

–**f** *filename*	Specifies a file name that contains a list of FNS contexts to be copied.
–**i** *old-naming-service*	Specifies the source naming service; currently only **nis** is supported.
–**o** *new-naming-service*	Specifies the target naming service; currently only **nisplus** is supported.

OPERANDS

The following operands are supported:

old-fns-context	The current FNS context.
new-fns-context	The new FNS context.

EXAMPLES

For example, the command

```
eg% fncopy .../fed-naming.eng.sun.com/service/printer \
        .../sun.com/orgunit/ssi.eng/service/printer
```

will copy the FNS printer context **.../fed-naming.eng.sun.com/service/printer** and its subcontexts and bindings to the FNS printer context **.../sun.com/orgunit/ssi.eng/service/printer**.

In the following example,

```
eg% fncopy –i nis –o nisplus –f /etc/ssi-users-list \
        thisorgunit/user org/ssi.eng/user
```

will copy the NIS FNS users' contexts specified in the file **/etc/ssi-users-list** to NIS+ FNS users' context of the orgunit **ssi.eng**.

EXIT STATUS

0 Operation was successful.

1 Operation failed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

fnbind(1), **fnunbind**(1), **fncreate**(1M), **fncreate_fs**(1M), **fncreate_printer**(1M), **fndestroy**(1M), **attributes**(5), **fns**(5)

NAME

fncreate – create an FNS context

SYNOPSIS

fncreate **–t** *context_type* [**–Dosv**] [**–f** *input_file*] [**–r** *reference_type*] *composite_name*

DESCRIPTION

fncreate creates an FNS context of type *context_type*, where a *context_type* must be one of **org**, **hostname**, **host**, **username**, **user**, **service**, **fs**, **site**, **nsid**, or **generic**. It takes as the last argument a composite name, *composite_name*, for the context to be created. In addition to creating the context named, **fncreate** also creates subcontexts of the named context using FNS Policies of what types of contexts should be bound in those contexts. See **fns_policies**(5).

fncreate discovers which naming service is in use and creates contexts in the appropriate naming service. When FNS is being initially set up, it will by default create contexts for NIS+. This default can be changed by the use of **fnselect**(1M) to explicitly select a naming service.

When using FNS for a NIS+ environment, **fncreate** creates NIS+ tables and directories in the NIS+ hierarchy. See **fns_nis+**(5) for more information on the necessary NIS+ credentials and the use of the environment variable **NIS_GROUP** when using **fncreate** and other FNS commands.

When using FNS for a NIS environment, **fncreate** creates NIS maps and hence must be executed as superuser on the NIS master of the FNS-related maps. See **fns_nis**(5) for more information specific to the use of FNS in a NIS environment.

When using FNS for an environment that uses **/etc** files for its naming information, **fncreate** creates files in the **/var/fn** directory. See **fns_files**(5) for more information specific to the use of FNS for files.

OPTIONS

–t *context_type* The following are valid entries for *context_type*:

org Create organization context, and default subcontexts, for an existing NIS+ domain, NIS domain, or **/etc** files environment.

For NIS+, *composite_name* is of the form *org/domain/* where *domain* is a NIS+ domain. An empty domain name indicates the creation of the organization context for the root NIS+ domain; otherwise, the domain name names the corresponding NIS+ domain. *domain* can be either the fully-qualified NIS+ domain name — dot ('.')-terminated — or the NIS+ domain name named relative to the NIS+ root domain.

For example, the following creates the root organization context and its subcontexts for the NIS+ root domain **Wiz.Com.**:

```
eg% fncreate –t org org//
```

The same thing could have been achieved using the following command:

```
eg% fncreate −t org org/Wiz.COM./
```

Typically, this is the first FNS context created.

To create the organization context for a subdomain of **Wiz.COM.**, execute either of the following commands:

```
eg% fncreate −t org org/sales/
```

or

```
eg% fncreate −t org org/sales.Wiz.COM./
```

Note that if the corresponding NIS+ domain does not exist, **fncreate** fails. See **nissetup**(1M) for setting up a NIS+ domain.

A **ctx_dir** directory is created under the directory of the organization named.

For NIS or an **/etc** files environment, *domain* should be **NULL** (empty) because NIS and **/etc** files do not support a hierarchy namespace of domains. For example, the following command creates the organization context for the NIS or **/etc** files environment:

```
eg% fncreate −t org org//
```

For NIS+, NIS, and **/etc** files, creating the organization context also creates the organization's immediate subcontexts **host**, **user**, and **service** and their subcontexts. This includes a context for every host entry in the corresponding hosts database of the naming service (that is, **hosts.org_dir** NIS+ table, or **hosts** NIS map, or **/etc/hosts** file), and a context for every user entry in the **passwd** database of the naming service (that is, **passwd.org_dir** NIS+ table, or **passwd** NIS map, or **/etc/passwd** file) unless the option −**o** is specified. Bindings for these subcontexts are recorded under the organization context.

hostname

Create a hostname context in which atomic host names can be bound, and bind the reference of the context to *composite_name*. If the suffix of *composite_name* is **host/**, the hostname context created is also bound to the composite name with this suffix replaced by **_host/**, and the reverse (that is, if a composite name with a **_host/** suffix was supplied, a binding would be created for **host/**). Also create a host context for every host entry in the corresponding hosts database of the naming service (**hosts.org_dir** NIS+ table, or **hosts** NIS map, or **/etc/hosts** file), unless either option −**o** or −**f** is specified. The following example creates host contexts for all hosts in the **sales** organization:

```
eg% fncreate –t hostname org/sales/host/
```

Typically, a hostname context need not be created explicitly since it is created by default, as a subcontext under **org**.

host Create a host context for a specific host, and its **service** and **fs** subcontexts, and bind the reference of the context to *composite_name*. For example, the following creates a host context and **service** and **fs** subcontexts for host **sylvan**:

```
eg% fncreate –t host org/sales/host/sylvan/
```

username
Create a username context in which atomic user names can be bound, and bind the reference of the context to *composite_name*. If the suffix of *composite_name* is **user/**, the username context created is also bound to the composite name with this suffix replaced by **_user/**, and the reverse. Also create a user context for every user entry in the corresponding **passwd** database of the naming service (that is, **passwd.org_dir** NIS+ table, or **passwd** NIS map, or **/etc/passwd** file), unless either the option –**o** or –**f** is specified. The following example creates username contexts for all users in the **sales** organization:

```
eg% fncreate –t username org/sales/user/
```

Typically, a username context need not be created explicitly since it is created by default, as a subcontext under **org**.

user Create a user context for a specific user, and its **service** and **fs** subcontexts, and bind the reference of the context to *composite_name*. For example, the following creates a user context and **service** and **fs** subcontexts for user **jsmith**:

```
eg% fncreate –t user org/sales/user/jsmith/
```

service Create a service context in which slash-separated left-to-right service names can be bound, and bind the reference of the context to *composite_name*. If the suffix of *composite_name* is **service/**, the service context created is also bound to the composite name with this suffix replaced by **_service/**, and the reverse. Typically, a service context need not be created explicitly since it is created by default, as a subcontext under **org**, **host**, or **user** contexts.

fs Create a file system context for a user or host, and bind the reference of the context to *composite_name*. The composite name must be the name of a host or a user, with either **fs/** or **_fs/** appended to it. If the suffix of *composite_name* is **fs/**, the file system context created is also bound to the composite name with this suffix replaced by **_fs/**, and the reverse.

Typically, a file system context need not be created explicitly since it is created by default, as a subcontext of a user or host context.

The file system context of a user is the user's home directory as stored in the **passwd** database of the naming service (that is, in NIS+ table **passwd.org_dir**, or **passwd** NIS map, or **/etc/passwd** file). The file system context of a host is the set of NFS file systems that the host exports.

Use the **fncreate_fs**(1M) command to create file system contexts for organizations and sites, or to create file system contexts other than the defaults for users and hosts.

site Create a site context in which dot-separated right-to-left site names can be bound, and a service subcontext, and bind the reference of the context to *composite_name*. If the suffix of *composite_name* is **site/**, the hostname context created is also bound to the composite name with this suffix replaced by **_site/**, and the reverse. Typically, a **site** context is created at the same level as the **org** context and is used for creating a geographical namespace that complements the organizational namespace of an enterprise.

nsid Create a context in which namespace identifiers can be bound. This context has a flat namespace, in which only atomic names can be bound. An example of such a context is the context to which the name **site/east/** is bound. This context can have the following bindings: **site/east/host**, **site/east/user**, and **site/east/service**.

generic

Create a generic context in which slash-separated left-to-right names can be bound, and bind the reference of the context to *composite_name*. The option **–r** can be used to specify the reference type to be associated with the context. If the **–r** option is omitted, the reference type used is the reference type of the parent context if the parent context is a generic context; otherwise, the reference type is **onc_fn_generic**.

–f *input_file* Create a context for every user or host listed in **input_file**. This option is only applicable when used with the **–t username** or **–t hostname** options. The format of the file is an atomic user name or host name per line. This option is used to create contexts for a subset of the **users/hosts** found in the corresponding **passwd** or **hosts** database of the naming service (that is, for NIS+ these are the **passwd.org_dir** or **hosts.org_dir** tables, respectively). If this option is omitted, **fncreate** creates a context for every **user/host** found in the corresponding **passwd** or **hosts** database.

−r *reference_type* Use *reference_type* as the reference type of the generic context being created. This option can be used only with the **−t generic** option.

−D Information about the creation of a context, and corresponding NIS+ directories and tables, or NIS maps, or files entry, is displayed as each context is created.

−o Only the context named by *composite_name* is created; no subcontexts are created. When this option is omitted, subcontexts are created according to the FNS Policies for the type of the new object.

−s Create the context and bind it in to supercede any existing binding associated with *composite_name*. If this option is omitted, **fncreate** fails if *composite_name* is already bound.

−v Information about the creation of a context is displayed as each context is created.

OPERANDS

The following operand is supported:

composite_name An FNS named object.

EXAMPLES

The following examples illustrate creation of a host context in the root organization and a user context in a sub-organization.

Create a context, and subcontexts, for the root organization:

```
eg% fncreate −t org org//
```

It causes the following commands to be invoked automatically:

```
eg% fncreate −t service org//service/
eg% fncreate −t hostname org//host/
eg% fncreate −t username org//user/
```

Create a context, and subcontexts, for host **sylvan**:

```
eg% fncreate −t host org//host/sylvan/
```

It causes the following commands to be invoked automatically:

```
eg% fncreate −t service org//host/sylvan/service/
eg% fncreate −t fs org//host/sylvan/fs/
```

Create a context, and subcontexts, associated with a sub-organization **dct**:

```
eg% fncreate −t org org/dct/
```

It causes the following commands to be invoked automatically:

```
eg% fncreate −t service org/dct/service/
eg% fncreate −t hostname org/dct/host/
eg% fncreate −t username org/dct/user/
```

Create a context, and subcontexts, for user **msmith**:

```
eg% fncreate −t user org/dct/user/msmith/
```

It causes the following commands to be invoked automatically:

```
eg% fncreate –t service org/dct/user/msmith/service/
eg% fncreate –t fs org/dct/user/msmith/fs/
```

The following examples create service contexts:

```
eg% fncreate –t service org/dct/service/fax
eg% fncreate –t service org/dct/service/fax/classA
```

EXIT STATUS

0 Operation was successful.

1 Operation failed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

nis(1), **fncheck**(1M), **fncreate_fs**(1M), **fndestroy**(1M), **fnselect**(1M), **nissetup**(1M), **xfn**(3N), **attributes**(5), **fns**(5), **fns_files**(5), **fns_nis**(5), **fns_nis+**(5), **fns_policies**(5), **fns_references**(5)

NAME

fncreate_fs – create FNS file system contexts

SYNOPSIS

fncreate_fs [**–r**] [**–v**] **–f** *input_file composite_name*

fncreate_fs [**–r**] [**–v**] *composite_name* [*mount_options*] [*mount_location* ...]

DESCRIPTION

The **fncreate_fs** command creates or updates the FNS file system context named by *composite_name*. A description of the context's bindings is provided in *input_file* if the first form of the command is used, or is given on the command line if the second form is used.

OPTIONS

–r Replace the bindings in the context named by *composite_name* with only those specified in the input. This is equivalent to destroying the context (and, recursively, its subcontexts), and then running **fncreate_fs** without this option. This option should be used with care.

–v Verbose. Display information about the contexts being created and modified.

–f *input_file* Read input from *input_file*. If *input_file* is '**-**' (hyphen), read from standard input instead.

OPERANDS

The following operand is supported:

composite_name An FNS named object.

USAGE

The **fncreate_fs** command populates the file system portions of the FNS namespace. The automounter (see **automount**(1M)) will then "mount" the FNS namespace under **/xfn**. The directory with the FNS name **org/engineering/fs**, for example, can be found on the file system as **/xfn/org/engineering/fs**.

The format of the input to **fncreate_fs** is similar, but not identical, to the format of indirect automount maps. Differences are enumerated in the **NOTES** section below.

Input File Format

The input file supplies the names and values to be bound in the context of *composite_name*. Its format is a sequence of lines of the form:

name [-*options*] [*location* ...]

For each such entry, a reference to the *location*(s) and the corresponding *options* is bound to the name *composite_name*/*name*. The *name* field may be a simple atomic name, a slash-separated hierarchical name, or '**.**' (period). If it is '**.**' then the reference is bound directly to *composite_name*. The *name* field must not begin with a slash.

The *location* field specifies the host or hosts that serve the files for *composite_name*/*name*. In the case of a simple NFS mount, *location* takes the form:

host:*path*

where *host* is the name of the host from which to mount the file system, and *path* is the path name of the directory to mount.

The *options* field is a comma-separated list of the mount options to use when mounting the location bound to *composite_name***/***name*. These options also apply to any subcontexts of *composite_name***/***name* that do not specify their own mount options. If *options* is given but *location* is not, the options apply to subcontexts only.

If neither *options* nor a *location* is given, then no reference is bound to *composite_name***/***name*. Any existing reference is unbound.

A single logical line may be continued across multiple input lines by escaping the newline with a '\' (backslash). Comments begin with a '**#**' that is either at the beginning of a line or is prefixed by whitespace, and end at the end of the line.

Command-line Input

If no *input_file* is specified on the command line, then the *options* and *location* fields given on the command line are bound directly to *composite_name*. This is equivalent to providing a one-line input file with a '**.**' in the *name* field.

Multiple Locations

Multiple *location* fields may be specified for NFS file systems that are exported from multiple, functionally-equivalent locations. If several locations in the list share the same path name, they may be combined using a comma-separated list of host names:

*host1***,** *host2***, . . . :** *path*

The hosts may be weighted, with the weighting factor appended to the host name as a non-negative integer in parentheses: the lower the number, the more desirable the server. The default weighting factor is **0** (most desirable). In the example:

alpha,bravo,charlie(1),delta(2):/usr/man

hosts **alpha** and **bravo** are the most desirable; host **delta**, the least.

See the **USAGE** section of **automount**(1M) for additional information on how the automounter interprets the location field.

Variable Substitution

Variable names, prefixed by '**$**', may be used with the *options* or *location* fields. For example, a *location* may be given as:

svr1:/export/$CPU

The automounter will substitute client-specific values for these variables when mounting the corresponding file systems. In the above example, **$CPU** is replaced by the output of **uname −p**; for example, "**sparc**". See the **USAGE** section of **automount**(1M) for more information on how the automounter treats variable substitution.

Alternate Input Format

For additional compatibility with automount maps (see **automount**(1M)), the following input format is accepted:

```
name              [options] [location ...] \
        /offset1  [options1] location1 ... \
        /offset2  [options2] location2 ... \
        ...
```

where each *offset* field is a slash-separated hierarchy. This is interpreted as being equivalent to:

```
name            [options] [location ...ˆ]
name/offset1    [options1] location1 ...
name/offset2    [options2] location2 ...
...
```

(the first line being omitted if both *options* and *location* are omitted).

This format is for compatibility only; it provides no additional functionality. Its use is deprecated.

EXAMPLES

The following examples illustrate the use of the **fncreate_fs** command.

The call:

```
example%  cat input1
src       –ro       svr1:/export/src
dist      –ro       svr2,svr3:/export/dist
example%  fncreate_fs –f input1 org/engineering/fs
```

creates a file system context for the engineering organization. It specifies that **org/engineering/fs/src** is a read-only NFS mount from server **svr1**, and that **org/engineering/fs/dist** is a read-only NFS mount from either **svr2** or **svr3**.

Once this is done, there are several equivalent ways to create the engineering organization's **src/cmd** context. It could be done using the composite name **org/engineering/fs**:

```
example%  cat input2
src/cmd             svr1:/export/cmd
example%  fncreate_fs –f input2 org/engineering/fs
```

Equivalently, it could be done using the composite name **org/engineering/fs/src**:

```
example%  cat input3
cmd                 svr1:/export/cmd
example%  fncreate_fs –f input3 org/engineering/fs/src
```

The same results could also be achieved by:

```
example%  fncreate_fs org/engineering/fs/src/cmd svr1:/export/cmd
```

Note that **cmd** will also be mounted read-only, since it is a subcontext of **src** and does not have mount options of its own.

In the first example of this section, the –**ro** mount option was specified for each entry in the input file. It could instead have been specified only once:

```
example%  cat input4
.         –ro
src                 svr1:/export/src
dist                svr2,svr3:/export/dist
example%  fncreate_fs –f input4 org/engineering/fs
```

The –**ro** option here applies to all bindings in the context **org/engineering/fs** and any of its subcontexts. In particular, it also applies to the **cmd** context from the above examples.

The following will change the NFS server for the **src** context:

```
example% fncreate_fs org/engineering/fs/src svr4:/export/src
```

Had the **–r** option been used, the **cmd** subcontext would have been destroyed as well:

```
example% fncreate_fs –r org/engineering/fs/src svr4:/export/src
```

Only the FNS context is destroyed. The **/export/cmd** directory on **svr1** is not affected.

The file system contexts of users and hosts are not usually created by **fncreate_fs** (see the **NOTES** section below). The defaults set by **fncreate**, however, may be overridden. For example, the call:

```
example% fncreate_fs user/jane/fs svr1:/export/home/jane
```

sets Jane's file system to be an NFS mount from **svr1**.

EXIT STATUS

0 Operation was successful.

1 Operation failed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

fnbind(1), **fnlist**(1), **fnlookup**(1), **fnunbind**(1), **nis+**(1), **automount**(1M), **fncreate**(1M), **fndestroy**(1M), **attributes**(5), **fns**(5), **fns_files**(5), **fns_nis**(5), **fns_nis+**(5), **fns_policies**(5)

NOTES

The **fncreate_fs** command affects the FNS file system namespace only. It does not have any effect on the servers that export the files and directories from which the namespace is constructed. Destroying an FNS context does not remove any files on any server.

FNS policies specify that file system contexts are bound after the namespace identifier **fs** in composite names (see **fns_policies**(5)). Therefore, *composite_name* must contain an **fs**. The alias **_fs** may be used in place of **fs**.

The context named by the components of *composite_name* preceding **fs** must exist prior to the call to **fncreate_fs**, since **fncreate_fs** creates *only* file system contexts.

Default file system contexts for hosts and users are generally created by the command **fncreate**(1M). These defaults may be overridden using **fncreate_fs**. Overriding a host's default file system context is unlikely to make sense.

The input file format is similar to the format of indirect automount maps (see **automount**(1M)). The differences are:

- the *name* field may be hierarchical, and may be '**.**'
- there are no included maps or special maps
- there may be entries with neither options nor locations
- the characters '*' and '**&**' have no special meaning

The process executing the **fncreate_fs** command may need certain credentials to update information in the underlying naming service. See **fns_nis**(5), **fns_nis+**(5), and **fns_files**(5) for more information.

NAME

fncreate_printer – create new printers in the FNS namespace

SYNOPSIS

fncreate_printer [**–sv**] *compositename printername printeraddr* [*printeraddr* ...]

fncreate_printer [**–sv**] [**–f** *filename*] *compositename*

DESCRIPTION

fncreate_printer creates a new printer context for an *organization*, *user*, *host*, or *site* object. *compositename* is the FNS name of the object. **fncreate_printer** uses *printername* to name the new printer and binds it to an FNS reference constructed from the set of *printeraddr*s. **fncreate_printer** may also be used to add new *printeraddr*s for an existing *printername*.

The command also supports creating a set of printers as listed in the file *filename*.

The new printer context is created with the FNS name <*compositename*>**/service/printer/**<*printername*>. If the intermediate **service** or **printer** names do not exist, their FNS contexts are also created by this command. Normally, these intermediate contexts would be created by an administrative script that uses **fncreate**(1M), and is run at the time a new FNS organization is set up. The reference bound to the FNS printer name is of type **onc_printers** and is constructed from the set of *printeraddr*s. A *printeraddr* is of the form <*addresstype*> = <*address*>. See **printers.conf**(4) for the format of *printeraddr* and also the examples below for currently supported address types and address strings.

An FNS printer name is accepted as a valid printer name by **lp**(1), **lpstat**(1), **cancel**(1), **lpmove**(1M), **lpr**(1B), **lpq**(1B), and **lprm**(1B).

The *printername* argument may be a slash-separated name. In this case, prior to creating the printer context denoted by the ‘‘leaf’’ name, this command will create printer context(s) for the intermediate node(s) if they do not already exist. See **EXAMPLES**.

fncreate_printer creates entries in the naming service determined by **fnselect**(1M). See **fnselect**(1M) for more information on the default naming service and on selecting a naming service. Furthermore, the process executing the **fncreate_printer** command may require certain credentials to update information in the underlying namespace. See **fns_nis+**(5), **fns_nis**(5), and **fns_files**(5) for more information.

OPTIONS

–s The new *address* supersedes an existing address with the same *addresstype*, if any, for <*compositename*>**/service/printer/**<*printername*>. If this option is omitted, it appends the *printeraddr* to an existing reference, or creates a new reference using *printeraddr* for the printer.

–v Displays information about individual printer contexts as they are created.

–f *filename* Use *filename* to obtain a list of printers for which to create contexts. If this option is omitted, **/etc/printers.conf** is used as the input file, in which case the **–s** option should be used to supersede the entries already present in this file.

OPERANDS

filename The file that contains a list of printers to be created. This file uses the same format as **/etc/printers.conf**. See **printers.conf**(4) for more information.

printername The name of the new printer context created.

printeraddr An address to be associated with the printer context name.

compositename The FNS name for the **org**, **host**, **user**, or **site** object for which the new printer contexts are created.

EXAMPLES

The following examples illustrate creating a set of printer contexts under an organization, a printer context for a user, and a printer context associated with a hierarchical printer name for a site, respectively.

Example 1

Create printers for an organization:

```
example% fncreate_printer –s org/marketing
```

This causes the creation of a printer context for every entry listed in the **/etc/printers.conf** file on the system where the command is executed. The printer contexts thus created are bound under the organization's printer context, **org/marketing/service/printer**. The **–s** flag is required to force the creation of the printer contexts in the underlying namespace, since the default **/etc/printers.conf** file is being used.

Example 2

Create a printer named **ps** for user **jsmith** and associate it with the **killtree** printer served by the print server **paperwaster**:

```
example% fncreate_printer –s usr/jsmith ps bsdaddr=paperwaster,killtree
```

This causes **jsmith**'s **ps** printername to be associated with the **killtree** printer on the server **paperwaster**, overwriting any existing address of type **bsdaddr**. The user can print to this printer using the command:

```
example% lp –d thisuser/service/printer/ps <filename>
```

Example 3

Create a printer with the hierarchical name **color/fast** under a site:

```
example% fncreate_printer site/bldg14/northwing color/fast \
                    bsdaddr=paperwaster,laser
```

This causes the printer named **site/bldg14/northwing/service/printer/color/fast** to be associated with the **laser** printer on server **paperwaster**. If the intermediate printer context **site/bldg14/northwing/service/printer/color** does not exist, it will also be created and associated with the same printer. If the printer name **site/bldg14/northwing/service/printer/color/fast** already exists and has an address of type **bsdaddr** associated with it, this command will fail.

EXIT STATUS

0 Successful operation.

1 Operation failed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

cancel(1), **lp**(1), **lpq**(1B), **lpr**(1B), **lprm**(1B), **lpstat**(1), **fncreate**(1M), **fnselect**(1M), **lpmove**(1M), **printers**(4), **printers.conf**(4), **attributes**(5), **fns**(5), **fns_files**(5), **fns_nis**(5), **fns_nis+**(5)

NAME

fndestroy – destroy an FNS context

SYNOPSIS

fndestroy *composite_name*

DESCRIPTION

fndestroy removes the context bound to *composite_name*. The context is not removed if there are subcontexts associated with *composite_name*.

EXAMPLES

The command

```
eg% fndestroy user/jsmith/
```

destroys the context named by **user/jsmith/** and removes the binding of **jsmith** from the context **user/**.

This command fails if the context **user/jsmith/** contains subcontexts, or if the invoker does not have the NIS+ credentials required to delete the NIS+ tables that store the user's bindings. See **fns**(5).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

fnlist(1), **fnlookup**(1), **fnunbind**(1), **fncreate**(1M), **attributes**(5), **fns**(5), **fns_policies**(5)

NAME

fnselect – select a specific naming service to use for the FNS Initial Context

SYNOPSIS

fnselect [–**D**]
fnselect *naming-service*

DESCRIPTION

fnselect is used to set the specified naming service to be used to construct the bindings in the FNS Initial Context. This setting affects the entire machine and affects applications that make subsequent calls to **fn_ctx_handle_from_initial**(3N). This setting can be changed only by an administrator who has root privilege on the machine.

OPTIONS

–**D** Displays the actual naming service used to generate the FNS Initial Context.

OPERANDS

naming-service The following are possible values for *naming-service*:

default	Use the FNS default algorithm for determining the target naming service.
nisplus	Use NIS+ as the target naming service.
nis	Use NIS as the target naming service.
files	Use **/etc** files as the target naming service.

USAGE

When the **default** option is selected, FNS determines the underlying naming service using the following algorithm:

- First, it checks for NIS+ with FNS installed.
- If the result is **TRUE**, then FNS assumes **nisplus** as the underlying naming service.
- Otherwise, it checks if the system is a NIS client.
- If **TRUE**, FNS assumes **nis** as the underlying naming service.
- Otherwise, FNS assumes **/etc files**.

fnselect without any arguments displays the service currently selected for the Initial Context (one of **default**, **nisplus**, **nis**, or **files**).

When the –**D** option is specified and the current setting is **default**, **fnselect** will use the algorithm that is used by FNS and display the actual naming service used for the FNS Initial Context.

EXAMPLES

The command

```
eg% fnselect nisplus
```

will select NIS+ as the underlying naming service for the FNS Initial Context.

The command

```
eg% fnselect
```

will print the naming service currently being used to generate the FNS Initial Context.

EXIT STATUS

0 Operation was successful.

1 Operation failed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

fnbind(1), **fnlist**(1), **fnlookup**(1), **fnunbind**(1), **fncreate**(1M), **fncreate_fs**(1M), **fncreate_printer**(1M), **fndestroy**(1M), **fn_ctx_handle_from_initial**(3N), **attributes**(5), **fns**(5), **fns_initial_context**(5)

NAME

fnsypd – update FNS context on an NIS master server

SYNOPSIS

/usr/sbin/fnsypd

DESCRIPTION

The **fnsypd** daemon is a Remote Procedure Call (RPC) service that accepts requests from NIS clients to update and modify Federated Naming Service (FNS) contexts. This daemon runs on an NIS master server with FNS on top of it. The **fnsypd** daemon requires the Secure Key Infrastructure (SKI) mechanism for authentication. The SKI mechanism is part of the SUNWski package. If SUNWski is not installed, authentication cannot be performed and users will receive "permission denied" error messages. The SUNWski man pages are located at **/opt/SUNWski/man**.

fnsypd enables users and hosts to modify only their respective FNS contexts. Organization, site, hostname and username contexts cannot be modified using **fnsypd**.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
1	An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfns

SEE ALSO

nis(1), **attributes**(5), **fns**(5), **fns_policies**(5)

NAME

format – disk partitioning and maintenance utility

SYNOPSIS

format [–**f** *command-file*] [–**l** *log-file*] [–**x** *data-file*] [–**d** *disk-name*] [–**t** *disk-type*]
[–**p** *partition-name*] [–**s**] [–**m**] [–**M**] [–**e**] [*disk-list*]

DESCRIPTION

format enables you to format, label, repair and analyze disks on your system. Unlike previous disk maintenance programs, **format** runs under SunOS. Because there are limitations to what can be done to the system disk while the system is running, **format** is also supported within the memory-resident system environment. For most applications, however, running **format** under SunOS is the more convenient approach.

format first uses the disk list defined in *data-file* if the –**x** option is used. **format** then checks for the **FORMAT_PATH** environment variable, a colon-separated list of filenames and / or directories. In the case of a directory, **format** searches for a file named **format.dat** in that directory; a filename should be an absolute pathname, and is used without change. **format** adds all disk and partition definitions in each specified file to the working set. Multiple identical definitions are silently ignored. If **FORMAT_PATH** is not set, the path defaults to **/etc/format.dat**.

disk-list is a list of disks in the form **c?t?d?** or **/dev/rdsk/c?t?d?s?**. With the latter form shell wildcard specifications are supported. For example, specifying **/dev/rdsk/c2**∗ will cause **format** to work on all drives connected to controller **c2** only. If no *disk-list* is specified, **format** lists all the disks present in the system.

OPTIONS

The following options are supported:

–**f** *command-file* Take command input from *command-file* rather than the standard input. The file must contain commands that appear just as they would if they had been entered from the keyboard. With this option, **format** does not issue **continue?** prompts; there is no need to specify y(es) or n(o) answers in the *command-file*. In non-interactive mode, **format** does not initially expect the input of a disk selection number. The user must specify the current working disk with the –**d** *disk-name* option when format is invoked, or specify **disk** and the disk selection number in the *command-file*.

–**l** *log-file* Log a transcript of the **format** session to the indicated *log-file*, including the standard input, the standard output and the standard error.

–**x** *data-file* Use the list of disks contained in *data-file*.

–**d** *disk-name* Specify which disk should be made current upon entry into the program. The disk is specified by its logical name (for instance, –**c0t1d0**). This can also be accomplished by specifying a single disk in the disk list.

–**t** *disk-type* Specify the type of disk which is current upon entry into the program, A disk's type is specified by name in the data file. This option can only be used if a disk is being made current as described above.

–p *partition-name*
Specify the partition table for the disk which is current upon entry into the program. The table is specified by its name as defined in the data file. This option can only be used if a disk is being made current, and its type is either specified or available from the disk label.

–e Enable SCSI expert menu. Note this option is not recommended for casual use.

–s Silent. Suppress all of the standard output. Error messages are still displayed. This is generally used in conjunction with the **–f** option.

–m Enable extended messages. Provides more detailed information in the event of an error.

–M Enable extended and diagnostic messages. Provides extensive information on the state of an SCSI device's mode pages, during formatting.

USAGE

The **format** utility's main menu items allow you to do the following tasks:

analyze Run read, write, and compare tests.

backup Search for backup labels.

current Display the device name, manufacturer and model of the current disk.

defect Retrieve and print defect lists.

disk Choose the disk that will be used in subsequent operations (known as the current disk.)

fdisk Run the **fdisk**(1M) program to create a **fdisk** partition for Solaris software (x86 based systems only).

format Format and verify the current disk.

inquiry Display the vendor, product name, and revision level of the current drive.

label Write a new label to the current disk.

partition Create and modify slices.

quit Exit the format menu.

repair Repair a specific block on the disk.

save Save new disk and slice information.

type Identify the manufacturer and model of the current disk.

verify Print the following information about the disk: manufacturer and model, number of cylinders, alternate cylinders, heads and sectors, and partition table.

volname Label the disk with a new eight character volume name.

ENVIRONMENT

FORMAT_PATH a colon-separated list of filenames and / or directories of disk and partition definitions. If a directory is specified, **format** searches for the file **format.dat** in that directory.

FILES

/etc/format.dat default data file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fmthard(1M), **prtvtoc**(1M), **format.dat**(4), **attributes**(5), **ipi**(7D), **sd**(7D)

See *Disk Management* in *System Administration Guide*.

x86 Only

fdisk(1M)

WARNINGS

When the **format** function is selected to format the Maxtor 207MB disk, the following message displays:

```
Mode sense page(4) reports rpm value as 0, adjusting it to 3600
```

This is a drive bug that may also occur with older third party drives. The above message is not an error; the drive will still function correctly.

NOTES

format provides a help facility you can use whenever format is expecting input. You can request help about what information is expected by simply entering a question mark (**?**) and **format** prints a brief description of what type of input is needed. If you enter a **?** at the menu prompt, a list of available commands is displayed.

For SCSI disks, formatting is done with both Primary and Grown defects list by default. However, if only Primary list is extracted in defect menu before formatting, formatting will be done with Primary list only.

NAME

fsck – check and repair file systems

SYNOPSIS

fsck [–**F** *FSType*] [–**m**] [–**V**] [*special* . . .]
fsck [–**F** *FSType*] [– **n** | **N** | **y** | **Y**] [–**V**] [–**o** *FSType-specific-options*]
[*special* . . .]

DESCRIPTION

fsck audits and interactively repairs inconsistent file system conditions. If the file system is inconsistent the default action for each correction is to wait for the user to respond **yes** or **no**. If the user does not have write permission **fsck** defaults to a **no** action. Some corrective actions will result in loss of data. The amount and severity of data loss may be determined from the diagnostic output.

FSType-specific-options are options specified in a comma-separated (with no intervening spaces) list of options or keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

special represents the character special device on which the file system resides, for example, **/dev/rdsk/c1t0d0s7**. Note: the character special device, not the block special device, should be used. **fsck** will not work on a block device if it is mounted.

If no *special* device is specified **fsck** checks the file systems listed in in **/etc/vfstab**. Those entries in **/etc/vfstab** which have a character special device entry in the **fsckdev** field and have a non-zero numeric entry in the **fsckpass** field will be checked. Specifying –**F** *FSType* limits the file systems to be checked to those of the type indicated.

If *special* is specified, but –**F** is not, the file system type will be determined by looking for a matching entry in **/etc/vfstab**. If no entry is found, the default local file system type specified in **/etc/default/fs** will be used.

If a file system type supports parallel checking, for example, **ufs,** some file systems eligible for checking may be checked in parallel. Consult the file system-specific man page (for example, **fsck_ufs**(1M)) for more information.

OPTIONS

–**F** *FSType* Specify the file system type on which to operate.

–**m** Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, **fsck** displays a message such as:

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s1 okay
```

–**n** | **N** Assume a **no** response to all questions asked by **fsck**; do not open the file system for writing.

–**V** Echo the expanded command line but do not execute the command. This option may be used to verify and to validate the command line.

–**y** | **Y** Assume a **yes** response to all questions asked by **fsck**.

−o *specific-options* These *specific-options* can be any combination of the following separated by commas (with no intervening spaces).

b=*n* Use block *n* as the super block for the file system. Block 32 is always one of the alternate super blocks. Determine the location of other super blocks by running **newfs**(1M) with the −**Nv** options specified.

c If the file system is in the old (static table) format, convert it to the new (dynamic table) format. If the file system is in the new format, convert it to the old format provided the old format can support the file system configuration. In interactive mode, **fsck** will list the direction the conversion is to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the file system. In preen mode, the direction of the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the first line of output from **fstyp**(1M).
Note: the **c** option is seldom used and is included only for compatibility with pre-4.1 releases. There is no guarantee that this option will be included in future releases.

f Force checking of file systems regardless of the state of their super block clean flag.

p Check and fix the file system non-interactively (“preen”). Exit immediately if there is a problem requiring intervention. This option is required to enable parallel file system checking.

w Check writable file systems only.

EXIT STATUS

0 file system is okay and does not need checking

1 erroneous parameters are specified

32 file system is unmounted and needs checking (**fsck** −**m** only)

33 file system is already mounted

34 cannot stat device

36 uncorrectable errors detected - terminate normally

37 a signal was caught during processing

39 uncorrectable errors detected - terminate immediately

40 for root, same as **0**.

USAGE

See **largefile**(5) for the description of the behavior of **fsck** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/default/fs default local file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: **LOCAL=ufs**.

LOCAL The default partition for a command if no FSType is specified.

/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

clri(1M), **crash**(1M), **fsck_cachefs**(1M), **fsck_s5fs**(1M), **fsck_ufs**(1M), **fsdb_ufs**(1M), **fsirand**(1M), **fstyp**(1M), **mkfs**(1M), **mkfs_ufs**(1M), **mountall**(1M), **newfs**(1M), **reboot**(1M), **fs_ufs**(4), **vfstab**(4), **attributes**(5), **largefile**(5)

WARNINGS

The operating system buffers file system data. Running **fsck** on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, the file system should be *unmounted* when **fsck** is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after **fsck** is run. Quite often, however, this will not be sufficient. A panic will probably occur if running **fsck** on a file system modifies the file system.

NOTES

This command may not be supported for all *FSTypes*.

Running **fsck** on file systems larger than 2 Gb fails if the user chooses to use the block interface to the device:

```
fsck /dev/dsk/c?t?d?s?
```

rather than the raw (character special) device:

```
fsck /dev/rdsk/c?t?d?s?
```

NAME

fsck_cachefs – check integrity of data cached with CacheFS

SYNOPSIS

fsck **–F** **cachefs** [**–m** | **–o** **noclean**] *cache_directory*

DESCRIPTION

The CacheFS version of the **fsck** command checks the integrity of a cache directory. By default it corrects any CacheFS problems it finds. There is no interactive mode. The most likely invocation of **fsck** for CacheFS file systems is at boot time from an entry in the **/etc/vfstab** file (see **vfstab**(4)).

OPTIONS

Two command line options are available:

–m Check, but do not repair.

–o noclean Force a check on the cache even if there is no reason to suspect there is a problem.

EXAMPLES

The following example forces a check on the cache directory **/cache3**:

```
example% fsck -F cachefs -o noclean /cache3
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cfsadmin(1M), **fsck**(1M), **mount_cachefs**(1M), **vfstab**(4), **attributes**(5)

NAME

fsck_s5fs – file system consistency check and interactive repair

SYNOPSIS

fsck –F s5fs [*generic_options*] [*special* . . .]
fsck –F s5fs [*generic_options*] [**–o** *specific-options*] [*special* . . .]

DESCRIPTION

fsck audits and interactively repairs inconsistent conditions on file systems. A file system to be checked may be specified by giving the name of the block or character special device or by giving the name of its mount point if a matching entry exists in **/etc/vfstab**. If no special device is specified, all s5 file systems specified in the **vfstab** with a **fsckdev** entry will be checked.

In the case of correcting serious inconsistencies, by default, **fsck** asks for confirmation before making a repair and waits for the operator to respond either **yes** or **no**. If the operator does not have write permission on the file system, **fsck** will default to a **–n** (no corrections) action. See **fsck**(1M).

Repairing some file system inconsistencies may result in loss of data. The amount and severity of data loss may be determined from the diagnostic output.

fsck automatically corrects innocuous inconsistencies such as unreferenced inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the superblock automatically. It displays a message for each inconsistency corrected that identifies the nature of the correction on which the file system took place. After successfully correcting a file system, **fsck** prints the number of files on that file system and the number of used and free blocks.

Inconsistencies checked are as follows:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Incorrect directory sizes.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks, file pointing to unallocated inode, inode number out of range, absence of '**.**' and '**..**' entries in any directory.
- Superblock checks: more blocks for inodes than there are in the file system.
- Bad free block list format.
- Total free block and / or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. If the **lost+found** directory does not exist, it is created.

OPTIONS

See generic **fsck**(1M) for *generic_options* and details for specifying *special*.

–o Specify **s5** file system specific options. These options can be any combination of the following separated by commas (with no intervening spaces):

f or **F**
Fast check; duplicate blocks and free list check only.

l After all other output is done, print i-number / pathname correspondences for damaged files.

t *scratchfile*
If there is insufficient memory and a temporary file is necessary to complete file system checking, use *scratchfile* as the temporary file.

T *scratchfile*
Same as above.

s *cyl:skip*
If it is necessary to rewrite (salvage) the free block list to correct an inconsistency, interleave the blocks such that, to the extent possible within each group of *cyl* consecutive free blocks, the interval between blocks is *skip*. For example, with an interleave of 8:3, in each group of eight consecutive free blocks, the order on the free list would be 1 4 7 2 5 8 3 6. If no *cyl:skip* is given, the value is either taken from the superblock, or, if unspecified (either has a value of 0), 400:7 is used. For obscure historical reasons, interleave specification of ‘‘3’’ and ‘‘4’’ (without colons) are taken to mean 200:5 and 418:7, respectively.

S *cyl:skip*
Same as above, except rewrite the free block list unconditionally.

q Quiet; produce less verbose output.

D Perform more extensive directory checking than normal.

p (‘‘preen’’) Check and fix the file system non-interactively. Exit immediately if there is a problem requiring intervention.

? Print usage message.

FILES

/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWs53

SEE ALSO

fsck(1M), **attributes**(5)

NOTES

It is usually faster to check the character special device than the block special device.

NAME

fsck_ufs – file system consistency check and interactive repair

SYNOPSIS

fsck –F ufs [*generic-options*] [*special* . . .]
fsck –F ufs [*generic-options*] [**–o** *specific-options*] [*special* . . .]

DESCRIPTION

The **fsck** utility audits and interactively repairs inconsistent conditions on file systems. A file system to be checked may be specified by giving the name of the block or character *special* device or by giving the name of its mount point if a matching entry exists in **/etc/vfstab**.

The *special* parameter represents the character special device, for example, **/dev/rdsk/c1t0d0s7**, on which the file system resides. The character special device, not the block special device should be used. The **fsck** utility will not work on a block device if the block device is mounted, unless the file system is error-locked.

If no *special* device is specified, all **ufs** file systems specified in the **vfstab** with a **fsckdev** entry will be checked. If the –**p** (‘‘preen’’) option is specified, **ufs** file systems with an **fsckpass** number greater than **1** are checked in parallel. See **fsck**(1M).

In the case of correcting serious inconsistencies, by default, **fsck** asks for confirmation before making a repair and waits for the operator to respond either **yes** or **no**. If the operator does not have write permission on the file system, **fsck** will default to a –**n** (no corrections) action. See **fsck**(1M).

Repairing some file system inconsistencies may result in loss of data. The amount and severity of data loss may be determined from the diagnostic output.

The **fsck** utility automatically corrects innocuous inconsistencies such as unreferenced inodes, too-large link counts in inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the super block. It displays a message for each inconsistency corrected that identifies the nature of the correction on the file system which took place. After successfully correcting a file system, **fsck** prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

Inconsistencies checked are as follows:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Incorrect directory sizes.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks, file pointing to unallocated inode, inode number out of range, and absence of ‘**.**’ and ‘**. .**’ as the first two entries in each directory.
- Super Block checks: more blocks for inodes than there are in the file system.
- Bad free block list format.
- Total free block and / or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. If the **lost+found** directory does not exist, it is created. If there is insufficient space in the **lost+found** directory, its size is increased.

An attempt to mount a **ufs** file system with the **–o nolargefiles** option will fail if the file system has ever contained a large file (a file whose size is greater than or equal to 2 Gbyte). Invoking **fsck** resets the file system state if no large files are present in the file system. A successful mount of the file system after invoking **fsck** indicates the absence of large files in the file system. An unsuccessful mount attempt indicates the presence of at least one large file. See **mount_ufs**(1M).

OPTIONS

The *generic-options* consist of the following options:

–m Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, **fsck** displays a message such as:

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s1 okay
```

–n | **N** Assume a **no** response to all questions asked by **fsck**; do not open the file system for writing.

–V Echo the expanded command line, but do not execute the command. This option may be used to verify and to validate the command line.

–y | **Y** Assume a **yes** response to all questions asked by **fsck**.

See generic **fsck**(1M) for the details for specifying *special*.

–o *specific-options* Specify **ufs** file system specific options. These options can be any combination of the following separated by commas (with no intervening spaces).

b=*n* Use block *n* as the super block for the file system. Block 32 is always one of the alternate super blocks. Determine the location of other super blocks by running **newfs**(1M) with the **–Nv** options specified.

c If the file system is in the old (static table) format, convert it to the new (dynamic table) format. If the file system is in the new format, convert it to the old format provided the old format can support the file system configuration. In interactive mode, **fsck** will list the direction the conversion is to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the file system. In preen mode, the direction of the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the first line of output from **fstyp**(1M).
Note: the **c** option is seldom used and is included only for

compatibility with pre-4.1 releases. There is no guarantee that this option will be included in future releases.

f Force checking of file systems regardless of the state of their super block clean flag.

p Check and fix the file system non-interactively (“preen”). Exit immediately if there is a problem requiring intervention. This option is required to enable parallel file system checking.

w Check writable file systems only.

FILES

/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

clri(1M), **crash**(1M), **fsck**(1M), **fsdb_ufs**(1M), **fsirand**(1M), **fstyp**(1M), **mkfs**(1M), **mkfs_ufs**(1M), **mount_ufs**(1M), **mountall**(1M), **newfs**(1M), **reboot**(1M), **fs_ufs**(4), **vfstab**(4), **attributes**(5), **largefile**(5)

WARNINGS

The operating system buffers file system data. Running **fsck** on a mounted file system can cause the operating system’s buffers to become out of date with respect to the disk. For this reason, the file system should be *unmounted* when **fsck** is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after **fsck** is run. Quite often, however, this will not be sufficient. A panic will probably occur if running **fsck** on a file system modifies the file system.

NOTES

It is usually faster to check the character special device than the block special device.

Running **fsck** on file systems larger than 2 Gb fails if the user chooses to use the block interface to the device:

```
fsck /dev/dsk/c?t?d?s?
```

rather than the raw (character special) device:

```
fsck /dev/rdsk/c?t?d?s?
```

NAME

fsdb – file system debugger

SYNOPSIS

fsdb [–**F** *FSType*] [–**V**] [–**o** *FSType-specific_options*] *special*

DESCRIPTION

fsdb is a file system debugger that allows for the manual repair of a file system after a crash. *special* is a special device used to indicate the file system to be debugged. **fsdb** is intended for experienced users only. *FSType* is the file system type to be debugged. Since different *FSTypes* have different structures and hence different debugging capabilities, the manual pages for the *FSType*-specific **fsdb** should be consulted for a more detailed description of the debugging capabilities.

OPTIONS

–**F** Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from **/etc/vfstab** by matching the *special* with an entry in the table, or by consulting **/etc/default/fs**.

–**V** Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from **/etc/vfstab**. This option may be used to verify and validate the command line.

–**o** Specify *FSType*-specific options.

USAGE

See **largefile**(5) for the description of the behavior of **fsdb** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/default/fs default local file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: LOCAL=ufs

LOCAL: The default partition for a command if no FSType is specified.

/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

vfstab(4), **attributes**(5), **largefile**(5)
Manual pages for the *FSType*-specific modules of **fsdb**.

NOTES

This command may not be supported for all *FSTypes*.

NAME

fsdb_ufs – ufs file system debugger

SYNOPSIS

fsdb –F ufs [*generic_options*] [*specific_options*] *special*

DESCRIPTION

The **fsdb_ufs** command is an interactive tool that can be used to patch up a damaged UFS file system. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an inode. These greatly simplify the process of correcting control block entries or descending the file system tree.

fsdb contains several error-checking routines to verify inode and block addresses. These can be disabled if necessary by invoking **fsdb** with the –**o** option or by the use of the **o** command.

fsdb reads a block at a time and will therefore work with raw as well as block I/O devices. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block. Note that in order to modify any portion of the disk, **fsdb** must be invoked with the **w** option.

Wherever possible, **adb-**like syntax was adopted to promote the use of **fsdb** through familiarity.

OPTIONS

The following option is supported:

–o Specify UFS file system specific options. These options can be any combination of the following separated by commas (with no intervening spaces). The options available are:

?	Display usage
o	Override some error conditions
p='string'	set prompt to string
w	open for write

USAGE

Numbers are considered hexadecimal by default. However, the user has control over how data is to be displayed or accepted. The **base** command will display or set the input/output base. Once set, all input will default to this base and all output will be shown in this base. The base can be overriden temporarily for input by preceding hexadecimal numbers with '**0x**', preceding decimal numbers with '**0t**', or octal numbers with '**0**'. Hexadecimal numbers beginning with **a-f** or **A-F** must be preceded with '**0x**' to distinguish them from commands.

Disk addressing by **fsdb** is at the byte level. However, **fsdb** offers many commands to convert a desired inode, directory entry, block, superblock etc. to a byte address. Once the address has been calculated, **fsdb** will record the result in **dot** (see next paragraph).

Several global values are maintained by **fsdb**:

- the current base (referred to as **base**),
- the current address (referred to as **dot**),
- the current inode (referred to as **inode**),

- the current count (referred to as **count**),
- and the current type (referred to as **type**).

Most commands use the preset value of **dot** in their execution. For example,

> 2:inode

will first set the value of **dot** to 2, ':', will alert the start of a command, and the **inode** command will set **inode** to 2. A count is specified after a ','. Once set, **count** will remain at this value until a new command is encountered which will then reset the value back to 1 (the default). So, if

> 2000,400/X

is typed, 400 hex longs are listed from 2000, and when completed, the value of **dot** will be **2000 + 400** ∗ **sizeof (long)**. If a carriage-return is then typed, the output routine will use the current values of **dot**, **count**, and **type** and display 400 more hex longs. A '∗' will cause the entire block to be displayed.

End of fragment, block and file are maintained by **fsdb**. When displaying data as fragments or blocks, an error message will be displayed when the end of fragment or block is reached. When displaying data using the **db**, **ib**, **directory**, or **file** commands an error message is displayed if the end of file is reached. This is mainly needed to avoid passing the end of a directory or file and getting unknown and unwanted results.

An example showing several commands and the use of carriage-return would be:

> 2:ino; 0:dir?d
or
> 2:ino; 0:db:block?d

The two examples are synonymous for getting to the first directory entry of the root of the file system. Once there, subsequent carriage-returns (or +, -) will advance to subsequent entries. Note that

> 2:inode; :ls
or
> :ls /

is again synonymous.

Expressions

The symbols recognized by **fsdb** are:

CARRIAGE-RETURN
update the value of **dot** by the current value of **type** and display using the current value of **count**.

numeric expressions may be composed of +, -, ∗, and % operators (evaluated left to right) and may use parentheses. Once evaluated, the value of **dot** is updated.

, *count* count indicator. The global value of **count** will be updated to **count**. The value of **count** will remain until a new command is run. A count specifier of '∗' will attempt to show a *blocks's* worth of information. The default for **count** is 1.

? *f* display in structured style with format specifier *f* (see **Formatted Output**

section).

/ *f* — display in unstructured style with format specifier *f* (see **Formatted Output** section).

. — the value of **dot**.

+*e* — increment the value of **dot** by the expression *e.* The amount actually incremented is dependent on the size of **type**:

dot = dot + e * sizeof (type)

The default for *e* is **1**.

-*e* — decrement the value of **dot** by the expression *e* (see +).

e* — multiply the value of **dot by the expression *e.* Multiplication and division don't use **type**. In the above calculation of **dot**, consider the **sizeof(type)** to be **1**.

%*e* — divide the value of **dot** by the expression *e* (see *).

< *name* — restore an address saved in register *name*. *name* must be a single letter or digit.

> *name* — save an address in register *name*. *name* must be a single letter or digit.

= *f* — display indicator. If *f* is a legitimate format specifier (see **Formatted Output** section), then the value of **dot** is displayed using format specifier *f*. Otherwise, assignment is assumed (see next item).

= [*s*] [*e*] — assignment indicator. The address pointed to by **dot** has its contents changed to the value of the expression *e* or to the ASCII representation of the quoted (") string *s*. This may be useful for changing directory names or ASCII file information.

=+ *e* — incremental assignment. The address pointed to by **dot** has its contents incremented by expression *e*.

=- *e* — decremental assignment. The address pointed to by **dot** has its contents decremented by expression *e*.

Commands

A command must be prefixed by a ':' character. Only enough letters of the command to uniquely distinguish it are needed. Multiple commands may be entered on one line by separating them by a space, tab or ';'.

In order to view a potentially unmounted disk in a reasonable manner, **fsdb** offers the *cd*, *pwd*, *ls* and *find* commands. The functionality of these commands substantially matches those of its UNIX counterparts (see individual command for details). The '*', '?', and '[-]' wild card characters are available.

base=b — display or set base. As stated above, all input and output is governed by the current **base**. If the '=b' is left off, the current **base** is displayed. Otherwise, the current **base** is set to *b*. Note that this is interpreted using the old value of **base**, so to ensure correctness use the '0', '0t', or '0x' prefix when changing the **base**. The default for **base** is hexadecimal.

block convert the value of **dot** to a block address.

cd *dir* change the current directory to directory *dir*. The current values of **inode** and **dot** are also updated. If no *dir* is specified, then change directories to inode **2** ("/").

cg convert the value of **dot** to a cylinder group.

directory If the current **inode** is a directory, then the value of **dot** is converted to a directory slot offset in that directory and **dot** now points to this entry.

file the value of **dot** is taken as a relative block count from the beginning of the file. The value of **dot** is updated to the first byte of this block.

find *dir* [−**name** *n*] [−**inum** *i*]
find files by name or i-number. **find** recursively searches directory **dir** and below for filenames whose i-number matches *i* or whose name matches pattern *n*. Note that only one of the two options (-name or -inum) may be used at one time. Also, the -print is not needed or accepted.

fill=*p* fill an area of disk with pattern *p*. The area of disk is delimited by **dot** and **count**.

fragment convert the value of *dot* to a fragment address. The only difference between the **fragment** command and the **block** command is the amount that is able to be displayed.

inode convert the value of *dot* to an inode address. If successful, the current value of **inode** will be updated as well as the value of *dot*. As a convenient shorthand, if ':inode' appears at the beginning of the line, the value of *dot* is set to the current **inode** and that inode is displayed in inode format.

ls [−**R**] [−**l**] *pat1 pat2* . . .
list directories or files. If no file is specified, the current directory is assumed. Either or both of the options may be used (but, if used, *must* be specified before the filename specifiers). Also, as stated above, wild card characters are available and multiple arguments may be given. The long listing shows only the i-number and the name; use the **inode** command with '?i' to get more information.

override toggle the value of override. Some error conditions may be overriden if override is toggled on.

prompt *p* change the **fsdb** prompt to *p*. *p* must be surrounded by (")s.

pwd display the current working directory.

quit quit **fsdb**.

sb the value of *dot* is taken as a cylinder group number and then converted to the address of the superblock in that cylinder group. As a shorthand, ':sb' at the beginning of a line will set the value of *dot* to *the* superblock and display it in superblock format.

! escape to shell

Inode Commands

In addition to the above commands, there are several commands that deal with inode fields and operate directly on the current **inode** (they still require the ':'). They may be used to more easily display or change the particular fields. The value of *dot* is only used by the '**:db**' and '**:ib**' commands. Upon completion of the command, the value of *dot* is changed to point to that particular field. For example,

```
> :ln=+1
```

would increment the link count of the current **inode** and set the value of *dot* to the address of the link count field.

at access time.

bs block size.

ct creation time.

db use the current value of *dot* as a direct block index, where direct blocks number from 0 - 11. In order to display the block itself, you need to 'pipe' this result into the **block** or **fragment** command. For example,

```
> 1:db:block,20/X
```

would get the contents of data block field 1 from the inode and convert it to a block address. 20 longs are then displayed in hexadecimal (see **Formatted Output** sub-section).

gid group id.

ib use the current value of *dot* as an indirect block index where indirect blocks number from 0 - 2. This will only get the indirect block itself (the block containing the pointers to the actual blocks). Use the **file** command and start at block 12 to get to the actual blocks.

ln link count.

mt modification time.

md mode.

maj major device number.

min minor device number.

nm although listed here, this command actually operates on the directory name field. Once poised at the desired directory entry (using the *directory* command), this command will allow you to change or display the directory name. For example,

```
> 7:dir:nm="foo"
```

will get the **7**th directory entry of the current **inode** and change its name to foo. Note that names cannot be made larger than the field is set up for. If an attempt is made, the string is truncated to fit and a warning message to this effect is displayed.

sz file size.

uid user id.

Formatted Output

There are two styles and many format types. The two styles are structured and unstructured. Structured output is used to display inodes, directories, superblocks and the like. Unstructured just displays raw data. The following table shows the different ways of displaying:

?

c	display as cylinder groups
i	display as inodes
d	display as directories
s	display as superblocks

/

b	display as bytes
c	display as characters
o O	display as octal shorts or longs
d D	display as decimal shorts or longs
x X	display as hexadecimal shorts or longs

The format specifier immediately follows the ’/’ or ’?’ character. The values displayed by ’/b’ and all ’?’ formats are displayed in the current **base**. Also, **type** is appropriately updated upon completion.

EXAMPLES

> 2000+400%(20+20)=D
will display **2010** in decimal (use of **fsdb** as a calculator for complex arithmetic).

> 386:ino?i display i-number **386** in an inode format. This now becomes the current **inode**.

> :ln=4 changes the link count for the current **inode** to **4**.

> :ln=+1 increments the link count by **1**.

> :ct=X display the creation time as a hexadecimal long.

> :mt=t display the modification time in time format.

> 0:file/c displays, in ASCII, block zero of the file associated with the current **inode**.

> 2:ino,*?d displays the first blocks worth of directory entries for the root inode of this file system. It will stop prematurely if the EOF is reached.

> 5:dir:inode; 0:file,*/c
changes the current inode to that associated with the **5**th directory entry (numbered from zero) of the current **inode.** The first logical block of the file is then displayed in ASCII.

> :sb displays the superblock of this file system.

> 1:cg?c displays cylinder group information and summary for cylinder group **1**.

> 2:inode; 7:dir=3 changes the i-number for the seventh directory slot in the root directory to **3**.

> 7:dir:nm="name" changes the name field in the directory slot to *name*.

> 2:db:block,*?d displays the third block of the current **inode** as directory entries.

> 3c3:fragment,20:fill=0x20
get fragment **3c3** and fill **20 type** elements with **0x20**.

> 2050=0xffff set the contents of address **2050** to **0xffffffff**. **0xffffffff** may be truncated depending on the current **type**.

> 1c92434="this is some text"
will place the ASCII for the string at **1c92434**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

clri(1M), **fsck_ufs**(1M), **dir_ufs**(4), **fs_ufs**(4) **attributes**(5)

WARNINGS

Since **fsdb** reads the disk raw, extreme caution is advised in determining its availability of **fsdb** on the system. Suggested permissions are 600 and owned by bin.

NOTES

The old command line syntax for clearing i-nodes using the ufs-specific **’-z i-number’** option is still supported by the new debugger, though it is obsolete and will be removed in a future release. Use of this flag will result in correct operation, but an error message will be printed warning of the impending obsolesence of this option to the command. The equivalent functionality is available using the more flexible **clri**(1M) command.

NAME

fsirand – install random inode generation numbers

SYNOPSIS

fsirand [**–p**] *special*

DESCRIPTION

fsirand installs random inode generation numbers on all the inodes on device *special*, and also installs a file system ID in the superblock. This helps increase the security of file systems exported by NFS.

fsirand must be used only on an unmounted file system that has been checked with **fsck**(1M)**.** The only exception is that it can be used on the root file system in single-user mode, if the system is immediately re-booted afterwards.

OPTIONS

–p Print out the generation numbers for all the inodes, but do not change the generation numbers.

USAGE

See **largefile**(5) for the description of the behavior of **fsirand** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fsck(1M), **attributes**(5), **largefile**(5)

NAME

fstyp – determine file system type

SYNOPSIS

fstyp [**–v**] *special*

DESCRIPTION

fstyp allows the user to determine the file system type of unmounted file systems using heuristic programs.

An **fstyp** module for each file system type to be checked is executed; each of these modules applies an appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks. If it is, the program prints on standard output the usual file system identifier for that type (for example, ‘‘ufs’’) and exits with a return code of **0**; if none of the modules succeed, the error message **unknown_fstyp (no matches)** is returned and the exit status is **1**. If more than one module succeeds, the error message **unknown_fstyp (multiple matches)** is returned and the exit status is **2**.

OPTIONS

–v Produce verbose output. This is usually information about the file systems superblock and varies across different *FSTypes*. See **fs_ufs**(4), **mkfs_ufs**(1M), and **tunefs**(1M) for details.

USAGE

See **largefile**(5) for the description of the behavior of **fstyp** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mkfs_ufs(1M), **tunefs**(1M), **fs_ufs**(4), **attributes**(5), **largefile**(5), **hsfs**(7FS), **pcfs**(7FS)

NOTES

The use of heuristics implies that the result of **fstyp** is not guaranteed to be accurate.

NAME

fusage – disk access profiler

SYNOPSIS

/usr/sbin/fusage [*mount_point* | *block_special_device* . . .]

DESCRIPTION

When used with no options, **fusage** reports block I/O transfers, in kilobytes, to and from all locally mounted file systems. The count data are cumulative since the time of the mount.

The report includes one section for each file system and advertised resource, and has one entry for each remotely mounted file system. Sections are ordered by device name; advertised resources that are not complete file systems will immediately follow the sections for the file systems they are in.

OPTIONS

mount_point Reports on the named file system.

block_special_device Block special device.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

crash(1M), **df**(1M), **mount**(1M), **attributes**(5)

NAME

fuser – identify processes using a file or file structure

SYNOPSIS

/usr/sbin/fuser [– [**c** | **f**] **ku**] *files* [[– [**c** | **f**] **ku**] *files*] . . .

DESCRIPTION

fuser displays the process IDs of the processes that are using the *files* specified as arguments. Each process ID is followed by a letter code, interpreted as follows: if the process is using the file as 1) its current directory, the code is **c**; 2) its root directory, the code is **r**; 3) an open file, the code is **o**; or 4) its text file, the code is **t**. For block special devices with mounted file systems, all processes using any file on that device are listed. For all types of files (text files, executables, directories, devices, and so on), only the processes using that file are reported.

If more than one group of files are specified, the options may be respecified for each additional group of files. A lone dash cancels the options currently in force.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error.

Any user with permission to read **/dev/kmem** and **/dev/mem** can use **fuser**. Only the super-user can terminate another user's process.

OPTIONS

–c Report on files that are mount points for file systems, and any files within that mounted file system.

–f Print a report for the named file, not for files within a mounted file system.

–k Send the **SIGKILL** signal to each process. Since this option spawns kills for each process, the kill messages may not show up immediately (see **kill**(2)).

–u Display the user login name in parentheses following the process ID.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ps(1), **mount**(1M), **kill**(2), **signal**(3C), **attributes**(5)

NOTES

Because **fuser** works with a snapshot of the system image, it may miss processes that begin using a file while **fuser** is running. Also, processes reported as using a file may have stopped using it while **fuser** was running. These factors should discourage the use of the –**k** option.

NAME

fwtmp, wtmpfix – manipulate connect accounting records

SYNOPSIS

/usr/lib/acct/fwtmp [–**ic**]
/usr/lib/acct/wtmpfix [*file*...]

DESCRIPTION

fwtmp reads from the standard input and writes to the standard output, converting binary records of the type found in **/var/adm/wtmp** to formatted ASCII records. The ASCII version is useful when it is necessary to edit bad records.

wtmpfix examines the standard input or named files in **utmp.h** format, corrects the time / date stamps to make the entries consistent, and writes to the standard output. A hyphen (–) can be used in place of *file* to indicate the standard input. If time / date corrections are not performed, **acctcon**(1M) will fault when it encounters certain date-change records.

Each time the date is set, a pair of date change records are written to **/var/adm/wtmp**. The first record is the old date denoted by the string "**old time**" placed in the **line** field and the flag **OLD_TIME** placed in the **type** field of the **utmp** structure. The second record specifies the new date and is denoted by the string **new time** placed in the **line** field and the flag **NEW_TIME** placed in the **type** field. **wtmpfix** uses these records to synchronize all time stamps in the file.

In addition to correcting time / date stamps, **wtmpfix** will check the validity of the **name** field to ensure that it consists solely of alphanumeric characters or spaces. If it encounters a name that is considered invalid, it will change the login name to **INVALID** and write a diagnostic to the standard error. In this way, **wtmpfix** reduces the chance that **acctcon** will fail when processing connect accounting records.

OPTIONS

–ic Denotes that input is in ASCII form, and output is to be written in binary form.

FILES

/var/adm/wtmp
/usr/include/utmp.h

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

acctcom(1), **ed**(1), **acct**(1M), **acctcms**(1M), **acctcon**(1M), **acctmerg**(1M), **acctprc**(1M), **acctsh**(1M), **runacct**(1M), **acct**(2), **acct**(4), **utmp**(4), **attributes**(5)

System Administration Guide

NAME

gencc – create a front-end to the cc command

SYNOPSIS

gencc

DESCRIPTION

The **gencc** command is an interactive command designed to aid in the creation of a front-end to the **cc** command. Since hard-coded pathnames have been eliminated from the C Compilation System (CCS), it is possible to move pieces of the CCS to new locations without recompilation. The new locations of moved pieces can be specified through the –**Y** option to the **cc** command. However, it is inconvenient to supply the proper –**Y** options with every invocation of the **cc** command. Further, if a system administrator moves pieces of the CCS, such movement should be invisible to users.

The front-end to the **cc** command that **gencc** generates is a one-line shell script that calls the **cc** command with the proper –**Y** options specified. The front-end to the **cc** command will also pass all user-supplied options to the **cc** command.

gencc prompts for the location of each tool and directory that can be respecified by a –**Y** option to the **cc** command. If no location is specified, it assumes that that piece of the CCS has not been relocated. After all the locations have been prompted for, **gencc** will create the front-end to the **cc** command.

gencc creates the front-end to the **cc** command in the current working directory and gives the file the same name as the **cc** command. Thus, **gencc** can not be run in the same directory containing the actual **cc** command. Further, if a system administrator has redistributed the CCS, the actual **cc** command should be placed in a location that is not typically in a user's path (for example, **/usr/lib**). Such placement will prevent users from accidentally invoking the **cc** command without using the front-end.

FILES

./cc	front-end to **cc**

SEE ALSO

cc(1B)

NOTES

gencc does not produce any warnings if a tool or directory does not exist at the specified location. Also, **gencc** does not actually move any files to new locations.

The **gencc** command is obsolete.

NAME

getdev – lists devices based on criteria

SYNOPSIS

/usr/bin/getdev [–**ae**] [*criteria* ...] [*device* ...]

DESCRIPTION

getdev generates a list of devices that match certain criteria. The criteria includes a list of attributes (given in expressions) and a list of devices. If no criteria are given, all devices are included in the list.

Devices must satisfy at least one of the criteria in the list unless the –**a** option is used. Then, only those devices which match all of the criteria in a list will be included.

Devices which are defined on the command line and which match the criteria are included in the generated list. However, if the –**e** option is used, the list becomes a set of devices to be *excluded* from the list.

criteria defines the criteria that a device must match to be included in the generated list. Should be specified by giving expressions.

device defines the devices which should be included in the generated list. This can be the pathname of the device or the device alias.

Criteria Expression Types

There are four possible expression types which the criteria specified in the *criteria* argument may follow:

attribute=*value* Selects all devices whose attribute *attribute* is defined and is equal to *value.*

attribute!=*value* Selects all devices whose attribute *attribute* is defined and does not equal *value.*

attribute:* Selects all devices which have the attribute *attribute* defined.

attribute!:* Selects all devices which do not have the attribute *attribute* defined.

See the **putdev**(1M) manual page for a complete listing and description of available attributes.

OPTIONS

–**a** Specifies that a device must match all criteria to be included in the list generated by this command. The option has no effect if no criteria are defined.

–**e** Specifies that the list of devices which follows on the command line should be *excluded* from the list generated by this command. (Without the –**e** the named devices are *included* in the generated list.) The flag has no effect if no devices are defined.

ERRORS

The command will exit with one of the following values:

0 = Successful completion of the task.

1 = Command syntax incorrect, invalid option used, or internal error occurred.

2 = Device table could not be opened for reading.

FILES

/etc/device.tab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

devattr(1M), **getdgrp**(1M), **putdev**(1M), **putdgrp**(1M), **attributes**(5)

NAME

getdgrp – lists device groups which contain devices that match criteria

SYNOPSIS

/usr/sbin/getdgrp [–ael] [*criteria* . . .**] [***dgroup* . . .**]**

DESCRIPTION

getdgrp generates a list of device groups that contain devices matching the given criteria. The criteria is given in the form of expressions.

criteria can be one expression or a list of expressions which a device must meet for its group to be included in the list generated by **getdgrp**. If no criteria are given, all device groups are included in the list.

Devices must satisfy at least one of the criteria in the list. However, the –**a** option can be used to define that a "logical and" operation should be performed. Then, only those groups containing devices which match all of the criteria in a list will be included.

dgroup defines a set of device groups to be included in the list. Device groups that are defined and which contain devices matching the criteria are included. However, if the –**e** option is used, this list defines a set of device groups to be excluded. When the –**e** option is used and *criteria* is also defined, the generated list will include device groups containing devices which match the criteria and are not in the command line list.

criteria defines criteria that a device must match before a device group to which it belongs can be included in the generated list.

dgroup defines device groups which should be included in or excluded from the generated list.

Criteria Expression Types

There are four possible expressions types which the criteria specified in the *criteria* argument may follow:

*attribute***=***value* Selects all device groups with a member whose attribute *attribute* is defined and is equal to *value.*

*attribute***!=***value* Selects all device groups with a member whose attribute *attribute* is defined and does not equal *value.*

*attribute***:*** Selects all device groups with a member which has the attribute *attribute* defined.

*attribute***!:*** Selects all device groups with a member which does not have the attribute *attribute* defined.

See **putdev**(1M) for a complete listing and description of available attributes.

OPTIONS

–**a** Specifies that a device must match all criteria to be included in the list generated by this command. The option has no effect if no criteria are defined.

–**e** Specifies that the list of device groups on the command line should be *excluded* from the list generated by this command. (Without the –**e** option the named device groups are *included* in the generated list.) The flag has no effect if no devices are defined.

–l Specifies that all device groups (subject to the **–e** option and the *dgroup* list) should be listed even if they contain no valid device members. This option has no affect if *criteria* is specified on the command line.

ERRORS

The command will exit with one of the following values:

0 Successful completion of the task.

1 Command syntax incorrect, invalid option used, or internal error occurred.

2 Device table or device group table could not be opened for reading.

FILES

/etc/device.tab
/etc/dgroup.tab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

devattr(1M), **getdev**(1M), **putdev**(1M), **putdgrp**(1M), **attributes**(5)

NAME

getent – get entries from administrative database

SYNOPSIS

getent *database* [*key* ...]

DESCRIPTION

getent gets a list of entries from the administrative database specified by *database*. The information comes from one or more of the sources that are specified for the *database* in **/etc/nsswitch.conf.**

database is the name of the database to be examined. This can be **passwd**, **group**, **hosts**, **services**, **protocols**, **ethers**, **networks**, or **netmasks**. For each of these databases, **getent** uses the appropriate library routines described in **getpwnam**(3C), **getgrnam**(3C), **gethostbyname**(3N), **getservbyname**(3N), **getprotobyname**(3N), **ethers**(3N), and **getnetbyname**(3N), respectively.

Each *key* must be in a format appropriate for searching on the respective database. For example, it can be a *username* or *numeric-uid* for **passwd**; *hostname* or *IP address* for **hosts**; or *service*, *service/protocol*, *port*, or *port/proto* for **services**.

getent prints out the database entries that match each of the supplied keys, one per line, in the format of the matching administrative file: **passwd**(4), **group**(4), **hosts**(4), **services**(4), **protocols**(4), **ethers**(3N), **networks**(4), or **netmasks**(4). If no key is given, all entries returned by the corresponding enumeration library routine (for example, **getpwent** or **gethostent**) are printed.

ERRORS

getent exits with one of the following values:

0 Successful completion of the task.

1 Command syntax incorrect, invalid option used, or internal error occurred.

2 At least one of the specified entry names was not found in the database.

3 There is no support for enumeration on this database.

FILES

/etc/nsswitch.conf	name service switch configuration file
/etc/passwd	password file
/etc/group	group file
/etc/hosts	host name database
/etc/services	Internet services and aliases
/etc/protocols	protocol name database
/etc/ethers	Ethernet address to hostname database or domain
/etc/networks	network name database
/etc/netmasks	network mask database

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ethers(3N), **getgrnam**(3C), **gethostbyname**(3N), **gethostent**(3N), **getnetbyname**(3N), **getprotobyname**(3N), **getpwnam**(3C), **getservbyname**(3N), **group**(4), **hosts**(4), **netmasks**(4), **networks**(4), **nsswitch.conf**(4), **passwd**(4), **protocols**(4), **services**(4), **attributes**(5)

NAME

gettable – get DoD Internet format host table from a host

SYNOPSIS

/usr/sbin/gettable *host*

DESCRIPTION

gettable is a simple program used to obtain the DoD Internet host table from a “hostname” server. The specified *host* is queried for the table. The table is placed in the file **hosts.txt**.

gettable operates by opening a TCP connection to the port indicated in the service specification for “hostname”. A request is then made for all names and the resultant information is placed in the output file.

gettable is best used in conjunction with the **htable**(1M) program which converts the DoD Internet host table format to that used by the network library lookup routines.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

htable(1M), **attributes**(5)
Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *HOSTNAME Server,* RFC 953, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

NOTES

Should allow requests for only part of the database.

NAME

getty – set terminal type, modes, speed, and line discipline

SYNOPSIS

/usr/lib/saf/ttymon [–**h**] [–**t** *timeout*] *line* [*speed* [*type* [*linedisc*]]]

/usr/lib/saf/ttymon –**c** *file*

DESCRIPTION

getty is a symbolic link to **/usr/lib/saf/ttymon**. It is included for compatibility with previous releases for the few applications that still call **getty** directly. **getty** can only be executed by the super-user, (a process with the user ID **root**). Initially **getty** prints the login prompt, waits for the user's login name, and then invokes the **login** command. **getty** attempts to adapt the system to the terminal speed by using the options and arguments specified on the command line.

When given no optional arguments, **getty** specifies the following: The *speed* of the interface is set to 300 baud, either parity is allowed, new-line characters are converted to carriage return-line feed, and tab expansion is performed on the standard output. **getty** types the login prompt before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pressing the BREAK key. This will cause **getty** to attempt the next *speed* in the series. The series that **getty** tries is determined by what it finds in **/etc/ttydefs**.

OPTIONS

line
The name of a TTY line in **/dev** to which **getty** is to attach itself. **getty** uses this string as the name of a file in the **/dev** directory to open for reading and writing.

–**h**
If the –**h** flag is not set, a hangup will be forced by setting the speed to zero before setting the speed to the default or a specified speed.

–**t** *timeout*
Specifies that **getty** should exit if the open on the line succeeds and no one types anything in *timeout* seconds.

speed
The *speed* argument is a label to a speed and TTY definition in the file **/etc/ttydefs**. This definition tells **getty** at what speed to run initially, what the initial TTY settings are, and what speed to try next, (should the user press the BREAK key to indicate that the speed is inappropriate). The default *speed* is 300 baud.

type and *linedisc*
These options are obsolete and will be ignored.

–**c** *file*
The –**c** option is no longer supported. Instead use **/usr/sbin/sttydefs –l** to list the contents of the **/etc/ttydefs** file and perform a validity check on the file.

FILES

/etc/ttydefs

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

ct(1C), **login**(1), **sttydefs**(1M), **ttymon**(1M), **ioctl**(2), **attributes**(5), **tty**(7D)

NAME

getvol – verifies device accessibility

SYNOPSIS

/usr/bin/getvol –n [**–l** *label*] *device*
/usr/bin/getvol [**–f** | **–F**] [**–ow**] [**–l** *label* | **–x** *label*] *device*

DESCRIPTION

getvol verifies that the specified device is accessible and that a volume of the appropriate medium has been inserted. The command is interactive and displays instructional prompts, describes errors, and shows required label information.

device names the device which should be verified for accessibility.

OPTIONS

–n Runs the command in non-interactive mode. The volume is assumed to be inserted upon command invocation.

–l *label* Specifies that the label *label* must exist on the inserted volume (can be overridden by the –o option).

–f Formats the volume after insertion, using the format command defined for this device in the device table.

–F Formats the volume after insertion and places a file system on the device. Also uses the format command defined for this device in the device table.

–o Allows the administrator to override a label check.

–w Allows administrator to write a new label on the device. User is prompted to supply the label text. This option is ineffective if the **–n** option is enabled.

–x *label* Specifies that the label *label* must exist on the device. This option should be used in place of the **–l** option when the label can only be verified by visual means. Use of the option causes a message to be displayed asking the administrator to visually verify that the label is indeed *label*.

ERRORS

The command will exit with one of the following values:

0 Successful completion of the task.

1 Command syntax incorrect, invalid option used, or internal error occurred.

3 Device table could not be opened for reading.

FILES

/etc/device.tab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5)

NOTES

This command uses the device table to determine the characteristics of the device when performing the volume label checking.

NAME

groupadd – add (create) a new group definition on the system

SYNOPSIS

/usr/sbin/groupadd [−**g** *gid* [−**o**]] *group*

DESCRIPTION

The **groupadd** command creates a new group definition on the system by adding the appropriate entry to the **/etc/group** file.

OPTIONS

−**g** *gid* The group id for the new group. This group id must be a non-negative decimal integer below **MAXUID** as defined in the <**param.h**> header. The group ID defaults to the next available (unique) number above the highest number currently assigned. For example, if groups 100, 105, and 200 are assigned as groups, the next default group number will be 201. (Group IDs from 0-99 are reserved by SunOS for future applications.)

−**o** This option allows the *gid* to be duplicated (non-unique).

group A string of no more than eight bytes consisting of characters from the set of lower case alphabetic characters and numeric characters. A warning message will be written if these restrictions are not met. A future Solaris release may refuse to accept group fields that do not meet these requirements. The group field must contain at least one character and must not contain a colon (**:**) or newline (**\n**).

ERRORS

The **groupadd** command exits with one of the following values:

0 Success.

2 Invalid command syntax. A usage message for the **groupadd** command is displayed.

3 An invalid argument was provided to an option.

4 *gid* is not unique (when −**o** option is not used).

9 *group* is not unique.

10 Cannot update the **/etc/group** file.

FILES

/etc/group
/usr/include/userdefs.h

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

users(1B), **groupdel**(1M), **groupmod**(1M), **grpck**(1M), **logins**(1M), **pwck**(1M), **useradd**(1M), **userdel**(1M), **usermod**(1M), **group**(4), **attributes**(5)

NOTES

groupadd only adds a group definition to the local system. If a network name service such as NIS or NIS+ is being used to supplement the local **/etc/group** file with additional entries, **groupadd** cannot change information supplied by the network name service.

However **groupadd** will verify the uniqueness of group name and group ID against the external name service.

NAME

groupdel – delete a group definition from the system

SYNOPSIS

/usr/sbin/groupdel *group*

DESCRIPTION

The **groupdel** command deletes a group definition from the system. It deletes the appropriate entry from the **/etc/group** file.

OPTIONS

group A string of printable characters that specifies the group to be deleted.

FILES

/etc/group

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

users(1B), **groupadd**(1M), **groupmod**(1M), **logins**(1M), **useradd**(1M), **userdel**(1M), **usermod**(1M), **attributes**(5)

DIAGNOSTICS

The **groupdel** command exits with one of the following values:

0 Success.

2 Invalid command syntax. A usage message for the **groupdel** command is displayed.

6 *group* does not exist.

10 Cannot update the **/etc/group** file.

NOTES

groupdel only deletes a group definition that is in the local **/etc/group** file. If a network nameservice such as NIS or NIS+ is being used to supplement the local **/etc/group** file with additional entries, **groupdel** cannot change information supplied by the network nameservice.

NAME

groupmod – modify a group definition on the system

SYNOPSIS

/usr/sbin/groupmod [**–g** *gid* [**–o**]] [**–n** *name*] *group*

DESCRIPTION

The **groupmod** command modifies the definition of the specified group by modifying the appropriate entry in the **/etc/group** file.

OPTIONS

–g *gid* The group ID for the new group. This group ID must be a non-negative decimal integer below **MAXUID** as defined in **<param.h>**. The group ID defaults to the next available (unique) number above 99. (Group IDs from 0-99 are reserved by SunOS for future applications.)

–o This option allows the *gid* to be duplicated (non-unique).

–n *name* A string of printable characters that specifies a new name for the group. It may not include a colon (**:**) or NEWLINE (**\n**).

OPERANDS

group The current name of the group to be modified.

EXIT STATUS

The **groupmod** command exits with one of the following values:

0 Success.

2 Invalid command syntax. A usage message for the **groupmod** command is displayed.

3 An invalid argument was provided to an option.

4 *gid* is not unique (when the **–o** option is not used).

6 *group* does not exist.

9 *name* already exists as a group name.

10 Cannot update the **/etc/group** file.

FILES

/etc/group group file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

users(1B), **groupadd**(1M), **groupdel**(1M), **logins**(1M), **useradd**(1M), **userdel**(1M), **usermod**(1M), **group**(4), **attributes**(5)

NOTES

groupmod only modifies group definitions in the **/etc/group** file. If a network name service such as NIS or NIS+ is being used to supplement the local **/etc/group** file with additional entries, **groupmod** cannot change information supplied by the network name service. However **groupmod** will verify the uniqueness of group name and group id against the external name service.

NAME

halt, poweroff – stop the processor

SYNOPSIS

/usr/sbin/halt [–**lnqy**]

/usr/sbin/poweroff [–**lnqy**]

DESCRIPTION

halt and **poweroff** write out any pending information to the disks and then stop the processor. **poweroff** will have the machine remove power, if possible.

halt and **poweroff** normally log the system shutdown to the system log daemon, **syslogd**(1M), and place a shutdown record in the login accounting file **/var/adm/wtmp**. These actions are inhibited if the –**n** or –**q** options are present.

OPTIONS

–**l** Suppress sending a message to the system log daemon, **syslogd**(1M), about who executed **halt**.

–**n** Prevent the **sync**(1M) before stopping.

–**q** Quick halt. No graceful shutdown is attempted.

–**y** Halt the system, even from a dialup terminal.

FILES

/var/adm/wtmp	login accounting file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

init(1M), **reboot**(1M), **shutdown**(1M), **sync**(1M), **syslogd**(1M), **attributes**(5)

NOTES

halt does not execute the **rc0** scripts as do **shutdown**(1M) and **init**(1M).

poweroff is equivalent to **init 5**.

NAME

hostconfig – configure a system’s host parameters

SYNOPSIS

/usr/bin/hostconfig **–p** *protocol* [**–d**] [**–n**] [**–v**] [**–i** *interface*] [**–f** *hostname*]

DESCRIPTION

The **hostconfig** program uses a network protocol to acquire a machine’s “host parameters” and then sets these parameters on the system. The program selects which protocol to use based on the argument to the required **–p** flag. Different protocols may set different host parameters. Currently, two protocols are defined but only one protocol is supported.

OPTIONS

–p bootparams Use the “whoami” call of the RPC “bootparams” protocol. This sets the system’s *hostname*, *domainname*, and *default IP router* parameters.

–p bootp Use the **BOOTP** protocol (not currently supported).

–d Enable “debug” output.

–n Run the network protocol, but do not set the acquired parameters into the system.

–v Enable verbose output.

–i *interface* Use only the named network interface to run the protocol.

–f *hostname* Run the protocol as if this machine were named *hostname*.

EXAMPLES

To configure a machine’s host parameters using the “bootparams whoami” protocol with a verbose output, use

```
example% hostconfig –p bootparams –v
```

To see what parameters would be set using the “bootparams whoami” protocol, use

```
example% hostconfig –p bootparams –n –v
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

hostname(1), **domainname**(1M), **route**(1M), **attributes**(5)

NAME

htable – convert DoD Internet format host table

SYNOPSIS

/usr/sbin/htable *filename*

DESCRIPTION

htable converts a host table in the format specified by RFC 952 to the format used by the network library routines. Three files are created as a result of running **htable**: **hosts**, **networks**, and **gateways**. The **hosts** file is used by the **gethostbyname**(3N) routines in mapping host names to addresses. The **networks** file is used by the **getnetbyname**(3N) routines in mapping network names to numbers. The **gateways** file is used by the routing daemon to identify "passive" Internet gateways.

If any of the files **localhosts**, **localnetworks**, or **localgateways** are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

htable is best used in conjunction with the **gettable**(1M) program which retrieves the DoD Internet host table from a host.

FILES

localhosts
localnetworks
localgateways

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

gettable(1M), **gethostbyname**(3N), **getnetbyname**(3N), **attributes**(5)
Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *DoD Internet Host Table Specification*, RFC 952, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

NOTES

Does not properly calculate the **gateways** file.

NAME

id – return user identity

SYNOPSIS

/usr/bin/id [*user*]
/usr/bin/id **–a** [*user*]

/usr/xpg4/bin/id [*user*]
/usr/xpg4/bin/id **–G** [**–n**] [*user*]
/usr/xpg4/bin/id **–g** [**–nr**] [*user*]
/usr/xpg4/bin/id **–u** [**–nr**] [*user*]

DESCRIPTION

If no *user* operand is provided, the **id** utility will write the user and group IDs and the corresponding user and group names of the invoking process to standard output. If the effective and real IDs do not match, both will be written. If multiple groups are supported by the underlying system, the supplementary group affiliations of the invoking process also will be written.

If a *user* operand is provided and the process has the appropriate privileges, the user and group IDs of the selected user will be written. In this case, effective IDs will be assumed to be identical to real IDs. If the selected user has more than one allowable group membership listed in the group database, these will be written in the same manner as the supplementary groups described in the preceding paragraph.

Formats

The following formats will be used when the **LC_MESSAGES** locale category specifies the "C" locale. In other locales, the strings **uid**, **gid**, **euid**, **egid**, and **groups** may be replaced with more appropriate strings corresponding to the locale.

> "**uid=%u(%s) gid=%u(%s)\n**" *<real user ID>*, *<user-name>*, *<real group ID>*, *<group-name>*

If the effective and real user IDs do not match, the following will be inserted immediately before the **\n** character in the previous format:

> " **euid=%u(%s)**"

with the following arguments added at the end of the argument list:

> *<effective user ID>*, *<effective user-name>*

If the effective and real group IDs do not match, the following will be inserted directly before the **\n** character in the format string (and after any addition resulting from the effective and real user IDs not matching):

> " **egid=%u(%s)**"

with the following arguments added at the end of the argument list:

> *<effective group-ID>*, *<effective group name>*

If the process has supplementary group affiliations or the selected user is allowed to belong to multiple groups, the first will be added directly before the NEWLINE character in the format string:

" groups=%u(%s)"

with the following arguments added at the end of the argument list:

<supplementary group ID>, *<supplementary group name>*

and the necessary number of the following added after that for any remaining supplementary group IDs:

",%u(%s)"

and the necessary number of the following arguments added at the end of the argument list:

<supplementary group ID>, *<supplementary group name>*

If any of the user ID, group ID, effective user ID, effective group ID or supplementary / multiple group IDs cannot be mapped by the system into printable user or group names, the corresponding (**%s**) and name argument will be omitted from the corresponding format string.

When any of the options are specified, the output format will be as described under **OPTIONS**.

OPTIONS

/usr/bin/id

The following option is supported for **/usr/bin/id** only:

–a Reports user name, user ID and all the groups to which the user belongs.

/usr/xpg4/bin/id

The following options are supported for **/usr/xpg4/bin/id** only:

–G Output all different group IDs (effective, real and supplementary) only, using the format "**%u\n**". If there is more than one distinct group affiliation, output each such affiliation, using the format " **%u**", before the newline character is output.

–g Output only the effective group ID, using the format "**%u\n**".

–n Output the name in the format "**%s**" instead of the numeric ID using the format "**%u**".

–r Output the real ID instead of the effective ID.

–u Output only the effective user ID, using the format "**%u\n**".

OPERANDS

The following operand is supported:

user The user (login) name for which information is to be written.

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **id**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

/usr/bin/id

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu SUNWcar

/usr/xpg4/bin/id

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4

SEE ALSO

fold(1), **logname**(1), **who**(1), **getgid**(2), **getgroups**(2), **getuid**(2), **attributes**(5), **environ**(5), **xpg4**(5)

NOTES

Output produced by the −**G** option and by the default case could potentially produce very long lines on systems that support large numbers of supplementary groups.

NAME

ifconfig – configure network interface parameters

SYNOPSIS

/sbin/ifconfig *interface* [*address_family*] [*address* [*dest_address*]] [**up**] [**down**]
[**auto-revarp**] [**netmask** *mask*] [**broadcast** *address*] [**metric** *n*] [**mtu** *n*]
[**trailers** | **–trailers**] [**private** | **–private**] [**arp** | **–arp**] [**plumb**] [**unplumb**]

/usr/sbin/ifconfig *interface* [*address_family*] [*address* [*dest_address*]] [**up**] [**down**]
[**auto-revarp**] [**netmask** *mask*] [**broadcast** *address*] [**metric** *n*] [**mtu** *n*]
[**trailers** | **–trailers**] [**private** | **–private**] [**arp** | **–arp**] [**plumb**] [**unplumb**]

/sbin/ifconfig *interface* { **auto-dhcp** | **dhcp** } [**primary**] [**wait** *seconds*]
drop | **extend** | **ping** | **release** | **start** | **status**

/usr/sbin/ifconfig *interface* { **auto-dhcp** | **dhcp** } [**primary**] [**wait** *seconds*]
drop | **extend** | **ping** | **release** | **start** | **status**

DESCRIPTION

The command **ifconfig** is used to assign an address to a network interface or to configure network interface parameters, or both. **ifconfig** must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. If no option is specified, **ifconfig** displays the current configuration for a network interface. If an address family is specified, **ifconfig** reports only the details specific to that address family. Only the super-user may modify the configuration of a network interface. Options appearing within braces (**{ }**) indicate that one of the options must be specified.

The two versions of **ifconfig**, **/sbin/ifconfig** and **/usr/sbin/ifconfig**, behave differently with respect to name services. The order in which names are looked up by **/sbin/ifconfig** when the system is booting is fixed and cannot be changed. In contrast, changing **/etc/nsswitch.conf** may affect the behavior of **/usr/sbin/ifconfig**. The system administrator may configure the source and lookup order in the tables via the name service switch. See **nsswitch.conf**(4) for more information.

DHCP Configuration

The third and fourth forms of this command are used to control DHCP (Dynamic Host Configuration Protocol) configuring of the interface. DHCP is only available on interfaces whose address family is **inet**. In this mode, **ifconfig** is used to control operation of **dhcpagent**(1M), the DHCP client daemon. Once an interface is placed under DHCP control (by using the **start** operand), **ifconfig** should not, in normal operation, be used to modify the address or characteristics of the interface. If the address of an interface under DHCP is changed, the agent will implicitly drop the interface from its control, although this will not occur until **dhcpagent** wakes up to conduct another DHCP operation on the interface.

OPTIONS

The following options are supported:

arp Enable the use of the Address Resolution Protocol (ARP) in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between TCP/IP addresses and 10Mb/s Ethernet addresses.

–arp Disable the use of the Address Resolution Protocol ARP.

auto-dhcp Use the Dynamic Host Configuration Protocol (DHCP) to automatically acquire an address for this interface. This option has a completely equivalent alias called **dhcp**.

primary Defines the interface as the "primary". The interface is defined as the preferred one for the delivery of client-wide configuration data. See **dhcpagent**(1M) and **dhcpinfo**(1) for details.

Only one interface can be the primary at any given time. If another interface is subsequently selected as the primary, it replaces the previous one. Nominating an interface as the primary one will not have much significance once the client work station has booted, as many applications will already have started and been configured with data read from the previous primary interface.

wait *seconds*

ifconfig will wait until the operation either completes or for the interval specified, whichever is the sooner. If no wait interval is given, and the operation is one that cannot complete immediately, **ifconfig** will exit immediately but the requested operation will continue. The exit status of **ifconfig** in this case will indicate merely the validity of the request, not whether that request was actually successful. The symbolic value **forever** may be used in place of a numeric, with obvious meaning.

drop The specified interface will be removed from the control of **dhcpagent**.

extend **extend dhcpagent** will try to extend the lease on the interface's IP address. This is not required, as the agent will automatically extend the lease well before it expires.

ping Checks whether the interface given is under DHCP control. An exit status of **0** means yes.

release The IP address on the interface is relinquished, and the interface marked as "down".

start DHCP will be started on the interface.

status Display the DHCP configuration status of the interface.

auto-revarp Use the Reverse Address Resolution Protocol (RARP) to automatically acquire an address for this interface.

broadcast *address* (**inet** only.) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's. A "**+**" (plus sign) given for the broadcast value causes the broadcast address to be reset to a default appropriate for the (possibly new) address and netmask. **Note**: The arguments of **ifconfig** are interpreted left to right. Therefore

ifconfig –a netmask + broadcast +

and

ifconfig –a broadcast + netmask +

may result in different values being assigned for the broadcast addresses of the interfaces.

dhcp This option is an alias for option **auto-dhcp** (see above).

down Mark an interface "down". When an interface is marked "down", the system does not attempt to transmit messages through that interface. If possible, the interface is reset to disable reception as well. This action does not automatically disable routes using the interface.

metric *n* Set the routing metric of the interface to *n*; if no value is specified, the default is **0**. The routing metric is used by the routing protocol. Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

mtu *n* Set the maximum transmission unit of the interface to *n*. For many types of networks, the mtu has an upper limit, for example, **1500** for Ethernet.

netmask *mask* (**inet** only.) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion. The mask can be specified in one of four ways:

a) with a single hexadecimal number with a leading 0x,
b) with a dot-notation address,
c) with a "**+**" (plus sign) address, or
d) with a pseudo host name / pseudo network name found in the network database **networks**(4).

If a "**+**" (plus sign) is given for the netmask value, the mask is looked up in the **netmasks**(4) database. This lookup finds the longest matching netmask in the database by starting with the interface's IP address as the key and iteratively masking off more and more low order bits of the address. This iterative lookup ensures that the **netmasks**(4) database can be used to specify the netmasks when variable length subnetmasks are used within a network number.

If a pseudo host name / pseudo network name is supplied as the netmask value, netmask data may be located in the **hosts** or **networks** database. Names are looked up by first using **gethostbyname**(3N). If not found there, the names are looked up in **getnetbyname**(3N). These interfaces may in turn use **nsswitch.conf**(4) to determine what data store(s) to use to fetch the actual value.

plumb Open the device associated with the physical interface name and set up the streams needed for TCP/IP to use the device. Before this is done, the interface will not show up in the output of **ifconfig –a**.

unplumb Destroy any streams associated with this device and close the device. After this command is executed, the device name should not show up in the output of **ifconfig –a**.

private Tells the **in.routed** routing daemon that the interface should not be advertised.

–private Specify unadvertised interfaces.

trailers This flag previously caused a non-standard encapsulation of **inet** packets on certain link levels. Drivers supplied with this release no longer use this flag. It is provided for compatibility, but is ignored.

–trailers Disable the use of a "trailer" link level encapsulation.

up Mark an interface "up". This happens automatically when setting the first address on an interface. The **up** option enables an interface after an **ifconfig down**, which reinitializes the hardware.

OPERANDS

The *interface* operand, as well as address parameters that affect it, are described below.

interface A string of the form, *name physical-unit*, for example, **le0** or **ie1**; or of the form *name physical-unit*:*logical-unit*, for example, **le0:1**. Five special interface names, **–a**, **–ad**, **–au**, **–adD**, and **–auD**, are reserved and refer to all or a subset of the interfaces in the system. If one of these interface names is given, the commands following it are applied to all of the interfaces that match:

- **–a** Apply the commands to all interfaces in the system.
- **–ad** Apply the commands to all "down" interfaces in the system.
- **–adD** Like **–ad**, but only apply the commands if the interface is not under DHCP (Dynamic Host Configuration Protocol) control.
- **–au** Apply the commands to all "up" interfaces in the system.
- **–auD** Like **–au**, but only apply the commands if the interface is not under DHCP control.

address_family Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, the parameters and addresses are interpreted according to the rules of some address family, specified by the *address_family* parameter. The address families currently supported are **ether** and **inet**. If no address family is specified, **inet** is assumed.

address For the TCP/IP family (**inet**), the *address* is either a host name present in the host name data base (see **hosts**(4)) or in the Network Information Service (NIS) map **hosts**, or a TCP / IP address expressed in the Internet standard "dot notation". Typically, an Internet address specified in dot notation consists of your system's network number and the machine's unique host number. A typical Internet address is **192.9.200.44**, where **192.9.200** is the network number and **44** is the machine's host number.

For the **ether** address family, the address is an Ethernet address represented as *x*:*x*:*x*:*x*:*x*:*x* where *x* is a hexadecimal number between **0** and **FF**.

Some, though not all, of the Ethernet interface cards have their own addresses. To use cards that do not have their own addresses, refer to section 3.2.3(4) of the IEEE 802.3 specification for a definition of the locally administered address space. The use of interface groups should be restricted to those cards with their own addresses (see **INTERFACE GROUPS**).

dest_address If the *dest_address* parameter is supplied in addition to the *address* parameter, it specifies the address of the correspondent on the other end of a point-to-point link.

LOGICAL INTERFACES

Solaris TCP / IP allows multiple logical interfaces to be associated with a physical network interface. This allows a single machine to be assigned multiple IP addresses, even though it may have only one network interface. Physical network interfaces have names of the form *driver-name physical-unit-number*, while logical interfaces have names of the form *driver-name physical-unit-number*:*logical-unit-number*. A physical interface is configured into the system using the **plumb** sub-command. For example:

```
ifconfig le0 plumb
```

Logical interfaces do not need to be "plumbed". Once a physical interface has been "plumbed", logical interfaces associated with the physical interface can be configured by naming them in subsequent **ifconfig** commands. However, only root can create or delete a logical interface. For example, when executed by root the command:

```
ifconfig le0:1
```

allocates a logical interface associated with the physical interface **le0** and reports its status. When executed by a non-privileged user, **ifconfig** will report the status of the interface if it already exists, or give an error message if it does not exist.

A logical interface can be configured with parameters (**address**, **netmask**, and so on) different from the physical interface with which it is associated. Logical interfaces that are associated with the same physical interface can be given different parameters as well. Each logical interface must be associated with a physical interface. So, for example, the logical interface **le0:1** can only be configured after the physical interface **le0** has been plumbed.

To delete a logical interface, simply name the interface specifying an address of **0**, after ensuring that the interface has been marked as "down". For example, the command:

```
ifconfig le0:1 0 down
```

will delete the logical interface **le0:1**.

INTERFACE GROUPS

If an interface (logical or physical) shares an IP prefix with another interface, these interfaces are collected into an *interface group*. IP uses an interface group to rotate source address selection when the source address is unspecified, and in the case of multiple physical interfaces in the same group, to scatter traffic across different IP addresses on a per-IP-destination basis. (See **netstat**(1M) for per-IP-destination information.)

This feature may be disabled by using **ndd**(1M).

EXAMPLES

If your workstation is not attached to an Ethernet, the **le0** interface should be marked "down" as follows:

```
example% ifconfig le0 down
```

To print out the addressing information for each interface, use the following command:

```
example% ifconfig –a
```

To reset each interface's broadcast address after the netmasks have been correctly set, use the next command:

```
example% ifconfig –a broadcast +
```

To change the Ethernet address for interface **le0**, use the following command:

```
example% ifconfig le0 ether aa:1:2:3:4:5
```

FILES

/etc/netmasks	netmask data

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

/usr/sbin

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

/sbin

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

dhcpinfo(1), **dhcpagent**(1M), **in.routed**(1M), **ndd**(1M), **netstat**(1M), **ethers**(3N), **gethostbyname**(3N), **getnetbyname**(3N), **hosts**(4), **netmasks**(4), **networks**(4), **nsswitch.conf**(4), **attributes**(5), **arp**(7P)

DIAGNOSTICS

ifconfig sends messages that indicate if:

- the specified interface does not exist
- the requested address is unknown
- the user is not privileged and tried to alter an interface's configuration

NOTES

It is recommended that the names **broadcast**, **down**, **private**, **trailers**, **up**, and the other possible option names not be selected when choosing host names. Choosing any one of these names as host names will cause bizarre problems that can be extremely difficult to diagnose.

NAME

in.comsat, comsat – biff server

SYNOPSIS

/usr/sbin/in.comsat

DESCRIPTION

comsat is the server process which listens for reports of incoming mail and notifies users who have requested to be told when mail arrives. It is invoked as needed by **inetd**(1M), and times out if inactive for a few minutes.

comsat listens on a datagram port associated with the **biff** service specification (see **services**(4)) for one line messages of the form

*user***@***mailbox***-***offset*

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a **biff y**), the *offset* is used as a seek offset into the appropriate mailbox file, and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the **From**, **To**, **Date**, or **Subject** lines are not printed when displaying the message.

FILES

/var/adm/utmp to find out who is logged in and on what terminals

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M), **services**(4), **attributes**(5)

NOTES

The message header filtering is prone to error.

NAME

in.dhcpd – Dynamic Host Configuration Protocol server

SYNOPSIS

/usr/lib/inet/in.dhcpd [**−denv**] [**−b automatic | manual**] [**−h** *relay_hops*]
[**−i** *interface, . . .*] [**−o** *DHCP_offer_Time_to_Live*] [**−t** *dhcptab_rescan_interval*]

/usr/lib/inet/in.dhcpd **−r** *IP_address* | *hostname, . . .* [**−dv**] [**−h** *relay_hops*]
[**−i** *interface, . . .*]

DESCRIPTION

in.dhcpd is a daemon that responds to Dynamic Host Configuration Protocol (DHCP) requests and optionally to BOOTP protocol requests. The daemon forks a copy of itself that runs as a background process. It must be run as root. The daemon has two run modes, DHCP server (with optional BOOTP compatibility mode) and BOOTP relay agent mode. The first synopsis illustrates the options available in the DHCP / BOOTP server mode. The second synopsis illustrates the options available when the daemon is run in BOOTP relay agent mode.

The DHCP and BOOTP protocols are used to provide configuration parameters to Internet hosts. Client machines are allocated their IP addresses as well as other host configuration parameters through this mechanism.

The DHCP / BOOTP server manages two types of databases:

dhcptab database (see **dhcptab**(4))
dhcp network databases (see **dhcp_network**(4))

The **dhcptab** database contains macro definitions defined using a **termcap**-like syntax which permits network administrators to define groups of DHCP configuration parameters to be returned to clients. A DHCP / BOOTP server will return hostname, network broadcast address, network subnet mask, or IP maximum transfer unit (MTU) if requested by a client attached to the same network as the server without having to be explicitly configured in the **dhcptab**.

The **dhcp network** databases contain client identifier to IP address mappings. These databases are named after the network they support. For example, **10_0_0_0** is the **dhcp network** database for the **10.0.0.0** network.

The **dhcp network** databases are consulted during runtime. A client request received from a network for which no **dhcp network** database exists is ignored.

Multiple DHCP servers on the same network operate much more efficiently if they share DHCP databases through NIS+ or NFS. Sharing allows DHCP servers to communicate through a common datastore, increasing redundancy and balancing load among cooperating servers.

The **hosts** database is consulted if the clients request their hostname. See **hosts**(4) and **nsswitch.conf**(4) for more details.

This command may change in future releases of Solaris software. Scripts, programs, or procedures that use this command might need modification when upgrading to future Solaris software releases.

OPTIONS

–b automatic | manual
This option enables BOOTP compatibility mode, allowing the DHCP server to respond to BOOTP clients. The option argument specifies whether the DHCP server should automatically allocate permanent lease IP addresses to requesting BOOTP clients if the clients are not registered in the server's database (**automatic**) or respond only to BOOTP clients who have been manually registered in the server's databases (**manual**). This option only affects DHCP server mode.

–d Debugging mode. The daemon remains as a foreground process, and displays verbose messages as it processes DHCP and / or BOOTP datagrams. Messages are displayed on the current TTY. This option can be used in both DHCP / BOOTP server mode and BOOTP relay agent mode.

–h *relay_hops*
Specifies the maximum number of relay agent hops that can occur before the daemon drops the DHCP / BOOTP datagram. The default number of relay agent hops is 4. This option affects both DHCP / BOOTP server mode and BOOTP relay agent mode.

–i *interface, . . .*
Selects the network interfaces that the daemon should monitor for DHCP / BOOTP datagrams. The daemon will ignore DHCP / BOOTP datagrams on network interfaces not specified in this list. This option is only useful on machines that have multiple network interfaces. If this option is not specified, then the daemon will listen for DHCP / BOOTP datagrams on all network interfaces. The option argument consists of a comma-separated list of interface names. It affects both DHCP / BOOTP server and BOOTP relay agent run modes.

–n Disable automatic duplicate IP address detection. When this option is specified, the DHCP server will not attempt to verify that an IP address it is about to offer a client is not in use. By default, the DHCP server pings an IP address before offering it to a DHCP / BOOTP client, to verify that the address is not in use by another machine.

–o *DHCP_offer_Time_To_Live*
Specifies the number of seconds the DHCP server should cache the offers it has extended to discovering DHCP clients. The default setting is **10** seconds. On slow network media, this value can be increased to compensate for slow network performance. This option only affects DHCP server mode.

–r *IP_address | hostname, . . .*
This option enables BOOTP relay agent mode. The option argument specifies a comma-separated list of IP addresses or hostnames of DHCP or BOOTP servers to which the relay agent is to forward BOOTP requests. When the daemon is started in this mode, any DHCP databases are ignored, and the daemon simply acts as a BOOTP relay agent.

A BOOTP relay agent listens to UDP port 68, and forwards BOOTP request packets received on this port to the destinations specified on the command line. It supports the **BROADCAST** flag described in RFC 1542. A BOOTP relay agent can run on any machine that has knowledge of local routers, and thus does not have to be an Internet gateway machine.

Note that the proper entries must be made to the **netmasks** database so that the DHCP server being served by the BOOTP relay agents can identify the subnet mask of the foreign BOOTP / DHCP client's network. See **netmasks**(4) for the format and use of this database.

−t *dhcptab_rescan_interval*
Specifies the interval in minutes that the DHCP server should use to schedule the automatic rereading of the **dhcptab** information. By default, the DHCP server reads the **dhcptab** database at startup or upon receipt of a **SIGHUP** signal. Typically, one would use this option if the changes to the **dhcptab** are relatively frequent. Once the contents of the **dhcptab** have stabilized, one can turn off this option to avoid needless reinitialization from the **dhcptab**.

−v Verbose mode. The daemon displays more messages than in the default mode. Note that verbose mode can reduce daemon efficiency due to the time taken to display messages. Messages are displayed to the current TTY if the debugging option is used; otherwise, messages are logged to the **syslogd** facility. This option can be used in both DHCP / BOOTP server mode and BOOTP relay agent mode.

EXAMPLES

The following command starts a DHCP server in BOOTP compatibility mode, permitting the server to automatically allocate permanent IP addresses to BOOTP clients which are not registered in the server's database; limits the server's attention to incoming datagrams on network devices **le2** and **tr0**; drops BOOTP packets whose hop count exceeds 2; configures the DHCP server to cache extended DHCP offers for 15 seconds; and schedules **dhcptab** rescans to occur every 10 minutes:

```
# in.dhcpd −i le2,tr0 −h 2 −o 15 −t 10 −b automatic
```

The following command starts the daemon in BOOTP relay agent mode, registering the hosts **bladerunner** and **10.0.0.5** as relay destinations, with debugging and verbose modes enabled, and drops BOOTP packets whose hop count exceeds 5:

```
# in.dhcpd −d −v −h 5 −r bladerunner,10.0.0.5
```

FILES

/var/dhcp/dhcptab file or NIS+ table

/var/dhcp/NNN_NNN_NNN_NNN

where **NNN_NNN_NNN_NNN** are database files(s) or NIS+ table(s) which are named for the network they support. For example, **10_0_0_0** is the **dhcp network** database which serves the **10.0.0.0** network. See **dhcp_network**(4) for more details.

/etc/hosts file or NIS+ table

/etc/init.d/dhcp file

/etc/default/dhcp configuration file. See **dhcp**(4) for more details.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu

SEE ALSO

dhcp_config(1M), **dhtadm**(1M), **pntadm**(1M), **syslogd**(1M), **dhcp**(4), **dhcp_network**(4), **dhcptab**(4), **ethers**(4), **hosts**(4), **netmasks**(4), **nsswitch.conf**(4), **attributes**(5)

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 2132, Silicon Graphics, Inc., Bucknell University, March 1997.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 2131, Bucknell University, March 1997.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

NAME

inetd – Internet services daemon

SYNOPSIS

inetd [**–d**] [**–s**] [**–t**] [**–r** *count interval*] [*configuration-file*]

DESCRIPTION

inetd is the server process for the Internet standard services. It is usually started up at system boot time. The *configuration-file* lists the services that **inetd** is to provide. If no *configuration-file* is given on the command line, **inetd** reads its configuration information from the file **/etc/inetd.conf**. See **inetd.conf**(4) for more information on the format of this file. **inetd** listens for service requests on the TCP or UDP ports associated with each of the service listed in the configuration file. When a request arrives, **inetd** executes the server program associated with the service.

A service can be configured to be ‘‘single-threaded’’, in which case **inetd** waits for the server process to exit before starting a second server process. RPC services can also be started by **inetd**.

inetd provides a number of simple Internet services internally. These include **echo**, **discard**, **chargen** (character generator), **daytime** (human-readable time), and **time** (machine-readable time, in the form of the number of seconds since midnight, January 1, 1900).

inetd rereads its configuration file once when it is started and again whenever it receives a hangup signal, **SIGHUP**. New services can be activated, and existing services deleted or modified by editing the configuration file, then sending **inetd** a **SIGHUP** signal.

OPTIONS

–d Runs **inetd** in the foreground and enables debugging output.

–s Allows you to run **inetd** ‘‘stand-alone,’’ outside the Service Access Facility (SAF). If the **–s** option is omitted, **inetd** will attempt to contact the service access controller (SAC) and will exit if SAC is not already running. See **sac**(1M)**.**

–t Instructs **inetd** to trace the incoming connections for all of its TCP services. It does this by logging the client’s IP address and TCP port number, along with the name of the service, using the **syslog**(3) facility. UDP services can not be traced. When tracing is enabled, **inetd** uses the syslog facility code ‘‘daemon’’ and ‘‘notice’’ priority level.

–r Allows **inetd** to detect and then suspend ‘‘broken’’ servers. The **–r** flag has the form:

–r *count interval*

count and *interval* are decimal numbers that represent the maximum *count* of invocations per *interval* of seconds a service may be started before the service is considered ‘‘broken’’.

Once considered ‘‘broken’’, a server is suspended for ten minutes. After ten minutes, **inetd** again enables service, hoping the server behaves correctly.

If the **–r** flag is not specified, **inetd** behaves as though **–r40 60** was specified.

OPERANDS

configuration-file
Lists the services **inetd** is to provide.

EXIT STATUS

inetd does not return an Exit Status.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

in.ftpd(1M), **in.rexecd**(1M), **in.rshd**(1M), **in.tftpd**(1M), **sac**(1M), **inetd.conf**(4), **attributes**(5)

Postel, Jon, ''Echo Protocol,'' RFC 862, Network Information Center, SRI International, Menlo Park, CA, May 1983.

Postel, Jon, ''Discard Protocol,'' RFC 863, Network Information Center, SRI International, Menlo Park, CA, May 1983.

Postel, Jon, ''Character Generator Protocol,'' RFC 864, Network Information Center, SRI International, Menlo Park, CA, May 1983.

Postel, Jon, ''Daytime Protocol,'' RFC 867, Network Information Center, SRI International, Menlo Park, CA, May 1983.

Postel, Jon, and Ken Harrenstien, ''Time Protocol,'' RFC 868, Network Information Center, SRI International, Menlo Park, CA, May 1983.

WARNINGS

Do not configure **udp** services as **nowait**. This will cause a race condition where the **inetd** program selects on the socket and the server program reads from the socket. Many server programs will be forked and performance will be severely compromised.

NOTES

For RPC services, **inetd** listens on all the transports (not only **tcp** and **udp**) as specified for each service in the **inetd.conf**(4) file.

NAME

in.fingerd, fingerd – remote user information server

SYNOPSIS

/usr/sbin/in.fingerd

DESCRIPTION

fingerd implements the server side of the Name / Finger protocol, specified in RFC 742. The Name / Finger protocol provides a remote interface to programs which display information on system status and individual users. The protocol imposes little structure on the format of the exchange between client and server. The client provides a single command line to the finger server which returns a printable reply.

fingerd waits for connections on TCP port 79. Once connected, it reads a single command line terminated by RETURN-LINEFEED and passes the arguments to **finger**(1), prepended with –**s**. **fingerd** closes its connections as soon as the output is finished.

You must invoke **fingerd** from **inetd**. See **inetd**(1M) for more information.

FILES

/var/adm/utmp	user and accounting information
/etc/passwd	system password file
/var/adm/lastlog	last login times
$HOME /.plan	user's plans
$HOME /.project	user's projects

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

finger(1), **inetd**(1M), **attributes**(5)

Harrenstien, Ken, *NAME/FINGER*, RFC 742, Network Information Center, SRI International, Menlo Park, Calif., December 1977.

NOTES

Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. **fingerd** should be taught to filter out IAC's and perhaps even respond negatively (IAC *will not)* to all option commands received.

NAME

infocmp – compare or print out terminfo descriptions

SYNOPSIS

/usr/bin/infocmp [–d] [–c] [–n] [–I] [–L] [–C] [–r] [–u] [–s d | i | l | c] [–v] [-V] [–1] [–w *width***] [–A** *directory***] [–B** *directory***] [***termname . . .***]**

DESCRIPTION

infocmp compares a binary **terminfo** entry with other terminfo entries, rewrites a **terminfo** description to take advantage of the **use=** terminfo field, or prints out a **terminfo** description from the binary file (**term**) in a variety of formats. It displays boolean fields first, then numeric fields, followed by the string fields. If no options are specified and zero, or one *termname* is specified, the –**I** option is assumed. If more than one *termname* is specified, the –**d** option is assumed.

OPTIONS

The –**d ,** –**c ,** and –**n** options can be used for comparisons. **infocmp** compares the **terminfo** description of the first terminal *termname* with each of the descriptions given by the entries for the other terminal's *termname*. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: **F** for boolean variables, –**1** for integer variables, and **NULL** for string variables.

–**d** Produce a list of each capability that is different between two entries. This option is useful to show the difference between two entries, created by different people, for the same or similar terminals.

–**c** Produce a list of each capability that is common between two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the –**u** option is worth using.

–**n** Produce a list of each capability that is in neither entry. If no *termname* is given, the environment variable **TERM** will be used for both of the *termnames*. This can be used as a quick check to see if anything was left out of a description.

The –**I ,** –**L ,** and –**C** options will produce a source listing for each terminal named.

–**I** Use the **terminfo** names.

–**L** Use the long C variable name listed in < **term.h** >.

–**C** Use the **termcap** names. The source produced by the –**C** option may be used directly as a **termcap** entry, but not all of the parameterized strings may be changed to the **termcap** format. **infocmp** will attempt to convert most of the parameterized information, but anything not converted will be plainly marked in the output and commented out. These should be edited by hand.

–**r** When using –**C ,** put out all capabilities in **termcap** form.

If no *termname* is given, the environment variable **TERM** will be used for the terminal name.

All padding information for strings will be collected together and placed at the beginning of the string where **termcap** expects it. Mandatory padding (padding information with a trailing '/') will become optional.

All **termcap** variables no longer supported by **terminfo ,** but are derivable from other **terminfo** variables, will be displayed. Not all **terminfo** capabilities will be translated; only those variables which were part of **termcap** will normally be displayed. Specifying the **−r** option will take off this restriction, allowing all capabilities to be displayed in **termcap** form.

Note that because padding is collected to the beginning of the capability, not all capabilities are displayed. Mandatory padding is not supported. Because **termcap** strings are not as flexible, it is not always possible to convert a **terminfo** string capability into an equivalent **termcap** format. A subsequent conversion of the **termcap** file back into **terminfo** format will not necessarily reproduce the original **terminfo** source.

Some common **terminfo** parameter sequences, their **termcap** equivalents, and some terminal types which commonly have such sequences, are:

terminfo	**termcap**	Representative Terminals
%p1%c	**%.**	adm
%p1%d	**%d**	hp, ANSI standard, vt100
%p1%'x'%+%c	**%+x**	concept
%i	**%i**	ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;	**%>xy**	concept
%p2 is printed before **%p1**	**%r**	hp

−u Produce a **terminfo** source description of the first terminal *termname* which is relative to the sum of the descriptions given by the entries for the other terminals' *termname*s. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with **use=** fields for the other terminals. In this manner, it is possible to retrofit generic **terminfo** entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times, or by different people so that each description is a full description, using **infocmp** will show what can be done to change one description to be relative to the other.

A capability is displayed with an at-sign (@) if it no longer exists in the first *termname*, but one of the other *termname* entries contains a value for it. A capability's value is displayed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries that has this capability gives a different value for that capability.

The order of the other *termname* entries is significant. Since the **terminfo** compiler **tic** does a left-to-right scan of the capabilities, specifying two **use=** entries that contain differing entries for the same capabilities will produce different results, depending on the order in which the entries are given. **infocmp** will flag any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability *after* a **use=** entry that contains, it will cause the second specification to be ignored. Using **infocmp** to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying superfluous **use=** fields. **infocmp** will flag any superfluous **use=** fields.

–s Sorts the fields within each type according to the argument below:

- **d** Leave fields in the order that they are stored in the **terminfo** database.
- **i** Sort by **terminfo** name.
- **l** Sort by the long C variable name.
- **c** Sort by the **termcap** name.

If the **–s** option is not given, the fields are sorted alphabetically by the **terminfo** name within each type, except in the case of the **–C** or the **–L** options, which cause the sorting to be done by the **termcap** name or the long C variable name, respectively.

–v Print out tracing information on standard error as the program runs.

–V Print out the version of the program in use on standard error and exit.

–1 Print the fields one to a line. Otherwise, the fields are printed several to a line to a maximum width of 60 characters.

–w *width* Changes the output to *width* characters.

The location of the compiled **terminfo** database is taken from the environment variable **TERMINFO**. If the variable is not defined, or the terminal is not found in that location, the system **terminfo** database, usually in **/usr/share/lib/terminfo**, is used. The options **–A** and **–B** may be used to override this location.

–A *directory* Set **TERMINFO** for the first *termname*.

–B *directory* Set **TERMINFO** for the other *termname*s. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people.

FILES

/usr/share/lib/terminfo/?/* Compiled terminal description database.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

captoinfo(1M), **tic**(1M), **curses**(3X), **terminfo**(4), **attributes**(5)

NAME

in.ftpd, ftpd – file transfer protocol server

SYNOPSIS

in.ftpd [**–dl**] [**–t** *timeout*]

DESCRIPTION

in.ftpd is the Internet File Transfer Protocol (FTP) server process. The server is invoked by the Internet daemon **inetd**(1M) each time a connection to the FTP service (see **services**(4)) is made.

OPTIONS

–d Debugging information is logged to the system log daemon **syslogd**(1M).

–l Each FTP session is logged to the system log daemon **syslogd**(1M).

–t *timeout* Set the inactivity timeout period to *timeout* seconds. The FTP server will timeout an inactive session after 15 minutes.

Requests

The FTP server currently supports the following FTP requests; case is not distinguished.

ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (**ls –lg**)
MKD	make a directory
MODE	specify data transfer *mode*
NLST	give name list of files in directory (**ls**)
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name
STOR	store a file

STOU	store a file with a unique name
STRU	specify data transfer *structure*
TYPE	specify data transfer *type*
USER	specify user name
XCUP	change to parent of current working directory
XCWD	change working directory
XMKD	make a directory
XPWD	print the current working directory
XRMD	remove a directory

The remaining FTP requests specified in RFC 959 are recognized, but not implemented.

The FTP server will abort an active file transfer only when the **ABOR** command is preceded by a Telnet “Interrupt Process” (IP) signal and a Telnet “Synch” signal in the command Telnet stream, as described in RFC 959.

in.ftpd interprets file names according to the “globbing” conventions used by **sh**(1). This allows users to utilize the metacharacters: ∗ **? [ ] { } ˜**

in.ftpd’s umask (which it uses to create files during **PUT** operations) may be adjusted by adding the line

UMASK=nnn

to **/etc/default/ftpd**.

The banner returned by **in.ftpd** in the parenthetical portion of its greeting is configurable. The default is equivalent to "‘uname -sr‘" and will be used if no banner is set in **/etc/default/ftpd**. To set the banner, add a line of the form

BANNER="..."

to **/etc/default/ftpd**. Nonempty banner strings are fed to shells for evaluation. The default banner may also be obtained by

BANNER="`uname -s` `uname -r`"

and no banner will be printed if **/etc/default/ftpd** contains

BANNER=""

in.ftpd authenticates users according to four rules.

First, the user name must be in the password data base, **/etc/passwd**, and have a password that is not NULL. A password must always be provided by the client before any file operations may be performed. The PAM framework (see **SECURITY** below) is used to verify that the correct password was entered.

Second, if the user name appears in the file **/etc/ftpusers**, **ftp** access is denied.

Third, **ftp** access is denied if the user's shell (from **/etc/passwd**) is not listed in the file **/etc/shells**. If the file **/etc/shells** does not exist, then the user's shell must be one of the following:

/usr/bin/sh **/usr/bin/csh** **/usr/bin/ksh**
/usr/bin/jsh **/bin/sh** **/bin/csh**
/bin/ksh **/bin/jsh** **/sbin/sh**
/sbin/jsh

Fourth, if the user name is "anonymous" or "ftp", an entry for the user name *ftp* must be present in the password and shadow files. The user is then allowed to log in by specifying any password — by convention this is given as the user's e-mail address (such as **user@host.Sun.COM**). Do not specify a valid shell in the password entry of the *ftp* user, and do not give it a valid password (use NP in the encrypted password field of the shadow file).

For anonymous ftp users, **in.ftpd** takes special measures to restrict the client's access privileges. The server performs a **chroot**(2) command to the home directory of the "ftp" user. In order that system security is not breached, it is recommended that the "ftp" subtree be constructed with care; the following rules are suggested.

~ftp Make the home directory owned by **root** and unwritable by anyone. This directory should not be on a file system mounted with the **nosuid** option.

~ftp/bin Make this directory owned by the superuser and unwritable by anyone. Make this a symbolic link to **~ftp/usr/bin** The program **ls**(1) must be present to support the list commands. This program should have mode 111.

~ftp/usr/lib Make this directory owned by the superuser and unwritable by anyone. Copy the following shared libraries from **/usr/lib** into this directory:

ld.so.1*
libc.so.1*
libdl.so.1*
libmp.so.2*
libnsl.so.1*
libsocket.so.1*
nss_compat.so.1*
nss_dns.so.1*
nss_files.so.1*
nss_nis.so.1*
nss_nisplus.so.1*
nss_xfn.so.1*
straddr.so*
straddr.so.2*

~ftp/etc Make this directory owned by the superuser and unwritable by anyone. Copies of the files **passwd**(4), **group**(4), and **netconfig**(4) must be present for the **ls**(1) command to work properly. These files should be mode 444.

˜ftp/pub Make this directory mode 777 and owned by **ftp**. Users should then place files which are to be accessible via the anonymous account in this directory.

˜ftp/dev Make this directory owned by the superuser and unwritable by anyone. First perform **ls –lL** on the device files listed below to determine their major and minor numbers, then use **mknod** to create them in this directory.

/dev/zero
/dev/tcp
/dev/udp
/dev/ticotsord

Set the read and write mode on these nodes to 666 so that passive **ftp** will not fail with "permission denied" errors.

˜ftp/usr/share/lib/zoneinfo
Make this directory mode 555 and owned by the superuser. Copy its contents from **/usr/share/lib/zoneinfo**. This enables **ls –l** to display time and date stamps correctly.

SECURITY

in.ftpd uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the module to be used for **in.ftpd**. Here is a partial **pam.conf** file with entries for the **in.ftpd** command using the UNIX authentication, account management, and session management module.

```
ftp    auth       required     /usr/lib/security/pam_unix.so.1
ftp    account    required     /usr/lib/security/pam_unix.so.1
ftp    session    required     /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **ftp** service, then the entries for the "other" service will be used. Unlike **login**, **passwd**, and other commands, the **ftp** protocol will only support a single password. Using multiple modules will prevent **in.ftpd** from working properly.

EXAMPLES

To set up anonymous ftp, add the following entry to the **/etc/passwd** file. In this example, **/export/ftp** was chosen to be the anonymous ftp area, and the shell is the non-existent file **/nosuchshell**. This prevents users from logging in as the ftp user.

```
ftp:x:30000:30000:Anonymous FTP:/export/ftp:/nosuchshell
```

Add the following entry to the **/etc/shadow** file:

```
ftp:NP:6445::::::
```

The following shell script sets up the anonymous ftp area. It presumes that names are resolved using NIS.

```
#!/bin/sh
# script to setup anonymous ftp area
#

# verify you are root
/usr/bin/id | grep -w ’uid=0’ >/dev/null 2>&1
```

```
if [ "$?" != "0" ]; then
    echo
    exit 1
fi

# handle the optional command line argument
case $# in

  # the default location for the anon ftp comes from the passwd file
  0) ftphome="‘getent passwd ftp | cut -d: -f6‘"
    ;;

  1) if [ "$1" = "start" ]; then
      ftphome="‘getent passwd ftp | cut -d: -f6‘"
    else
      ftphome=$1
    fi
    ;;

  *) echo "Usage: $0 [anon-ftp-root]"
    exit 1
    ;;
esac

if [ -z "${ftphome}" ]; then
  echo "$0: ftphome must be non-null"
  exit 2
fi

case ${ftphome} in
    /*) # ok
      ;;

    *) echo "$0: ftphome must be an absolute pathname"
      exit 1
      ;;
esac

# This script assumes that ftphome is neither / nor /usr so ...
if [ -z "${ftphome}" -o "${ftphome}" = "/" -o "${ftphome}" = "/usr" ]; then
  echo "$0: ftphome must be non-null and neither / or /usr"
  exit 2
fi

# If ftphome does not exist but parent does, create ftphome
```

```
if [ ! -d ${ftphome} ]; then
   # lack of -p below is intentional
   mkdir ${ftphome}
fi
chown root ${ftphome}
chmod 555 ${ftphome}

echo Setting up anonymous ftp area ${ftphome}

# Ensure that the /usr directory exists
if [ ! -d ${ftphome}/usr ]; then
   mkdir -p ${ftphome}/usr
fi
# Now set the ownership and modes to match the man page
chown root ${ftphome}/usr
chmod 555 ${ftphome}/usr

# Ensure that the /usr/bin directory exists
if [ ! -d ${ftphome}/usr/bin ]; then
   mkdir -p ${ftphome}/usr/bin
fi
# Now set the ownership and modes to match the man page
chown root ${ftphome}/usr/bin
chmod 555 ${ftphome}/usr/bin

# this may not be the right thing to do
# but we need the bin -> usr/bin link
rm -f ${ftphome}/bin
ln -s usr/bin ${ftphome}/bin

# Ensure that the /usr/lib and /etc directories exist
if [ ! -d ${ftphome}/usr/lib ]; then
   mkdir -p ${ftphome}/usr/lib
fi
chown root ${ftphome}/usr/lib
chmod 555 ${ftphome}/usr/lib

if [ ! -d ${ftphome}/etc ]; then
   mkdir -p ${ftphome}/etc
fi
chown root ${ftphome}/etc
chmod 555 ${ftphome}/etc

# a list of all the commands that should be copied to ${ftphome}/usr/bin
# /usr/bin/ls is needed at a minimum.
```

```
ftpcmd="
    /usr/bin/ls
"

# ${ftphome}/usr/lib needs to have all the libraries needed by the above
# commands, plus the runtime linker, and some name service libraries
# to resolve names. We just take all of them here.

ftplib="‘ldd $ftpcmd | nawk ’$3 ˜ /lib/ { print $3 }’ | sort | uniq‘"
ftplib="$ftplib /usr/lib/nss_* /usr/lib/straddr* /usr/lib/libmp.so*"
ftplib="$ftplib /usr/lib/libnsl.so.1 /usr/lib/libsocket.so.1 /usr/lib/ld.so.1"
ftplib="‘echo $ftplib | tr ’ ’ ’0 | sort | uniq‘"

cp ${ftplib} ${ftphome}/usr/lib
chmod 555 ${ftphome}/usr/lib/*

cp ${ftpcmd} ${ftphome}/usr/bin
chmod 111 ${ftphome}/usr/bin/*

# you also might want to have separate minimal versions of passwd and group
cp /etc/passwd /etc/group /etc/netconfig ${ftphome}/etc
chmod 444 ${ftphome}/etc/*

# need /etc/default/init for timezone to be correct
if [ ! -d ${ftphome}/etc/default ]; then
    mkdir ${ftphome}/etc/default
fi
chown root ${ftphome}/etc/default
chmod 555 ${ftphome}/etc/default
cp /etc/default/init ${ftphome}/etc/default
chmod 444 ${ftphome}/etc/default/init

# Copy timezone database
mkdir -p ${ftphome}/usr/share/lib/zoneinfo
(cd ${ftphome}/usr/share/lib/zoneinfo
  (cd /usr/share/lib/zoneinfo; find . -print | cpio -o) 2>/dev/null | cpio -imdu 2>/dev/null
  find . -print | xargs chmod 555
  find . -print | xargs chown root
)


# Ensure that the /dev directory exists
if [ ! -d ${ftphome}/dev ]; then
   mkdir -p ${ftphome}/dev
fi
```

```
# make device nodes. ticotsord and udp are necessary for
# 'ls' to resolve NIS names.

for device in zero tcp udp ticotsord ticlts
do
  line=`ls -lL /dev/${device} | sed -e 's/,//'`
  major=`echo $line | awk '{print $5}'`
  minor=`echo $line | awk '{print $6}'`
  rm -f ${ftphome}/dev/${device}
  mknod ${ftphome}/dev/${device} c ${major} ${minor}
done

chmod 666 ${ftphome}/dev/*

## Now set the ownership and modes
chown root ${ftphome}/dev
chmod 555 ${ftphome}/dev

# uncomment the below if you want a place for people to store things,
# but beware the security implications
#if [ ! -d ${ftphome}/pub ]; then
#   mkdir -p ${ftphome}/pub
#fi
#chown ftp ${ftphome}/pub
#chmod 1777 ${ftphome}/pub
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

FILES

/etc/default/ftpd

SEE ALSO

ftp(1), **ld.so.1.1**(1), **ls**(1), **sh**(1), **aset**(1M), **inetd**(1M), **mknod**(1M), **syslogd**(1M), **chroot**(2), **getsockopt**(3N), **pam**(3), **group**(4), **inetd.conf**(4), **netconfig**(4), **netrc**(4), **pam.conf**(4), **passwd**(4), **services**(4), **attributes**(5), **pam_unix**(5)

Postel, Jon, and Joyce Reynolds, *File Transfer Protocol (* FTP), RFC 959, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

DIAGNOSTICS

in.ftpd logs various errors to **syslogd**, with a facility code of **daemon**.

Info Severity

These messages are logged only if the **–l** flag is specified.

FTPD: connection from *host* **at** *time*

A connection was made to **ftpd** from the host *host* at the date and time *time*.

FTPD: User *user* **timed out after** *timeout* **seconds at** *time*

The user *user* was logged out because they had not entered any commands after *timeout* seconds; the logout occurred at the date and time *time*.

Debug Severity

These messages are logged only if the **–d** flag is specified.

FTPD: command: *command*

A command line containing *command* was read from the FTP client.

lost connection

The FTP client dropped the connection.

<—— *replycode*
<—— *replycode–*

A reply was sent to the FTP client with the reply code *replycode*. The next message logged will include the message associated with the reply. If a – follows the reply code, the reply is continued on later lines.

NOTES

The anonymous ftp account is inherently dangerous and should be avoided when possible.

The name service caching daemon **/usr/sbin/nscd** may interfere with some of the functionality of anonymous ftp. The *sublogin* feature does not work unless caching for **passwd** is disabled in **/etc/nscd.conf**.

The server must run as the superuser to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the superuser only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

/etc/ftpusers contains a list of users who cannot access the system; the format of the file is one user name per line.

NAME

init, telinit – process control initialization

SYNOPSIS

/sbin/init [**0123456abcQqSs**]

/etc/telinit [**0123456abcQqSs**]

DESCRIPTION

init is a general process spawner. Its primary role is to create processes from information stored in the file **/etc/inittab**.

Run Level Defined

At any given time, the system is in one of eight possible run levels. A run level is a software configuration under which only a selected group of processes exists. Processes spawned by **init** for each of these run levels are defined in **/etc/inittab**. **init** can be in one of eight run levels, **0**–**6** and **S** or **s** (**S** and **s** are identical). The run level changes when a privileged user runs **/sbin/init**. This sends appropriate signals to the original **init** spawned by the operating system at boot time, saying which run level to invoke.

init and System Booting

When the system is booted, **init** is invoked and the following occurs. First, it reads **/etc/default/init** to set environment variables. This is typically where **TZ** (time zone) and locale-related environments such as **LANG** or **LC_CTYPE** get set.

init then looks in **/etc/inittab** for the **initdefault** entry (see **inittab**(4)). If the **initdefault** entry:

exists
: **init** usually uses the run level specified in that entry as the initial run level to enter.

does not exist
: **/etc/inittab**, **init** asks the user to enter a run level from the system console.

S or **s**
: **init** goes to the single-user state. In this state, the system console device (**/dev/console**) is opened for reading and writing and the command **/sbin/su**, (see **su**(1M)), is invoked. Use either **init** or **telinit** to change the run level of the system. Note that if the shell is terminated (using an end-of-file), **init** only re-initializes to the single-user state if **/etc/inittab** does not exist.

0-6
: **init** enters the corresponding run level. Run levels **0**, **5**, and **6** are reserved states for shutting the system down. Run levels **2**, **3**, and **4** are available as multi-user operating states.

If this is the first time since power up that **init** has entered a run level other than single-user state, **init** first scans **/etc/inittab** for **boot** and **bootwait** entries (see **inittab**(4)). These entries are performed before any other processing of **/etc/inittab** takes place, providing that the run level entered matches that of the entry. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. **init** then scans **/etc/inittab** and executes all other entries that are to be processed for that run level.

To spawn each process in **/etc/inittab**, **init** reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by **/etc/inittab**, **init** waits for one of its descendant processes to die, a **powerfail** signal, or a signal from another **init** or **telinit** process to change the system's run level. When one of these conditions occurs, **init** re-examines **/etc/inittab**.

inittab Additions

New entries can be added to **/etc/inittab** at any time; however, **init** still waits for one of the above three conditions to occur before re-examining **/etc/inittab**. To get around this, **init Q** or **init q** command wakes **init** to re-examine **/etc/inittab** immediately.

When **init** comes up at boot time and whenever the system changes from the single-user state to another run state, **init** sets the **ioctl**(2) states of the console to those modes saved in the file **/etc/ioctl.syscon**. **init** writes this file whenever the single-user state is entered.

Run Level Changes

When a run level change request is made, **init** sends the warning signal (**SIGTERM**) to all processes that are undefined in the target run level. **init** waits five seconds before forcibly terminating these processes by sending a kill signal (**SIGKILL**).

When **init** receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in **/var/adm/utmp** and **/var/adm/wtmp** if it exists (see **who**(1)). A history of the processes spawned is kept in **/var/adm/wtmp.**

If **init** receives a **powerfail** signal (**SIGPWR**) it scans **/etc/inittab** for special entries of the type **powerfail** and **powerwait**. These entries are invoked (if the run levels permit) before any further processing takes place. In this way **init** can perform various cleanup and recording functions during the powerdown of the operating system.

/etc/defaults/init File

Default values can be set for the following flags in **/etc/default/init**. For example:
TZ=US / Pacific

TZ Either specifies the timezone information (see **ctime**(3C)) or the name of a timezone information file **/usr/share/lib/zoneinfo**.

LC_CTYPE Character characterization information.

LC_MESSAGES Message translation.

LC_MONETARY Monetary formatting information.

LC_NUMERIC Numeric formatting information.

LC_TIME Time formatting information.

LC_ALL If set, all other **LC_*** environmental variables take-on this value.

LANG If **LC_ALL** is not set, and any particular **LC_*** is also not set, the value of **LANG** is used for that particular environmental variable.

telinit

telinit, which is linked to **/sbin/init**, is used to direct the actions of **init**. It takes a one-character argument and signals **init** to take the appropriate action.

SECURITY

init uses **pam**(3) for session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the session management module to be used for **init**. Here is a partial **pam.conf** file with entries for **init** using the UNIX session management module.

```
        init    session    required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **init** service, then the entries for the "other" service will be used.

OPTIONS

0 Go into firmware.

1 Put the system in system administrator mode. All file systems are mounted. Only a small set of essential kernel processes are left running. This mode is for administrative tasks such as installing optional utility packages. All files are accessible and no users are logged in on the system.

2 Put the system in multi-user mode. All multi-user environment terminal processes and daemons are spawned. This state is commonly referred to as the multi-user state.

3 Extend multi-user mode by making local resources available over the network.

4 Is available to be defined as an alternative multi-user environment configuration. It is not necessary for system operation and is usually not used.

5 Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible.

6 Stop the operating system and reboot to the state defined by the **initdefault** entry in **/etc/inittab**.

a, **b**, **c** process only those **/etc/inittab** entries having the **a**, **b**, or **c** run level set. These are pseudo-states, which may be defined to run certain commands, but which do not cause the current run level to change.

Q, **q** Re-examine **/etc/inittab**.

S, **s** Enter single-user mode. When this occurs, the terminal that executed this command becomes the system console. This is the only run level that doesn't require the existence of a properly formatted **/etc/inittab** file. If this file does not exist, then by default, the only legal run level that **init** can enter is the single-user mode. When the system comes up to **S** or **s**, file systems for users' files are not mounted and only essential kernel processes are running. When the system comes down to **S** or **s**, all mounted file systems remain mounted, and all processes started by **init** that should only be running in multi-user mode are killed. In addition, any process that has a **utmp** entry will be killed. This last condition insures that all port monitors started by the SAC are killed and all services started by these port monitors, including **ttymon** login services, are killed. Other processes not started directly by **init** will remain running. For example, **cron** remains running.

FILES

/etc/inittab	controls process dispatching by init
/var/adm/utmp	accounting information
/var/adm/wtmp	history of all logins since file was last created
/etc/ioctl.syscon	

/dev/console	system console device
/etc/default/init	environment variables.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

login(1), **sh**(1), **stty**(1), **who**(1), **shutdown**(1M), **su**(1M), **ttymon**(1M), **ioctl**(2), **kill**(2), **ctime**(3C), **pam**(3), **inittab**(4), **pam.conf**(4), **utmp**(4), **utmpx**(4), **attributes**(5), **pam_unix**(5), **termio**(7I)

DIAGNOSTICS

If **init** finds that it is respawning an entry from **/etc/inittab** more than ten times in two minutes, assumes that there is an error in the command string in the entry, and generates an error message on the system console. It will then refuse to respawn this entry until either five minutes has elapsed or it receives a signal from a user-spawned **init** or **telinit**. This prevents **init** from eating up system resources when someone makes a typographical error in the **inittab** file, or a program is removed that is referenced in **/etc/inittab**.

NOTES

init and **telinit** can be run only by a privileged user.

The **S** or **s** state must not be used indiscriminately in **/etc/inittab**. When modifying this file, it is best to avoid adding this state to any line other than **initdefault**.

If a default state is not specified in the **initdefault** entry in **/etc/inittab**, state **6** is entered. Consequently, the system will loop by going to firmware and rebooting continuously.

If the **utmp** file cannot be created when booting the system, the system will boot to state "**s**" regardless of the state specified in the **initdefault** entry in **/etc/inittab**. This can occur if the **/var** file system is not accessible.

NAME

in.lpd – BSD print protocol adaptor

SYNOPSIS

/usr/lib/print/in.lpd

DESCRIPTION

in.lpd implements the network listening service for the BSD print protocol specified in RFC 1179. The BSD print protocol provides a remote interface for systems to interact with a local spooling system. The protocol defines five standard requests from the client to the server: starting queue processing, transfering print jobs, retrieving terse status, retrieving verbose status, and canceling print jobs.

in.lpd is started from **inetd** (see **inetd**(1M)). **inetd** waits for connections on TCP port 515. Upon receipt of a connect request, **in.lpd** is started to service the connection. Once the request has been filled, **in.lpd** closes the connection and exits.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

FILES

/etc/printers.conf	System printer configuration database.
printers.conf.byname	NIS version of **/etc/printers.conf**.
fns.ctx_dir.*domain*	NIS+ version of **/etc/printers.conf**.
/usr/lib/print/bsd-adaptor/bsd_*.so*	Spooler translation modules.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

inetd(1M), **printers.conf**(4), **attributes**(5)

NAME

in.named, named – internet domain name server

SYNOPSIS

in.named [**–q**] [**–r**] [**–b** *bootfile*] [**–d** *debuglevel*] [**–p** *port*[*/localport*]]

DESCRIPTION

in.named is the Internet domain name server. **in.named** spawns the **named-xfer** process whenever it needs to perform a zone transfer (see **named-xfer**(1M)).

The **in.named** name service is used by hosts on the Internet to provide access to the Internet distributed naming database. See *RFC 1034* and *RFC 1035* for more information on the Internet name-domain system.

With no arguments, **in.named** reads the default boot file **/etc/named.boot** for any initial data, and listens for queries.

Any additional arguments beyond those shown in the **SYNOPSIS** section are interpreted as the names of boot files. If multiple boot files are specified, only the last is used.

The name server reads the boot file to obtain instructions on where to find its initial data.

OPTIONS

–b *bootfile*
Use *bootfile* rather than **/etc/named.boot**. This option allows filenames to begin with a leading dash.

–d *level*
Print debugging information. *level* is a number indicating the level of messages printed.

–p *port***/***localport*
Use different, port numbers. The default is the standard port number as returned by **getservbyname**(3N) for service **domain**. The **–p** argument can specify up to two port numbers. The specification of two port numbers requires a '/' (slash) separator. In this case, the first port is used when contacting remote servers, and the second one is the service port bound by the local instance of **in.named**. This option is used mostly for debugging purposes.

–q
Trace all incoming queries. Note: this option is ignored in favor of the boot file directive, **options query-log**, when both options are used.

–r
Turns recursion off in the server. Answers can come only from local (primary or secondary) zones. This option can be used on root servers. Note: This option will probably be eventually abandoned in favor of the boot file directive, **options no-recursion**.

USAGE

/etc/named.boot File Directives

The following is a sample **/etc/named.boot** file containing directives to guide the **in.named** process at startup time.

Long directives must fit on a single line; they will be misinterpreted if continued on subsequent lines. The semicolon character initiates the beginning of a line of ignored (comment) text. Text is ignored between the semicolon and the next newline character.

```
;boot file for the name server
directory /usr/local/adm/named

;type       domain                   source host/file            backup_file

cache       .                        root.cache
primary     Berkeley.EDU             berkeley.edu.zone
primary     32.128.IN-ADDR.ARPA      ucbhosts.rev
secondary   CC.Berkeley.EDU          128.32.137.8 128.32.137.3   cc.zone.bak
secondary   6.32.128.IN-ADDR.ARPA    128.32.137.8 128.32.137.3   cc.rev.bak
primary     0.0.127.IN-ADDR.ARPA     localhost.rev
forwarders  10.0.0.78 10.2.0.78
limit       transfers-in 10
options     forward-only
            query-log fake-iquery
```

The following list includes all of the directives that may be placed in the boot file. If a list item is present in the previous sample file, a discussion of the item in the sample file is included. These discussions are always introduced with the phrase, “Example Directive.”

bogusns *address-list*

Specifies that this server implements a special form of primitive access control. This server will not process queries which need to be sent to name server addresses (specified as dotted quads, not as domain names) given in *address-list*. This directive is useful when you know that some popular server has bad data in a zone or cache, and you want to avoid contamination while the problem is being fixed.

cache *domain root-server-hints-file*

Specifies that this server plays a caching name-server role by reading and occasionally updating *root-server-hints-file*. Access to this cache is not usually required during normal operation. During startup, however, it is used to obtain hints for finding the current root servers. The initial data in *root-server-hints-file* typically includes the locations of root domain servers for a local network that is not directly connected to the Internet. Otherwise, it lists Internet name servers.

The information in *root-server-hints-file* typically includes a list of root servers specified in terms of their name and address, and formatted in terms of resource records of type **NS** (nameserver) and type **A** (Internet address). (The resource record formats are described later in the section, **Format of Resource Records in Zone Files**.) The **root.cache** cache file for a server that is being connected directly to the Internet should be retrieved periodically from **FTP.RS.INTERNIC.NET**, since it contains a list of root servers that is occasionally updated.

Example Directive: The **cache** line in the example configuration file specifies that **root.cache** is the place for **in.named** to find and maintain a backup cache of root name servers.

directory *dir*

Causes the name server to change its working directory to the specified directory. This can help ensure the correct processing of any **$INCLUDE** directives that may reside in files that are referenced within **primary** directives in the configuration file.

Example Directive: The **directory** line in the example configuration file specifies that **/usr/local/adm/named** becomes the current path, when relative pathnames need to be resolved within the configuration file or within files that the configuration file references.

forwarders *address-list*

Specifies the addresses of sitewide servers that will accept recursive queries from other servers. If the boot file specifies one or more forwarders, this server sends all queries for data not in the cache to the forwarders first. Each forwarder is asked in turn until an answer is returned or the list is exhausted. If no answer is forthcoming from any forwarder, the server continues as it would have without the forwarders line unless it is in a forward-only mode.

The forwarding facility is useful for generating a large sitewide cache on a master, and for reducing traffic over links to outside servers. It can also be used to allow servers to run, that do not have direct access to the Internet, but wish to look up exterior names anyway.

Example Directive: The **forwarders** line in the example configuration file specifies **10.0.0.78** and **10.2.0.78** as two sitewide servers that can accept recursive queries from other servers.

include *include-file*

Causes the contents of *include-file* to be interpreted as additional boot directives, as though these contents had actually been entered at this location in the boot file. This can be useful for dealing with many zones or for implementing logical groupings of zones which are maintained by different people. Quotes around the filename are not necessary.

limit *parameter-name size*

Resets the internal limits of the **BIND** name server. Some of the supported limit values (**datasize**, for example) are implemented by the system and others (such as **transfers-in**) are implemented by **BIND** itself. The possible *parameter-name* and *size* values are:

datasize

The **datasize** limit value is not supported.

transfers-in *max-processes*

Sets the maximum number of **named-xfer** subprocesses which **BIND** will spawn at any one time.

transfers-per-ns *max-transfers*

Sets the maximum number of zone transfers to be simultaneously initiated to any given remote name server.

loglevel *syslog-priority*

Sets the minimum priority level of **syslog**(3) messages. Syslog messages at a lower priority (larger numerical value) are suppressed. The default is 5, corresponding to **LOG_NOTICE**, which filters out most informational chatter. Setting the loglevel to 7 **(LOG_DEBUG)** allows all messages. Suppressing at level 3 **(LOG_ERR)** or higher may mean that **in.named** will be unable to report fatal problems.

See **syslog** and **sys/syslog.h** for priorities and corresponding numerical values.

listen-backlog *backlog*

Sets the backlog parameter for **listen**(3N). The maximum value is limited by the tcp_conn_req_max TCP configuration parameter. The default backlog is 5. The listen-backlog value is changed only when **in.named** is restarted; a **SIGHUP** is not sufficient.

Example Directive: The **limit** line in the example configuration file specifies that as many as ten concurrent **BIND** subprocesses can be spawned on this server.

max-fetch *size*

Establishes a limit that is identical to the one previously described for **limit transfers-in**. This directive is provided for backward compatibility.

options *option-list*

Switch on certain options that effect changes to the behavior of **BIND**. More than one boolean option can be specified in a single directive. The currently defined options are:

fake-iquery

Since **in.named** always responds to inverse queries, the fake-iquery option has no effect.

forward-only

Causes the server to query only its forwarders. This option is normally used on a system that wishes to run a server but for physical or administrative reasons cannot be given access to the Internet.

no-fetch-glue

Keeps **BIND** from fetching missing glue when constructing the additional data section of a response. This can be used in conjunction with **no-recursion** to prevent **BIND**'s cache from ever growing in size or becoming corrupted.

no-recursion

Causes **BIND** to answer with a referral rather than actual data whenever it receives a query for a name for which this server does not play an authoritative role. Avoid setting this option on a server that is listed in any host's **resolv.conf** file.

query-log

Causes all queries to be logged through **syslog( )** (see **syslogd**(1M) and **syslog**(3)). Because this option results in the generation of considerable data, don't turn it on lightly.

Example Directive: The **options** line in the example configuration file specifies **forward-only**, **query-log**, and **fake-iquery**, requesting the associated modifications in **BIND** behavior for this server.

primary *zone master-file*

Specifies that this server plays the primary name-server role for *zone*, by following the authoritative specifications in *master-file*. Each master file should begin with an SOA record for the zone (see **Zone File Format**).

Example Directives:

The first **primary** line in the example configuration file specifies that the file, **berkeley.edu.zone**, contains authoritative data for the **Berkeley.EDU** zone. Accordingly, the master file, **berkeley.edu.zone**, must contain data in the master file format described in *RFC 883*. All domain names are relative to the origin, which in this case is **Berkeley.EDU** (see below for a more detailed description).

The second **primary** line in the example configuration file states that the file **ucbhosts.rev** contains authoritative data for the zone, **32.128.IN-ADDR.ARPA.** These data are used to translate addresses in network 128.32 to hostnames.

The third **primary** line in the example configuration file is interpreted in the same way as the preceding ones.

secondary *zone address-list zone-backup-file*

Specifies that this server plays a secondary name-server role for zone *domain* by fetching authoritative data moderated through the IP addresses in *address-list*. This server will try as many as 10 addresses specified in *address-list*, in the order that they are listed. Any argument following *address-list* that lacks the dotted-quad address format is interpreted as the *zone-backup-file*, the file into which it places a backup of the transferred zone. This secondary copy of the zone on this server is nevertheless considered authoritative for the specified domain. Whenever a new copy of the domain is received by automatic zone transfer from one of the master servers, this file is updated. If no *zone-backup-file* is specified, a temporary file is used. The temporary file is deleted after each successful zone transfer. Not specifying a *zone-backup-file* is not recommended because it is a needless waste of bandwidth.

Example Directive: The first **secondary** line in the example configuration file specifies that the authoritative data for the zone, **CC.Berkeley.EDU**, is taken from the name server at IP address, **128.32.137.8**. If the transfer fails for this first address, this server tries 128.32.137.3. Furthermore, when it boots, this name server will load the zone from the backup file **cc.zone.bak** if it exists, providing a complete copy even if the master servers are unreachable.

The second **secondary** line in the example configuration file specifies that the address-to-hostname mapping for the subnet **128.32.136** should be obtained from the same list of master servers as those specified in the previous line.

slave Establishes forward-only mode. This option is allowed for backward compatibility. Its meaning is identical to a line containing **options forward-only**.

sortlist *address-list*
Specifies that this server will return host addresses in a sort order moderated by *address-list*. This is used to indicate networks that are to be preferred over other networks. Queries for host addresses from hosts on the same network as the server receives responses, with local network addresses listed first, then addresses on the sort list, then other addresses.

tcplist *address-list*
xfrnets *address-list*
Specifies that this server implements primitive access control in accordance with *address-list*. This directive causes a name server to only answer zone transfer requests from hosts which are on networks listed in *address-list*. This directive may also be given as **tcplist** for compatibility with older, interim servers.

Zone File Format

The zone files are also known as the authoritative master files (data files) for a zone. In the boot file, references were made to these files as part of the specification of any **primary** directives.

Two classes of entries populate the zone files, directives and resource records. The start of the zone file is likely to contain one or two directives that establish a context that modifies the way subsequent records are interpreted.

Resource records for a zone determine how a zone is managed by establishing zone characteristics. For example, one type of zone record establishes the zone's mailbox information.

The very first record of each zone file should be a Start-of-Authority record (**SOA**) for a zone. A multiple-line **SOA** record is presented below. The meaning of the values in this sample will become clearer with the help of a list that describes the purpose of each field in the zone record (see the **SOA** list subitem under the *rr-type* list item in, **Format of Resource Records in Zone Files**).

```
@  IN  SOA  ucbvax.Berkeley.EDU.  rwh.ucbvax.Berkeley.EDU.  (
1989020501      ;serial
10800           ;refresh
3600            ;retry
3600000         ;expire
86400 )         ;minimum
```

Resource records normally end at the end of a line, but may be continued across lines between opening and closing parentheses (as demonstrated by the preceding sample).

Comments are introduced by semicolons. They continue to the end of the line.

Directives in Zone Files

There are two control directives that help determine how the zone file is processed, **$INCLUDE** and **$ORIGIN**.

The **$INCLUDE** directive refers to still another file within which zone characteristics are described. Such files typically contain groups of resource records, but they may also contain further directives.

The **$ORIGIN** directive establishes a current origin that is appended to any *domain* values that do not end with a `.´ (dot). The placeholder *domain* represents the first resource record field as shown in **Format of Resource Records in Zone Files**.

The format for these directives is

$INCLUDE *filename opt-current-domain*
$ORIGIN *current-domain*

where:

current-domain Specifies the value of the current origin that remains in effect for this configuration file unless a subsequent **$ORIGIN** directive overrides it for the remaining portion of the file.

filename Specifies a file, the contents of which are, in effect, incorporated into the configuration file at the location of the corresponding **$INCLUDE** directive.

opt-current-domain Optionally defines a current origin that is applicable only to the records residing in the specified file in the corresponding **$INCLUDE** directive. This directive overrides the origin given in a preceding **$ORIGIN** directive, but only for the scope of the included text. See also *current-domain*.

Neither the *opt-current-domain* argument of **$INCLUDE** nor the **$ORIGIN** directive in the included file can affect the current origin in effect for the remaining records in the main configuration file (as defined by those **$ORIGIN** directives that reside there).

Format of Resource Records in Zone Files

The format of the resource records is

domain opt-ttl opt-class rr-type rr-data . . .

Where:

domain Specifies the domain being described by the current line and any following lines that lack a value for this field. Beware of any *domain* values that you enter without full qualification, because the value of the current origin will be appended to them. The value of the current origin is appended when *domain* does not end with a dot.

A *domain* value specified as the symbol **@** is replaced with the value of the current origin. The *current-domain* or any locally-overriding *opt-current-domain* value is used as its replacement. (For a discussion of these placeholders, see the earlier discussion of the **$ORIGIN** and **$INCLUDE** directives.)

A *domain* value specified as a `.´ (dot) represents the root.

opt-ttl Specifies the number of seconds corresponding to the time-to-live value applicable to the zone characteristic that is defined in the remaining fields. This field is optional. It defaults to zero. Zero is interpreted as the minimum value specified in the SOA record for the zone.

opt-class Specifies the object address type; currently only one type is supported, **IN**, for objects connected to the Internet.

rr-type rr-data . . .

Specifies values that describe a zone characteristic.

Permissible *rr-type* and other field values are listed below. (Field values are listed in the order that they must appear.)

A *address*

Specifies the host address (in dotted-quad format). DCE or AFS server

CNAME *canonical-name*

Specifies in a domain-name format the canonical name for the alias (*domain*).

HINFO *cpu-type OS-type*

Host information supplied in terms of a CPU type and an OS type.

MX *preference mail-exchanger*

Specifies in domain-name format a mail exchanger preceded by a preference value (between **0** and **32767**), with lower numeric values representing higher logical preferences.

NS *authoritative-server*

Specifies in domain-name format an authoritative name server.

NULL Specifies a null zone record.

PTR *domain-pointer*

Specifies in domain-name format a domain name pointer.

RP *mailbox txt-referral*

Offers details about how to reach a responsible person for the domain name.

SOA *host-domain maintainer-addr serial-no refresh retry expire ttl*

Establishes the start of a zone of authority in terms of the domain of the originating host (*host-domain*), the domain address of the maintainer (*maintainer-addr*), a serial number (*serial-no*), the refresh period in seconds (*refresh*), the retry period in seconds (*retry*), the expiration period in seconds (*expire*), and the minimum time-to-live period in seconds (*ttl*). See *RFC 1035.*

The serial number should be changed each time the master file is changed. Secondary servers check the serial number at intervals specified by the refresh time in seconds; if the serial number changes, a zone transfer will be done to load the new data.

If a master server cannot be contacted when a refresh is due, the retry time specifies the interval at which refreshes should be attempted. If a master server cannot be contacted within the interval given by the expire time, all data from the zone is discarded by secondary servers. The minimum value is the time-to-live used by records in the file with no explicit time-to-live value.

The serial number can be given as a dotted number. However, this is a very unwise thing to do, since the translation to normal integers is via concatenation rather than multiplication and addition. You could spell out the year, month, day of month, and 0..99 version number and still fit it inside the unsigned 32-bit size of this field. This strategy should work for the forseeable future (but is questionable after the year 4293).

For more detailed information, see *RFC 883*.

rr-data . . . See the description of *rr-type*.

Consult *Name Server Operations Guide for BIND* for further information about the supported types of resource records.

EXIT STATUS

The **in.named** process returns the following exit statuses:

0 Successful completion.
1 An error occurred.

FILES

/etc/named.boot	name server configuration boot file
/etc/named.pid	the process ID (on older systems)
/var/tmp/named.run	debug output
/var/tmp/named.stats	nameserver statistics data
/var/tmp/named_dump.db	dump of the name servers database
/var/tmp/named.pid	the process ID (on newer systems)

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

kill(1), **named-xfer**(1M), **syslogd**(1M), **getservbyname**(3N), **listen**(3N), **resolver**(3N), **signal**(3C), **syslog**(3), **resolv.conf**(4), **attributes**(5)

Vixie, Paul;Dunlap, Keven J., Karels, Michael J., *Name Server Operations Guide for BIND* (public domain), Internet Software Consortium, 1995

Braden, R. (Editor), *Requirements for Internet Hosts - Applications and Support*, RFC 1123, Internet Engineering Task Force - Network Working Group, October 1989

Mockapetris, Paul, *Domain Names - Concepts and Facilities*, RFC 1034, Network Information Center, SRI International, Menlo Park, Calif., November 1987

Mockapetris, Paul, *Domain Names - Implementation and Specification*, RFC 1035, Network Information Center, SRI International, Menlo Park, Calif., November 1987

Mockapetris, Paul, *Domain System Changes and Observations*, RFC 973, Network Information Center, SRI International, Menlo Park, Calif., January 1986

Partridge, Craig, *Mail Routing and the Domain System*, RFC 974, Network Information Center, SRI International, Menlo Park, Calif., January 1986

NOTES

The following signals have the specified effect when sent to the server process using the **kill**(1) command:

SIGHUP Causes the server to read **/etc/named.boot** and reload the database. **SIGHUP** also causes the server to check the serial number on all secondary zones. Normally the serial numbers are only checked at the intervals specified by the SOA record at the start of each zones-definition file.

SIGINT Dumps the current database and cache to **/var/tmp/named_dump.db**.

SIGIOT Dumps statistical data into **/var/tmp/named.stats**. Statistical data are appended to the file.

SIGUSR1 Turns on debugging at the lowest level when received the first time; receipt of each additional **SIGUSR1** signal causes the server to increment the debug level.

SIGUSR2 Turns off debugging completely.

SIGWINCH Toggles logging of all incoming queries through the **syslog** system daemon (see **syslogd**(1M)).

NAME

in.rarpd, rarpd – DARPA Reverse Address Resolution Protocol server

SYNOPSIS

/usr/sbin/in.rarpd [**–d**] **–a**

/usr/sbin/in.rarpd [**–d**] *device unit*

DESCRIPTION

in.rarpd starts a daemon that responds to Reverse Address Resolution Protocol (RARP) requests. The daemon forks a copy of itself that runs in background. It must be run as root.

RARP is used by machines at boot time to discover their Internet Protocol (IP) address. The booting machine provides its Ethernet address in a RARP request message. Using the **ethers** and **hosts** databases, **in.rarpd** maps this Ethernet address into the corresponding IP address which it returns to the booting machine in an RARP reply message. The booting machine must be listed in both databases for **in.rarpd** to locate its IP address. **in.rarpd** issues no reply when it fails to locate an IP address.

in.rarpd uses the STREAMS-based Data Link Provider Interface (DLPI) message set to communicate directly with the datalink device driver.

OPTIONS

–a Get the list of available network interfaces from IP using the SIOCGIFADDR ioctl and start a RARP daemon process on each interface returned.

–d Print assorted debugging messages while executing.

EXAMPLES

The following command starts an **in.rarpd** for each network interface name returned from **/dev/ip:**

```
example# /usr/sbin/in.rarpd –a
```

The following command starts one **in.rarpd** on the device **/dev/le** with the device instance number 0.

```
example# /usr/sbin/in.rarpd le 0
```

FILES

/etc/ethers	file or NIS+ map
/etc/hosts	file or NIS+ map
/tftpboot	
/dev/ip	
/dev/arp	

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

boot(1M), **ifconfig**(1M), **ethers**(4), **hosts**(4), **netconfig**(4), **attributes**(5), **dlpi**(7P)

RFC-903, *A Reverse Address Resolution Protocol*, Network Information Center, SRI International.

Unix International, *Data Link Provider Interface*, Version 2, May 7, 1991, Sun Microsystems, 800-6915-01.

NAME

in.rdisc, rdisc – network router discovery daemon

SYNOPSIS

/usr/sbin/in.rdisc [**–a**] [**–f**] [**–s**] [*send-address*] [*receive-address*]

/usr/sbin/in.rdisc –r [**–p** *preference*] [**–T** *interval*] [*send-address*] [*receive-address*]

DESCRIPTION

in.rdisc implements the ICMP router discovery protocol. The first form of the command is used on hosts and the second form is used on routers. On a host, **in.rdisc** is invoked at boot time to populate the network routing tables with default routes. On a router, it is also invoked at boot time in order to start advertising the router to all the hosts.

Host (First Form)

On a host, **in.rdisc** listens on the **ALL_HOSTS** (224.0.0.1) multicast address for **ROUTER_ADVERTISE** messages from routers. The received messages are handled by first ignoring those listed router addresses with which the host does not share a network. Among the remaining addresses, the ones with the highest preference are selected as default routers and a default route is entered in the kernel routing table for each one of them.

Optionally, **in.rdisc** can avoid waiting for routers to announce themselves by sending out a few **ROUTER_SOLICITATION** messages to the **ALL_ROUTERS** (224.0.0.2) multicast address when it is started.

A timer is associated with each router address. The address will no longer be considered for inclusion in the routing tables if the timer expires before a new *advertise* message is received from the router. The address will also be excluded from consideration if the host receives an *advertise* message with the preference being maximally negative.

Router (Second Form)

When **in.rdisc** is started on a router, it uses the **SIOCGIFCONF ioctl**(2) to find the interfaces configured into the system and it starts listening on the **ALL_ROUTERS** multicast address on all the interfaces that support multicast. It sends out *advertise* messages to the **ALL_HOSTS** multicast address advertising all its **IP** addresses. A few initial *advertise* messages are sent out during the first 30 seconds and after that it will transmit *advertise* messages approximately every 600 seconds.

When **in.rdisc** receives a *solicitation* message, it sends an *advertise* message to the host that sent the *solicitation* message.

When **in.rdisc** is terminated by a signal, it sends out an *advertise* message with the preference being maximally negative.

OPTIONS

–a Accept all routers independent of the preference they have in their *advertise* messages. Normally, **in.rdisc** only accepts (and enters in the kernel routing tables) the router or routers with the highest preference.

–f Run **in.rdisc** forever even if no routers are found. Normally, **in.rdisc** gives up if it has not received any *advertise* message after soliciting three times, in which case it exits with a non-zero exit code. If **–f** is not specified in the first form then **–s** must be specified.

–r Act as a router, rather than a host.

−s Send three *solicitation* messages initially to quickly discover the routers when the system is booted. When **−s** is specified, **in.rdisc** exits with a non-zero exit code if it can not find any routers. This can be overridden with the **−f** option.

−p *preference* Set the preference transmitted in the *solicitation* messages. The default is zero.

−T *interval* Set the interval between transmitting the *advertise* messages. The default time is 600 seconds.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

in.routed(1M), **ioctl**(2), **attributes**(5), **icmp**(7P), **inet**(7P)

Deering, S.E.,ed ‘‘*ICMP Router Discovery Messages*,’’ RFC 1256, Network Information Center, SRI International, Menlo Park, Calif., September 1991.

NAME

in.rexecd, rexecd – remote execution server

SYNOPSIS

in.rexecd

DESCRIPTION

in.rexecd is the server for the **rexec**(3N) routine. The server provides remote execution facilities with authentication based on user names and passwords. It is invoked automatically as needed by **inetd**(1M), and then executes the following protocol:

1) The server reads characters from the socket up to a null (**\0**) byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.

3) A null terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null terminated password of at most 16 characters is retrieved on the initial socket.

5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted and a diagnostic message is returned.

7) A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M), **rexec**(3N), **attributes**(5)

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

username too long
The name is longer than 16 characters.

password too long
The password is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect.
No password file entry for the user name existed.

Password incorrect.
The wrong password was supplied.

No remote directory.
The **chdir** command to the home directory failed.

Try again.
A **fork** by the server failed.

/usr/bin/sh: ...
The user's login shell could not be started.

NAME

in.rlogind, rlogind – remote login server

SYNOPSIS

/usr/sbin/in.rlogind

DESCRIPTION

in.rlogind is the server for the **rlogin**(1) program. The server provides a remote login facility with authentication based on privileged port numbers.

in.rlogind is invoked by **inetd**(1M) when a remote login connection is established, and executes the following protocol:

- The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.
- The server checks the client's source address. If an entry for the client exists in both **/etc/hosts** and **/etc/hosts.equiv**, a user logging in from the client is not prompted for a password. If the address is associated with a host for which no corresponding entry exists in **/etc/hosts**, the user is prompted for a password, regardless of whether or not an entry for the client is present in **/etc/hosts.equiv** (see **hosts**(4) and **hosts.equiv**(4)).

Once the source port and address have been checked, **in.rlogind** allocates a pseudo-terminal and manipulates file descriptors so that the slave half of the pseudo-terminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the **login**(1) program, invoked with the **–r**.

The login process then proceeds with the **pam**(3) authentication process. See **SECURITY** below. If automatic authentication fails, it reprompts the user to login.

The parent of the login process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, a packet protocol is invoked to provide Ctrl-S / Ctrl-Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, **TERM**; see **environ**(4).

SECURITY

in.rlogind uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **in.rlogind**. Here is a partial **pam.conf** file with entries for the **rlogin** command using the "rhosts" and UNIX authentication modules, and the UNIX account, session management, and password management modules.

```
rlogin   auth      sufficient   /usr/lib/security/pam_rhosts_auth.so.1
rlogin   auth      required     /usr/lib/security/pam_unix.so.1
rlogin   account   required     /usr/lib/security/pam_unix.so.1
rlogin   session   required     /usr/lib/security/pam_unix.so.1
```

With this configuration, the server checks the client's source address. If an entry for the client exists in both **/etc/hosts** and **/etc/hosts.equiv**, a user logging in from the client is not prompted for a password. If the address is associated with a host for which no corresponding entry exists in **/etc/hosts**, the user is prompted for a password, regardless of whether or not an entry for the client is present in **/etc/hosts.equiv** (see **hosts**(4) and **hosts.equiv**(4)).

If there are no entries for the **rlogin** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords. Removing the "pam_rhosts_auth.so.1" entry will disable the **/etc/hosts.equiv** and **˜/.rhosts** authentication protocol and the user would always be forced to type the password. The *sufficient* flag indicates that authentication through the pam_rhosts_auth.so.1 module is "sufficient" to authenticate the user. Only if this authentication fails is the next authentication module used.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

login(1), **rlogin**(1), **in.rshd**(1M), **inetd**(1M), **pam**(3), **environ**(4), **hosts**(4), **hosts.equiv**(4), **pam.conf**(4), **attributes**(5), **pam_rhosts_auth**(5), **pam_unix**(5)

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

Hostname for your address unknown.
No entry in the host name database existed for the client’s machine.

Try again.
A *fork* by the server failed.

/usr/bin/sh: . . .
The user’s login shell could not be started.

NOTES

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ‘‘open’’ environment.

A facility to allow all data exchanges to be encrypted should be present.

NAME

in.routed, routed – network routing daemon

SYNOPSIS

/usr/sbin/in.routed [**–s**] [**–q**] [**–t**] [**–g**] [**–S**] [**–v**] [*logfile*]

DESCRIPTION

in.routed is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up-to-date kernel routing table entries.

In normal operation, **in.routed** listens on **udp**(7P) socket 520 (decimal) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When **in.routed** is started, it uses the **SIOCGIFCONF** **ioctl**(2) to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed the host will forward packets between networks. **in.routed** then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, **in.routed** formulates a reply based on the information maintained in its internal tables. The *response* packet contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned, provides a metric relative to the sender.

request packets received by **in.routed** are used to update the routing tables if one of the following conditions is satisfied:

- No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (that is, the hop count is not infinite).
- The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.
- The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, **in.routed** records the change in its internal tables and generates a *response* packet to all directly connected hosts and networks. **in.routed** waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables to allow possible unstable situations to settle.

In addition to processing incoming packets, **in.routed** also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

In addition to the facilities described above, **in.routed** supports the notion of "distant" **passive** and **active** gateways. When **in.routed** is started up, it reads the file **gateways** to find gateways which may not be identified using the **SIOCGIFCONF ioctl**. Gateways specified in this manner should be marked **passive** if they are not expected to exchange routing information, while gateways marked **active** should be willing to exchange routing information (that is, they should have a **in.routed** process running on the machine). Passive gateways are maintained in the routing tables forever. Information regarding their existence is not included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of time, the associated route is deleted.

The **gateways** is comprised of a series of lines, each in the following format:

< **net** | **host** > *filename1* **gateway** *filename2* **metric** *value* < **passive** | **active** >

The **net** or **host** keyword indicates if the route is to a network or specific host.

filename1 is the name of the destination network or host. This may be a symbolic name located in **networks** or **hosts**, or an Internet address specified in "dot" notation; see **inet**(3N).

filename2 is the name or address of the gateway to which messages should be forwarded.

value is a metric indicating the hop count to the destination host or network.

The keyword **passive** or **active** indicates if the gateway should be treated as passive or active (as described above).

OPTIONS

–g Is used on internetwork routers to offer a route to the ''default'' destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

–q Is the opposite of the –**s** option.

–s Forces **in.routed** to supply routing information whether it is acting as an internetwork router or not.

–S If **in.routed** is not acting as an internetwork router it will, instead of entering the whole routing table in the kernel, only enter a default route for each internetwork router. This reduces the the memory requirements without losing any routing reliability.

–t All packets sent or received are printed on standard output. In addition, **in.routed** will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process. Any other argument supplied is interpreted as the name of the file in which **in.routed**'s actions should be logged. This log contains information about any changes to the routing tables and a history of recent messages sent and received which are related to the changed route.

–v Allows a logfile (whose name must be supplied) to be created showing the changes made to the routing tables with a timestamp.

FILES

/etc/gateways	for distant gateways
/etc/networks	associations of Internet Protocol network numbers with network names
/etc/hosts	Internet host table

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ioctl(2), **attributes**(5), **inet**(7P), **udp**(7P)

NOTES

The kernel's routing tables may not correspond to those of **in.routed** for short periods of time while processes that utilize existing routes exit; the only remedy for this is to place the routing process in the kernel.

in.routed should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information.

NAME

in.rshd, rshd – remote shell server

SYNOPSIS

in.rshd *host.port*

DESCRIPTION

in.rshd is the server for the **rsh**(1) program. The server provides remote execution facilities with authentication based on privileged port numbers.

in.rshd is invoked by **inetd**(1M) each time a shell service is requested, and executes the following protocol:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection. The client's host address (in hex) and port number (in decimal) are the arguments passed to **in.rshd**.

2) The server reads characters from the socket up to a null (**\0**) byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see **hosts**(4)), the server aborts the connection. Please refer to the **SECURITY** section below for more details.

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the *server*'s machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the *client*'s machine.

7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) **in.rshd** then validates the user according to the following steps. The remote user name is looked up in the password file and a **chdir** is performed to the user's home directory. If the lookup fails, the connection is terminated. If the **chdir** fails, it does a **chdir** to **/** (root). If the user is not the super-user, (user ID 0), and if the pam_rhosts_auth PAM module is configured for authentication, the file **/etc/hosts.equiv** is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. See **SECURITY** below for a discussion of PAM authentication. If the lookup fails, or the user is the super-user, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated

9) A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. (The PATH variable is

set to **/usr/bin**.) The shell inherits the network connections established by **in.rshd**.

SECURITY

in.rshd uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **in.rshd**. Here is a partial **pam.conf** file with entries for the **rsh** command using rhosts authentication, UNIX account management, and session management module.

```
rsh    auth       required     /usr/lib/security/pam_rhosts_auth.so.1
rsh    account    required     /usr/lib/security/pam_unix.so.1
rsh    session    required     /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **rsh** service, then the entries for the "other" service will be used. To maintain the authentication requirement for **in.rshd**, the rsh entry must always be configured with the pam_rhosts_auth.so.1 module. Multiple authentication modules can not be listed for the **rsh** service.

FILES

/etc/hosts.equiv

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

rsh(1), **inetd**(1M), **pam**(3), **hosts**(4), **pam.conf**(4), **attributes**(5), **pam_rhosts_auth**(5), **pam_unix**(5)

DIAGNOSTICS

The following diagnostic messages are returned on the connection associated with **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 in step 9 above (0 is returned above upon successful completion of all the steps prior to the command execution).

locuser too long
The name of the user on the client's machine is longer than 16 characters.

remuser too long
The name of the user on the remote machine is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Login incorrect.
No password file entry for the user name existed.

Permission denied.
The authentication procedure described above failed.

Can't make pipe.

The pipe needed for the **stderr** was not created.

Try again.
A *fork* by the server failed.

NOTES

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an “open” environment.

A facility to allow all data exchanges to be encrypted should be present.

NAME

in.rwhod, rwhod – system status server

SYNOPSIS

/usr/sbin/in.rwhod [**–m** [*ttl*]]

DESCRIPTION

in.rwhod is the server which maintains the database used by the **rwho**(1) and **ruptime**(1) programs. Its operation is predicated on the ability to broadcast or multicast messages on a network.

in.rwhod operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast or multicast on a network. As a consumer of information, it listens for other **in.rwhod** servers' status messages, validating them, then recording them in a collection of files located in the directory **/var/spool/rwho**.

The **rwho** server transmits and receives messages at the port indicated in the **rwho** service specification, see **services**(4). The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];     /* tty name */
        char    out_name[8];     /* user id */
        long    out_time;        /* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the **w**(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the **uname**(2) system call. The array at the end of the message contains information about the users who are logged in to the sending machine. This information includes the contents of the **utmp**(4) entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the **rwho** server are discarded unless they originated at a **rwho** server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **in.rwhod** are placed in files named **whod.***hostname* in the directory **/var/spool/rwho**. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 60 seconds. **in.rwhod** performs an **nlist**(3E) on **/dev/ksyms** every 10 minutes to guard against the possibility that this file is not the system image currently operating.

OPTIONS

–m [*ttl*] Use the rwho IP multicast address (224.0.1.3) when transmitting. Receive announcements both on this multicast address and on the IP broadcast address. If *ttl* is not specified **in.rwhod** will multicast on all interfaces but with the IP TimeToLive set to 1 (that is, packets will not be forwarded by multicast routers.) If *ttl* is specified **in.rwhod** will only transmit packets on one interface and setting the IP TimeToLive to the specified *ttl*.

FILES

/dev/ksyms system namelist

/var/spool/rwho/rwhod/.∗
information about other machines

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ruptime(1), **rwho**(1), **w**(1), **uname**(2), **nlist**(3E), **services**(4), **utmp**(4), **attributes**(5)

WARNINGS

This service can cause network performance problems when used by several hosts on the network. It is not run at most sites by default. If used, include the multicast option (**-m**).

NOTES

This service takes up progressively more network bandwidth as the number of hosts on the local net increases. For large networks, the cost becomes prohibitive.

in.rwhod should relay status information between networks. People often interpret the server dying as a machine going down.

NAME

install – install commands

SYNOPSIS

/usr/sbin/install –c *dira* [**–m** *mode*] [**–u** *user*] [**–g** *group*] [**–o**] [**–s**] *file*
/usr/sbin/install –f *dirb* [**–m** *mode*] [**–u** *user*] [**–g** *group*] [**–o**] [**–s**] *file*
/usr/sbin/install –n *dirc* [**–m** *mode*] [**–u** *user*] [**–g** *group*] [**–o**] [**–s**] *file*
/usr/sbin/install –d | –i [**–m** *mode*] [**–u** *user*] [**–g** *group*] [**–o**] [**–s**] *dirx* . . .
/usr/sbin/install [**–m** *mode*] [**–u** *user*] [**–g** *group*] [**–o**] [**–s**] *file* [*dirx* . . .]

DESCRIPTION

install is most commonly used in ‘‘makefiles’’ (see **make**(1S)) to install a *file* in specific locations, or to create directories within a file system. Each *file* is installed by copying it into the appropriate directory.

install uses no special privileges to copy files from one place to another. The implications of this are:

- You must have permission to read the files to be installed.
- You must have permission to copy into the destination directory.
- You must have permission to change the modes on the final copy of the file if you want to use the **–m** option.
- You must be super-user if you want to specify the ownership of the installed file with the **–u** or **–g** options. If you are not the super-user, the installed file will be owned by you, regardless of who owns the original.

install prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx* . . .) are given, **install** searches a set of default directories (**/bin**, **/usr/bin**, **/etc**, **/lib**, and **/usr/lib**, in that order) for a file with the same name as *file*. When the first occurrence is found, **install** issues a message saying that it is overwriting that file with *file*, and proceeds to do so. If the file is not found, the program states this and exits.

If one or more directories (*dirx* . . .) are specified after *file*, those directories are searched before the default directories.

OPTIONS

–c *dira* Install *file* in the directory specified by *dira*, if *file* does not yet exist. If it is found, **install** issues a message saying that the file already exists, and exits without overwriting it.

–f *dirb* Force *file* to be installed in given directory, even if the file already exists. If the file being installed does not already exist, the mode and owner of the new file will be set to **755** and **bin ,** respectively. If the file already exists, the mode and owner will be that of the already existing file.

–n *dirc* If *file* is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file will be set to **755** and **bin**, respectively.

–d Create a directory. Missing parent directories are created as required as in **mkdir –p**. If the directory already exists, the owner, group and mode will be set to the values given on the command line.

−i Ignore default directory list, searching only through the given directories (*dirx* . . .).

−m *mode* The mode of the new file is set to *mode*. Set to **0755** by default.

−u *user* The owner of the new file is set to *user*. Only available to the super-user. Set to **bin** by default.

−g *group* The group id of the new file is set to *group*. Only available to the super-user. Set to **bin** by default.

−o If *file* is found, save the ‘‘found’’ file by copying it to **OLD***file* in the directory in which it was found. This option is useful when installing a frequently used file such as **/bin/sh** or **/lib/saf/ttymon**, where the existing file cannot be removed.

−s Suppress printing of messages other than error messages.

USAGE

See **largefile**(5) for the description of the behavior of **install** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

chgrp(1), **chmod**(1), **chown**(1), **cp**(1), **make**(1S), **mkdir**(1), **chown**(1M), **attributes**(5), **largefile**(5)

NAME

installboot – install bootblocks in a disk partition

SYNOPSIS

SPARC

installboot *bootblk raw-disk-device*

x86

installboot *pboot bootblk raw-disk-device*

DESCRIPTION

The **boot**(1M) program, **ufsboot**, is loaded from disk by the bootblock program which resides in the boot area of a disk partition.

The **ufs** boot objects are platform-dependent, and reside in the **/usr/platform/***platform-name***/lib/fs/ufs** directory. The platform name can be found using the **–i** option of **uname**(1).

OPERANDS

bootblk The name of the bootblock code.

raw-disk-device The name of the disk device onto which the bootblock code is to be installed; it must be a character device which is readable and writable. Naming conventions for a SCSI or IPI drive are **c***?***t***?***d***?***s***?* and **c***?***d***?***s***?* for an IDE drive.

pboot The name of the partition boot file.

EXAMPLES

SPARC

To install a **ufs** bootblock on slice **0** of target **0** on controller **1** of the platform where the command is being run, use:

```
example# installboot /usr/platform/‘uname –i‘/lib/fs/ufs/bootblk \
        /dev/rdsk/c1t0d0s0
```

x86

To install the **ufs** bootblock and partition boot program on slice **2** of target **0** on controller **1** of the platform where the command is being run, use:

```
example# installboot /usr/platform/‘uname –i‘/lib/fs/ufs/pboot \
        /usr/platform/‘uname –i‘/lib/fs/ufs/bootblk /dev/rdsk/c1t0d0s2
```

FILES

/usr/platform/*platform-name***/lib/fs/ufs** directory where **ufs** boot objects reside.

/platform/*platform-name***/ufsboot** second level program to boot from a disk or CD

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

od(1), **uname**(1), **boot**(1M), **init**(1M), **kadb**(1M), **kernel**(1M), **reboot**(1M), **rpc.bootparamd**(1M), **init.d**(4), **attributes**(5)

Solaris Advanced Installation Guide

SPARC

monitor(1M)

x86

fdisk(1M), **fmthard**(1M)

WARNINGS

The **installboot** utility will fail if the *bootblk*, *pboot* or *openfirmware* files do not exist or if the raw disk device is not a character device.

NAME

installf – add a file to the software installation database

SYNOPSIS

installf [–**c** *class*] [[–**M**] –**R** *root_path*] [–**V** *fs_file*] *pkginst pathname*
[*ftype* [*major minor*] [*mode owner group*]]

installf [–**c** *class*] [[–**M**] –**R** *root_path*] [–**V** *fs_file*] *pkginst*

installf –**f** [–**c** *class*] [[–**M**] –**R** *root_path*] [–**V** *fs_file*] *pkginst*

DESCRIPTION

installf informs the system that a pathname not listed in the **pkgmap**(4) file is being created or modified. It should be invoked before any file modifications have occurred.

When the second synopsis is used, the pathname descriptions will be read from standard input. These descriptions are the same as would be given in the first synopsis but the information is given in the form of a list. The descriptions should be in the form:

pathname [*ftype* [*major minor*] [*mode owner group*]]

After all files have been appropriately created and / or modified, **installf** should be invoked with the –**f** synopsis to indicate that installation is final. Links will be created at this time and, if attribute information for a pathname was not specified during the original invocation of **installf**, or was not already stored on the system, the current attribute values for the pathname will be stored. Otherwise, **installf** verifies that attribute values match those given on the command line, making corrections as necessary. In all cases, the current content information is calculated and stored appropriately.

OPTIONS

–**c** *class*
Class to which installed objects should be associated. Default class is **none**.

–**f**
Indicates that installation is complete. This option is used with the final invocation of **installf** (for all files of a given class).

–**M**
Instruct **installf** not to use the **$***root_path***/etc/vfstab** file for determining the client’s mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

–**R** *root_path*
Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, **/export/root/client1**).

–**V** *fs_file*
Specify an alternative *fs_file* to map the client’s file systems. For example, used in situations where the **$***root_path***/etc/vfstab** file is nonexistent or unreliable.

OPERANDS

pkginst
Name of package instance with which the pathname should be associated.

pathname
Pathname that is being created or modified.

ftype
A one-character field that indicates the file type. Possible file types

include:

b	block special device
c	character special device
d	directory
e	a file to be edited upon installation or removal
f	a standard executable or data file
l	linked file
p	named pipe
s	symbolic link
v	volatile file (one whose contents are expected to change)
x	an exclusive directory

major The major device number. The field is only specified for block or character special devices.

minor The minor device number. The field is only specified for block or character special devices.

mode The octal mode of the file (for example, 0664). A question mark (**?**) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

owner The owner of the file (for example, **bin** or **root**). The field is limited to 14 characters in length. A question mark (**?**) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

group The group to which the file belongs (for example, **bin** or **sys**). The field is limited to 14 characters in length. A question mark (**?**) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

EXAMPLES

The following example shows the use of **installf**, invoked from an optional pre-install or post-install script:

```
#create /dev/xt directory
#(needs to be done before drvinstall)
installf $PKGINST /dev/xt d 755 root sys | |
          exit 2
majno=‘/usr/sbin/drvinstall –m /etc/master.d/xt
   –d $BASEDIR/data/xt.o –v1.0‘ | |
          exit 2
i=00
while [ $i –lt $limit ]
do
```

```
    for j in 0 1 2 3 4 5 6 7
    do
        echo /dev/xt$i$j c $majno ‘expr $i * 8 + $j‘
            644 root sys |
        echo /dev/xt$i$j=/dev/xt/$i$j
    done
    i=‘expr $i + 1‘
    [ $i –le 9 ] && i="0$i" #add leading zero
done | installf $PKGINST – | | exit 2
# finalized installation, create links
installf –f $PKGINST | | exit 2
```

EXIT STATUS

0 Successful operation.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pkginfo(1), **pkgmk**(1), **pkgparam**(1), **pkgproto**(1), **pkgtrans**(1), **pkgadd**(1M), **pkgask**(1M), **pkgchk**(1M), **pkgrm**(1M), **removef**(1M), **pkgmap**(4), **space**(4), **attributes**(5)

Application Packaging Developer’s Guide

NOTES

When *ftype* is specified, all applicable fields, as shown below, must be defined:

ftype	Required Fields
p, **x**, **d**, **f**, **v**, or **e**	**mode owner group**
c or **b**	**major minor mode owner group**

The **installf** command will create directories, named pipes and special devices on the original invocation. Links are created when **installf** is invoked with the **–f** option to indicate installation is complete.

Links should be specified as *path1***=***path2*. *path1* indicates the destination and *path2* indicates the source file.

Files installed with **installf** will be placed in the class **none**, unless a class is defined with the command. Subsequently, they will be removed when the associated package is deleted. If this file should not be deleted at the same time as the package, be certain to assign it to a class which is ignored at removal time. If special action is required for the file before removal, a class must be defined with the command and an appropriate class action script delivered with the package.

When classes are used, **installf** must be used in one of the following forms:

```
installf –c class1 . . .
installf –f –c class1 . . .
installf –c class2 . . .
installf –f –c class2 . . .
```

NAME

install_scripts, add_install_client, rm_install_client, setup_install_server, check – scripts used to install the Solaris software

SYNOPSIS

cdrom-mnt-pt/**Solaris_2.6/Tools/add_install_client** [**–i** *IP_address*]
[**–e** *Ethernet_address*] [**–s** *server_name*:*path*] [**–c** *server_name*:*path*]
[**–n** [*server*]: *name_service*[(*netmask*)]] [**–p** *server_name*:*path*]
host_name platform_group

cdrom-mnt-pt/**Solaris_2.6/Tools/rm_install_client** *host_name*

cdrom-mnt-pt/**Solaris_2.6/Tools/setup_install_server** [**–b**] *install_dir_path*

cdrom-mnt-pt/**Solaris_2.6/Tools/jumpstart_sample/check** [**–p** *install_dir_path*]
[**–r** *rulesfile*]

DESCRIPTION

These commands are located on slice 0 of the Solaris CD. (If the Solaris CD has been copied to a local disk, *cdrom_mnt_pt* is the path to the copied Solaris CD.) They can be used for a variety of installation tasks. Specifically:

- Use **add_install_client** and **rm_install_client** to add or remove clients for network installation (these commands update the **bootparams**(4) file). The **add_install_client** command must be run from the install server's Solaris installation image (a mounted Solaris CD or a Solaris CD copied to disk) or the boot server's boot directory (if a boot server is required). The Solaris installation image or the boot directory must be the same Solaris release that you want installed on the client.
- Use **setup_install_server** to copy the Solaris CD to a disk (to set up an install server) or to copy just the boot software of the Solaris CD to a disk (to set up a boot server). An install server is required to install clients over the network. A boot server is also required for network installations if the install server and clients to be installed are on different subnets (the boot server must be located on the client's subnet).
- Use **check** to validate the rules in a **rules** file (this is only necessary if a custom JumpStart installation is being set up).

OPTIONS

The following options are supported:

add_install_client

–i *IP_address* Specify the IP address of the client to be installed.

–e *Ethernet_address* Specify the Ethernet address of the system to be installed.

–s *server_name*:*path* This option is required only when using **add_install_client** from a boot server. Specify the name of the server and the absolute path of the Solaris installation image that will be used for this installation. *path* is either the path to a mounted Solaris CD or a path to a directory with a copy of the Solaris CD.

–c *server_name* **:** *path* This option is required only to specify a JumpStart directory for a custom JumpStart installation. *server_name* is the host name of the server with a JumpStart directory. *path* is the absolute path to the JumpStart directory.

–n [*server*]**:** *name_service*[(*netmask*)]
This option specifies which name service should be used during system configuration. This sets the **ns** keyword in the **bootparams**(4) file.

name_service Valid entries are **nis**, **nisplus**, and **none**.

server The name of the server or IP address of the specified name service. If the server specified is on a different subnet, then the *netmask* may be needed to enable the client to contact the server.

netmask A series of four numbers separated by periods, specifying which portion of an IP address is the network part, and which is the host part.

–p *server_name* **:** *path* This option is the location of the user-defined **sysidcfg** file for preconfiguring system / network information. *server_name* is either a valid host name or IP address. *path* is the absolute path to the file.

setup_install_server

–b This option sets up the server only as a boot server.

check

–p *install_dir_path* Validates the **rules** file by using the **check** script from a specified Solaris installation image, instead of the **check** script from the system you are using. *install_dir_path* is the path to a Solaris installation image on a local disk or a mounted Solaris CD.

Use this option to run the most recent version of **check** if your system is running a previous version of Solaris.

–r *rulesfile* Specifies a rules file other than the one named **rules**. Using this option, the validity of a rule can be tested before integrating it into the rules file. **check** will report whether or not the rule is valid, but it will not create the **rules.ok** file necessary for a custom JumpStart installation.

OPERANDS

The following operands are supported:

add_install_client

host_name This is the name of the client to be installed.

platform_group Vendor-defined grouping of hardware platforms for the purpose of distributing specific software. Examples of valid platform groups are:

System	Platform Name
x86	**i86pc**
SPARCstation 1+	**sun4c**
SPARCstation 5	**sun4m**

Use the **uname**(1) command (option −**m**) to determine a system's platform group.

rm_install_client *host_name* This is the name of the client to be removed.

setup_install_server *install_dir_path* The absolute path of the directory in which the Solaris software is to be copied. The directory must be empty.

EXAMPLES

The following **add_install_client** commands add clients for network installation from a mounted Solaris CD on an install server:

```
example# cd /cdrom/cdrom0/s0/Solaris_2.6/Tools
example# ./add_install_client system_1 sun4c
example# ./add_install_client system_2 sun4m
```

The following **setup_install_server** command copies the mounted Solaris CD to a directory named **/export/install** on the local disk:

```
example# cd /cdrom/cdrom0/s0/Solaris_2.6/Tools
example# ./setup_install_server /export/install
```

The following **add_install_client** commands add clients for network installation from a mounted Solaris CD on an install server. The −**c** option specifies a server and path to a JumpStart directory that has a rules file and a profile file for performing a custom JumpStart installation. Also, the Solaris CD has been copied to the **/export/install** directory.

```
example# cd /export/install/Solaris_2.6/Tools
example# ./add_install_client −c install_server:/jumpstart system_1 i86pc
example# ./add_install_client −c install_server: /jumpstart system_2 i86pc
```

The following **rm_install_client** commands remove clients for network installation:

```
example# cd /export/install/Solaris_2.6/Tools
example# ./rm_install_client holmes
example# ./rm_install_client watson
```

The following **setup_install_server** command copies the boot software of a mounted Solaris CD to a directory named **/boot_dir** on a system that is going to be a boot server for a subnet:

```
example# cd /cdrom/cdrom0/s0/Solaris_2.6/Tools
example# ./setup_install_server –b /boot_dir
```

The following **check** command validates the syntax of the rules file used for a custom JumpStart installation:

```
example# cd jumpstart_dir_path
example# ./check –p /cdrom/cdrom0/s0
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.
1 An error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	Solaris CD

SEE ALSO

uname(1), **bootparams**(4), **attributes**(5)

Solaris Advanced Installation Guide

NAME

in.talkd, talkd – server for talk program

SYNOPSIS

in.talkd

DESCRIPTION

talkd is a server used by the **talk**(1) program. It listens at the UDP port indicated in the ‘‘talk’’ service description; see **services**(4). The actual conversation takes place on a TCP connection that is established by negotiation between the two machines involved.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

talk(1), **inetd**(1M), **services**(4), **attributes**(5)

NOTES

The protocol is architecture dependent.

NAME

in.telnetd, telnetd – DARPA TELNET protocol server

SYNOPSIS

/usr/sbin/in.telnetd

DESCRIPTION

in.telnetd is a server that supports the DARPA standard TELNET virtual terminal protocol. **in.telnetd** is invoked in the internet server (see **inetd**(1M)), normally for requests to connect to the TELNET port as indicated by the **/etc/services** file (see **services**(4)).

in.telnetd operates by allocating a pseudo-terminal device for a client, then creating a login process which has the slave side of the pseudo-terminal as its standard input, output, and error. **in.telnetd** manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session starts up, **in.telnetd** sends TELNET options to the client side indicating a willingness to do *remote echo* of characters, and to *suppress go ahead*. The pseudo-terminal allocated to the client is configured to operate in “cooked” mode, and with **XTABS**, **ICRNL**, and **ONLCR** enabled (see **termio**(7I)).

in.telnetd is willing to do: *echo*, *binary*, *suppress go ahead*, and *timing mark*. **in.telnetd** is willing to have the remote client do: *binary*, *terminal type*, *terminal size*, *logout option*, and *suppress go ahead*.

in.telnetd also allows environment variables to be passed, provided that the client negotiates this during the initial option negotiation. The DISPLAY environment variable may be sent this way, either by the TELNET general environment passing methods, or via the XDISPLOC TELNET option. DISPLAY can be passed in the environment option during the same negotiation where XDISPLOC is used. Note that if you use both methods, use the same value for both. Otherwise, the results may be unpredictable.

These options are specified in Internet standards RFC 1096, RFC 1408, RFC 1571, and RFC 1572.

The banner printed by **in.telnetd** is configurable. The default is (more or less) equivalent to "‘uname -sr‘" and will be used if no banner is set in **/etc/default/telnetd**. To set the banner, add a line of the form

```
BANNER="..."
```

to **/etc/default/telnetd**. Nonempty banner strings are fed to shells for evaluation. The default banner may be obtained by

```
BANNER="\\r\\n\\r\\n‘uname -s‘ ‘uname -r‘\\r\\n\\r\\n"
```

and no banner will be printed if **/etc/default/telnetd** contains

```
BANNER=""
```

SECURITY

in.telnetd uses **pam**(3) for authentication, account management, session management, and password management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **in.telnetd**. Here is a partial **pam.conf** file with entries for the **telnet** command using the UNIX authentication, account management, session management, and password management modules.

```
telnet   auth       required   /usr/lib/security/pam_unix.so.1
telnet   account    required   /usr/lib/security/pam_unix.so.1
telnet   session    required   /usr/lib/security/pam_unix.so.1
telnet   password   required   /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **telnet** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords.

FILES

/etc/default/telnetd

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

telnet(1), **inetd**(1M), **pam**(3), **pam.conf**(4), **services**(4), **attributes**(5), **pam_unix**(5), **termio**(7I)

Alexander, S., ‘‘*TELNET Environment Option,*’’ RFC 1572, Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Borman, Dave, ‘‘*TELNET Environment Option,*’’ RFC 1408, Network Information Center, SRI International, Menlo Park, Calif., January 1993.

Borman, Dave, ‘‘*TELNET Environment Option Interoperability Issues,*’’ RFC 1571, Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Crispin, Mark, ‘‘*TELNET Logout Option,*’’ RFC 727, Network Information Center, SRI International, Menlo Park, Calif., April 1977.

Marcy, G., ‘‘*TELNET X Display Location Option,*’’ RFC 1096, Network Information Center, SRI International, Menlo Park, Calif., March 1989.

Postel, Jon, and Joyce Reynolds, ‘‘*TELNET Protocol Specification,*’’ RFC 854, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Waitzman, D., ‘‘*TELNET Window Size Option,*’’ RFC 1073, Network Information Center, SRI International, Menlo Park, Calif., October 1988.

NOTES

Some TELNET commands are only partially implemented.

Binary mode has no common interpretation except between similar operating systems

The terminal type name received from the remote client is converted to lower case.

The *packet* interface to the pseudo-terminal should be used for more intelligent flushing of input and output queues.

in.telnetd never sends TELNET *go ahead* commands.

NAME

in.tftpd, tftpd – Internet Trivial File Transfer Protocol server

SYNOPSIS

in.tftpd [**–s**] [*homedir*]

DESCRIPTION

tftpd is a server that supports the Internet Trivial File Transfer Protocol (TFTP). This server is normally started by **inetd**(1M) and operates at the port indicated in the **tftp** Internet service description in the **/etc/inetd.conf** file. By default, the entry for **in.tftpd** in **etc/inetd.conf** is commented out. To make **in.tftpd** operational, the comment character(s) must be deleted from the file. See **inetd.conf**(4).

Before responding to a request, the server attempts to change its current directory to *homedir*; the default directory is **/tftpboot**.

The use of **tftp** does not require an account or password on the remote system. Due to the lack of authentication information, **in.tftpd** will allow only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable. Note that this extends the concept of "public" to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling this service.

in.tftpd runs with the user ID and group ID set to **[GU]ID_NOBODY** under the assumption that no files exist with that owner or group. However, nothing checks this assumption or enforces this restriction.

OPTIONS

–s Secure. When specified, the directory change to *homedir* must succeed. The daemon also changes its root directory to *homedir*.

FILES

/etc/inetd.conf

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

tftp(1), **inetd**(1M), **netconfig**(4), **attributes**(5)

Sollins, K.R., *The TFTP Protocol (Revision 2)*, RFC 783, Network Information Center, SRI International, Menlo Park, Calif., June 1981.

NAME

in.tnamed, tnamed – DARPA trivial name server

SYNOPSIS

/usr/sbin/in.tnamed [**–v**]

DESCRIPTION

in.tnamed is a server that supports the DARPA Name Server Protocol. The name server operates at the port indicated in the "name" service description (see **services**(4)), and is invoked by **inetd**(1M) when a request is made to the name server.

OPTIONS

–v   Invoke the daemon in verbose mode.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

uucp(1C), **inetd**(1M), **services**(4), **attributes**(5)

Postel, Jon, *Internet Name Server*, IEN 116, SRI International, Menlo Park, California, August 1979.

NOTES

The protocol implemented by this program is obsolete. Its use should be phased out in favor of the Internet Domain Name Service (DNS) protocol.

NAME

in.uucpd, uucpd – UUCP server

SYNOPSIS

/usr/sbin/in.uucpd [–**n**]

DESCRIPTION

in.uucpd is the server for supporting UUCP connections over networks.

in.uucpd is invoked by **inetd**(1M) when a UUCP connection is established (that is, a connection to the port indicated in the “uucp” service specification) and executes the following protocol (see **services**(4)):

1) The server prompts with **login:**. The **uucico**(1M) process at the other end must supply a username.

2) Unless the username refers to an account without a password, the server then prompts with **Password:**. The **uucico** process at the other end must supply the password for that account.

If the username is not valid, or is valid but refers to an account that does not have **/usr/lib/uucp/uucico** as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, **uucico** is run, with the user ID, group ID, group set, and home directory for that account, with the environment variables **USER** and **LOGNAME** set to the specified username, and with a –**u** flag specifying the username. Unless the –**n** flag is specified, entries are made in **/var/adm/utmp, /var/adm/wtmp,** and **/var/adm/lastlog** for the username. **in.uucpd** must be invoked by a user with appropriate privilege (usually root) in order to be able to verify that the password is correct.

SECURITY

in.uucpd uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **in.uucpd**. Here is a partial **pam.conf** file with entries for **uucp** using the UNIX authentication, account management, and session management module.

```
uucp    auth       required     /usr/lib/security/pam_unix.so.1
uucp    account    required     /usr/lib/security/pam_unix.so.1
uucp    session    required     /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **uucp** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the peer may be prompted for multiple passwords.

FILES

/var/adm/utmp	accounting
/var/adm/wtmp	accounting
/var/adm/lastlog	time of last login

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

SEE ALSO

inetd(1M), **uucico**(1M), **pam**(3), **pam.conf**(4), **services**(4), **attributes**(5), **pam_unix**(5)

DIAGNOSTICS

All diagnostic messages are returned on the connection, after which the connection is closed.

user read
An error occurred while reading the username.

passwd read
An error occurred while reading the password.

Login incorrect.
The username is invalid or refers to an account with a login shell other than **/usr/lib/uucp/uucico**, or the password is not the correct password for the account.

NAME

iostat – report I/O statistics

SYNOPSIS

/usr/bin/iostat [**–cdDeEIMnpPtx**] [**–l** *n*] [*disk* ...] [*interval* [*count*]]

DESCRIPTION

The **iostat** utility iteratively reports terminal, disk, and tape I/O activity, as well as CPU utilization. The first line of output is for all time since boot; each subsequent line is for the prior interval only.

To compute this information, the kernel maintains a number of counters. For each disk, the kernel counts reads, writes, bytes read, and bytes written. The kernel also takes hi-res time stamps at queue entry and exit points, which allows it to keep track of the residence time and cumulative residence-length product for each queue. Using these values, **iostat** produces highly accurate measures of throughput, utilization, queue lengths, transaction rates and service time. For terminals collectively, the kernel simply counts the number of input and output characters.

For more general system statistics, use **sar**(1), **sar**(1M), or **vmstat**(1M).

See *Solaris 1.x to 2.x Transition Guide* for device naming conventions for disks.

OPTIONS

The **iostat** utility's activity class options default to **tdc** (terminal, disk, and CPU). If any activity class options are specified, the default is completely overridden. Therefore, if only **–d** is specified, neither terminal nor CPU statistics will be reported. The last disk option specified (**–d**, **–D**, or **–x**) is the only one that is used.

–c Report the percentage of time the system has spent in user mode, in system mode, waiting for I/O, and idling.

–d For each disk, report the number of kilobytes transferred per second, the number of transfers per second, and the average service time in milliseconds.

–D For each disk, report the reads per second, writes per second, and percentage disk utilization.

–e Display device error summary statistics. The total errors, hard errors, soft errors and transport errors are displayed.

–E Display all device error statistics.

–I Report the counts in each interval, rather than rates (where applicable).

–l *n* Limit the number of disks included in the report to *n*; the disk limit defaults to 4 for **–d** and **–D**, and unlimited for **–x**. Note: disks explicitly requested (see *disk* below) are not subject to this disk limit.

–M Display data throughput in MB/sec instead of KB/sec.

–n Display device names in cXtYdZ format.

–p For each disk, report per-partition statistics in addition to per-device statistics.

–P For each disk, report per-partition statistics only, no per-device statistics.

–t Report the number of characters read and written to terminals per second.

–x For each disk, report extended disk statistics. The output is in tabular form.

OPERANDS

The following operands are supported:

disk Explicitly specify the disks to be reported; in addition to any explicit disks, any active disks up to the disk limit (see **–l** above) will also be reported.

count Only print *count* reports.

interval Report once each *interval* seconds.

EXAMPLES

```
example% iostat -xtc 5 2
                                   extended disk statistics         tty            cpu
disk    r/s  w/s   Kr/s   Kw/s  wait  actv  svc_t   %w    %b   tin  tout  us  sy  wt  id
sd0     6.2  0.0   21.5    0.0   0.0   0.1   24.1    0    15     0    84   4  94   2   0
sd1     1.8  0.0   14.3    0.0   0.0   0.1   41.6    0     7
sd2     0.0  0.0    0.0    0.0   0.0   0.0    0.0    0     0
sd3     5.6  0.2   25.7    0.2   0.0   0.1   22.5    0    13
                                   extended disk statistics         tty            cpu
disk    r/s  w/s   Kr/s   Kw/s  wait  actv  svc_t   %w    %b   tin  tout  us  sy  wt  id
sd0     2.6  3.0   20.7   22.7   0.1   0.2   59.2    6    19     0    84   3  85  11   0
sd1     4.2  1.0   33.5    8.0   0.0   0.2   47.2    2    23
sd2     0.0  0.0    0.0    0.0   0.0   0.0    0.0    0     0
sd3    10.2  1.6   51.4   12.8   0.1   0.3   31.2    3    31
example%
```

The fields have the following meanings:

disk name of the disk
r/s reads per second
w/s writes per second
Kr/s kilobytes read per second
Kw/s kilobytes written per second
wait average number of transactions waiting for service (queue length)
actv average number of transactions actively being serviced (removed from the queue but not yet completed)
svc_t average service time, in milliseconds
%w percent of time there are transactions waiting for service (queue non-empty)
%b percent of time the disk is busy (transactions in progress)

```
example% iostat -xnp
                                      extended disk statistics
disk          r/s  w/s  Kr/s  Kw/s  wait  actv  svc_t   %w   %b
c0t0d0        0.3  0.0   2.3   0.1   0.0   0.0    6.2    0    0
c0t0d0s2      0.0  0.0   0.0   0.0   0.0   0.0    0.0    0    0
c0t0d0s6      0.1  0.0   0.7   0.0   0.0   0.0    6.3    0    0
c0t0d0s7      0.2  0.0   1.6   0.1   0.0   0.0    6.1    0    0
example%
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

sar(1), **sar**(1M), **vmstat**(1M), **attributes**(5)

Solaris 1.x to 2.x Transition Guide
System Administration Guide

NAME

kadb – a kernel debugger

SYNOPSIS

SPARC

ok boot *device_specifier* **kadb** [**–d**] [*boot-flags*]

> b kadb [**–d**] [*boot-flags*]

x86

select (b)oot or (i)nterpreter: b kadb [**–d**] [*b-flags*]

select (b)oot or (i)nterpreter: i kadb [**–d**] [*b-flags*]

DESCRIPTION

The utility **kadb** is an interactive debugger with a user interface similar to that of **adb**(1), the assembly language debugger. **kadb** must be loaded prior to the standalone program it is to debug. **kadb** runs with the same environment as the standalone program, so it shares many resources with that program. The debugger is cognizant of and able to control multiple processors, should they be present in a system.

Unlike **adb**(1), **kadb** runs in the same supervisor virtual address space as the program being debugged, although it maintains a separate context. The debugger runs as a *co-process* that cannot be killed (no '**:k**') or rerun (no '**:r**'). There is no signal control (no '**:i**', '**:t**', or '**$i**'), although the keyboard facilities (CTRL–C, CTRL–S, and CTRL–Q) are simulated.

In the case of the UNIX system kernel, the keyboard abort sequence will suspend kernel operations and break into the debugger. This behavior may be disabled via the **kbd**(1) command and may not be the current default on all systems. See **kb**(7M) for more information.

On a SPARC based system, the keyboard abort sequence is **L1-A** for the console, and BREAK for a serial line. On an x86 based system, the keyboard abort sequence is CTRL-ALT-D for the console. **kadb** gives the prompt:

kadb[#]

where # is the CPU it is currently executing on.

Since the kernel is composed of the core image file (**kernel/unix**) and the set of loadable modules already in memory, **kadb** has the capability of debugging all of these by traversing special data structures. **kadb** makes use of this feature to allow it to reference any kernel data structure, even if it resides in a loadable module. **kadb** will set the **'–d'** flag by default so the program being debugged can tell it is being watched. If this flag is not given, **kadb** will load and immediately run the default kernel .

On a SPARC based system, *device-specifier* specifies the device from which to load. See **monitor**(1M).

On an x86 based system, select boot (**b kadb**) or interpreter (**i kadb**).

Most **adb**(1) commands function in **kadb** as expected. As with '**adb –k**', **$p** works when debugging kernels. The verbs **?** and **/** are equivalent in **kadb**, since there is only one address space in use.

Additional Commands

kadb has the following additional commands. Note: for the general syntax of **adb** commands, see **adb**(1).

[Like **:e** in **adb**(1), but requires only one keystroke and no RETURN character.

] Like **:s** in **adb**(1), but requires only one keystroke and no RETURN character.

Additional x86 Commands

:a Sets a hardware access (read or write) breakpoint using the processor hardware facilities. The syntax and action for this is the same as **:b** in **adb**(1), with the following exceptions:

- The breakpoint will trigger if any bytes from the breakpoint for length bytes are being accessed. See **$l** below for setting the length of a hardware breakpoint.
- Breakpoints should be aligned for the length specified. Any address is valid for length 1. Addresses divisible by 2 should be used for length 2 (short). Addresses divisible by 4 should be used for length 4 (long).
- For an access to be detected, it must complete the instruction that caused the access. This means that this breakpoint will stop after the access, not before.
- There are a limited number (4) of hardware breakpoint registers and, when set, this will use one.
- Since this breakpoint does not modify memory locations, this will work on locations that are not in core at the time the breakpoint is set.

:w Like **:a** above, but will only breakpoint on a write to the specified address.

:p Like **:a** above, but will only breakpoint when an instruction at the specified address is run. Note that the **$l** operation has no effect on this type of breakpoint. This breakpoint will occur before the instruction is executed.

$l Sets the default data length for an access or write breakpoint. The syntax for this command is **[length]$l**. The length can be set to **1** for byte, **2** for short, and **4** for long word accesses. If length is left off, byte will be assumed. Once set, this value will affect any newly set access or write breakpoints, but will not affect ones set before this operation.

$b Has two additional columns that **adb**(1) does not have. The first is the **type** column which indicates **soft** for a normal breakpoint, **access** for an access hardware breakpoint, **write** for a write hardware breakpoint, and **inst** for an instruction hardware breakpoint. The second is the **len** column which for access and write breakpoints indicate the length of the operation to break on.

:i Will input a byte for display from the address-specified I/O port. For example, **330:i** will input from address port **330**.

:o Will output a byte to the address-specified I/O port. **[address],[data]:o** will output data byte to address I/O port. For example **330,80:o** will output **80** to address port **330**.

Additional UltraSparc Commands

<N>:x Switch the active CPU to <N>. Thereafter, commands such as **$r** and **$c** will display the registers and stack of the new CPU.

OPTIONS

The following options are supported:

–d Interactive startup. If this flag is given, **kadb** will stop after it is loaded and display the prompt as **kadb:** and the name of the default program to debug. For example:

kadb: kernel/unix

The user may either press RETURN to use the default, or BACK SPACE and type in the name of another program to debug.

boot-flags
You can specify boot flags as arguments when invoking **kadb**. Boot flags are passed to the program being debugged. See the manual page **boot**(1M) for relevant boot flags.

USAGE

Kernel Macros

As with **adb**(1), macros can be used with **kadb**, but they cannot be read from a file at run-time. The **kadb** command **$M** lists all the macros built into **kadb**.

FILES

/platform/*platform-name***/kadb**	primary debugger path
/platform/*hardware-class-name***/kadb**	alternative debugger path
/platform/*platform-name***/kernel/unix**	primary default kernel
/platform/*hardware-class-name***/kernel/unix**	alternative default kernel

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcar

SEE ALSO

adb(1), **uname**(1), **boot**(1M), **attributes**(5), **kb**(7M)

SPARC Only

kbd(1), **kernel**(1M), **monitor**(1M), **obpsym**(1M)

DIAGNOSTICS

When there is no current command or format, **kadb** comments about syntax errors, abnormal termination of commands, and the like.

WARNINGS

SPARC Only

On a SPARC based system, **kadb** cannot reliably single-step over instructions which change the processor status register.

NOTES

platform-name can be found using the **−i** option of **uname**(1). *hardware-class-name* can be found using the **−m** option of **uname**(1).

NAME

kdmconfig – configure or unconfigure keyboard, display, and mouse options for OpenWindows and internationalization

SYNOPSIS

kdmconfig

kdmconfig [**–fv**] [**–s** *hostname*] **–c** | **–t** | **–u** | **–d** *filename*

DESCRIPTION

The **kdmconfig** program configures or unconfigures the **/etc/openwin/server/etc/OWconfig** file with the keyboard, display, and mouse information relevant to a client's machine on x86 based systems for Solaris software. **kdmconfig** can also be used to set up the *display*, *pointer*, and *keyboard* entries in the **bootparams**(4) database on a server machine or the *monitor*, *keyboard*, *display*, and *pointer* keywords in a **sysidcfg**(4) file. **kdmconfig** can only be run by the super-user. Upon completion of device selection, **kdmconfig** prompts the user to test the configuration, which is done by running the window system.

The **kdmconfig** program is normally run during installation and upon reboot, but it can also be run from the command line after the system has been installed. When configuring a client during an initial installation or a reconfigure reboot, the **sysidconfig**(1M) program will invoke **kdmconfig** with the **–c** option, and when the user executes the **sys-unconfig**(1M) program, **kdmconfig** will be executed with the **–u** option. Similarly, when you run **kdmconfig** from the command line, use the **–u** option to unconfigure the existing OpenWindows configuration. You can then rerun **kdmconfig** with the **–cf** options to create a new OpenWindows configuration. To edit the existing configuration, run **kdmconfig** from the command line without options. After each reboot, **kdmconfig** will be invoked by the system with the **–t** (test mode) option to ensure autoconfiguration capability and identify possible conflicts between the current configuration and the one recorded in the **OWconfig** file.

OPTIONS

The valid options are:

–c Run the program in the configuration mode. This mode is used to create or update the **OWconfig** file. When invoked in this way, **kdmconfig** first looks for the relevant configuration information in the **bootparams**(4) databases. It also takes into account the information returned from device probes, unless the **–s** option is also used. The **bootparams**(4) databases available to the client are all of the **/etc/bootparams** files on servers on the same subnet as the client, provided the server machine is running the **bootparamd**(1M) daemon. **kdmconfig** is invoked with the **–c** option when called by **sysidconfig**(1M)**.**

–d *filename* Set up a **sysidcfg**(4) file. This option displays the same screens as the **–c** option, but the information you specify is saved as **sysidcfg**(4) keywords (*monitor*, *keyboard*, *display*, and *pointer*). This enables you to use a **sysidcfg**(4) file to preconfigure a system's device information and bypass **kdmconfig** during an installation.

filename is the **sysidcfg**(4) file that is created, and it is created in the directory where **kdmconfig** is being run unless a path is specified. If filename already exists in the specified directory, the keywords are appended to the existing file.

–f Force screens mode. When this option is invoked, no network probing will be performed. This is helpful when debugging the client's configuration environment. Note that the –**s** option implies the use of –**f**, bypassing network probing when setting up a server.

–s *hostname* Set up the **bootparams**(4) database on this machine for the specified client. This option presents the same screens as it does when run on a client, but instead writes the resulting information to the **/etc/bootparams** file. Also, –**s** implies the use of the –**f** option. That is, the program will always present the screens to the user when invoked this way. This option will reconfigure the **nsswitch.conf**(4) file to look for a **bootparams**(4) database on a local server. This option is only available to the super-user.

–t Run the program in test mode. In this mode, **kdmconfig** will use device probe information to determine whether the **OWconfig** file contains complete and up-to-date information about the keyboard, display, and mouse. If the information is accurate, **kdmconfig** will exit silently. Otherwise, **kdmconfig** will prompt for the super-user password and proceed to a normal editing session (as though it had been run without options).

–u Unconfigure the system, returning it to an "out-of-the-box" state. In this state, the factory default keyboard, mouse, and display are selected as a result of removing the device configuration entries from the **/etc/openwin/server/etc/OWconfig** file. This may result in an unusable configuration for the display server.

–v Enable verbose mode. Normally, **kdmconfig** will not produce any output. This option is helpful for debugging, as it records the different actions taken by **kdmconfig** on **stderr**.

No Options

Run without options, **kdmconfig** is used to edit the current configuration. **kdmconfig** uses the information from the **OWconfig** file in addition to information obtained from the **bootparams**(4) file and from device probes. In other respects, it is similar to using the –**c** option of **kdmconfig**.

FILES

/etc/openwin/server/etc/OWconfig	OpenWindows configuration file
/etc/bootparams	contains list of clients that diskless clients use for booting
/etc/nsswitch.conf	name service configuration file

x86 Only **/dev/openprom** installed devices and properties

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWos86r

SEE ALSO

bootparamd(1M), **sys-unconfig**(1M), **sysidconfig**(1M), **bootparams**(4), **nsswitch.conf**(4), **sysidcfg**(4), **attributes**(5)

NAME

kerbd – generates and validates Kerberos tickets for kernel RPC

SYNOPSIS

/usr/sbin/kerbd [**–dg**]

DESCRIPTION

kerbd is the usermode daemon which interfaces between kernel RPC and the Kerberos key distribution center (KDC) for the purposes of generating and validating Kerberos authentication tickets. In addition, **kerbd** maps Kerberos user names into local user and group ids. By default, all groups that the requested user belongs to will be included in the grouplist credential. **kerbd** is automatically started when the system enters the multi-user state.

OPTIONS

–d Run in debug mode. **kerbd** will output various information about Kerberos tickets being processed.

–g Do not initialize the grouplist in the user credential when mapped from Kerberos' principal name. If this option is selected, only each user's group from the **passwd** entry will be included in mapped credentials.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

kdestroy(1), **kerberos**(1), **kinit**(1), **krb.conf**(4), **attributes**(5)

NAME

kernel – UNIX system executable file containing basic operating system services

SYNOPSIS

/platform/*platform-name***/kernel/unix** [–**afrsv**]
/kernel/genunix

DESCRIPTION

The operating system image, or kernel, is the collection of software made up of the core image file (**unix**) and all of the modules loaded at any instant in time. The system will not function without a kernel to control it.

The kernel is loaded by the **boot**(1M) command in a machine-specific way. The kernel may be loaded from disk or CD-ROM ("diskfull boot") or over the network ("diskless boot"). In either case, the directories under **/platform** and **/kernel** must be readable and must contain executable code which is able to perform the required kernel service. If the –**a** flag is given, the user is able to supply different pathnames for the default locations of the kernel and modules.

The **’moddir’** variable contains a colon-separated list of directories that the kernel searches for modules. **’moddir’** can be set in the **/etc/system** file; it defaults to **/platform/***platform-name***/kernel:/platform/***hardware-class-name***/kernel:/kernel:/usr/kernel**. (where *platform-name* can be found using the –**i** option of **uname**(1), and *hardware-class-name* can be found using the –**m** option of **uname**(1)).

On x86 based systems the value of **’moddir’** defaults to **/platform/***platform-name***/kernel:/kernel:/usr/kernel** (where *platform-name* is the name of the platform implementation).

The kernel configuration can be controlled using the **/etc/system** file (see **system**(4)).

/kernel/genunix is the platform-independent component of the base kernel.

OPTIONS

–**a** Ask the user for configuration information, such as where to find the system file, where to mount **root**, and even override the name of the kernel itself. Default responses will be contained in square brackets ([]), and the user may simply enter RETURN to use the default response (note that RETURN is labeled ENTER on some keyboards). **/dev/null** may be entered at the prompt which asks for the pathname of the system configuration file. See **system**(4).

–**f** Causes Autoclient systems to **flush** and reinitialize the client system’s local cache. This flag is ignored for all non-Autoclient systems.

–**r** Reconfiguration boot. The system will probe all attached hardware devices and assign nodes in the file system to represent only those devices actually found. It will also configure the logical namespace in **/dev** as well as the physical namespace in **/devices**. See **add_drv**(1M) and **rem_drv**(1M) for additional information about maintaining device drivers.

–**s** Boot only to init level ’s’. See **init**(1M).

–**v** Boot with verbose messages enabled. If this flag is not given, the messages are still printed, but the output is directed to the system logfile. See **syslogd**(1M).

EXAMPLES

SPARC

To boot in single-user interactive mode, respond to the **ok** prompt with one of the following:

boot –as

boot kernel/unix –as

boot disk3 kernel/unix –as

x86

To boot in single-user interactive mode, respond to the **>** prompt one of the following:

b –as

b kernel/unix –as

FILES

/kernel Contains kernel components common to all platforms within a particular instruction set that are needed for booting the system. of the core image file.

/platform/*platform-name***/kernel**
The platform-specific kernel components.

/platform/*hardware-class-name***/kernel**
The kernel components specific to this hardware class.

/usr/kernel Contains kernel components common to all platforms within a particular instruction set.

Each of the directories listed above can potentially contain the following subdirectories:

drv	Loadable device drivers
exec	The modules that execute programs stored in various file formats.
fs	File system modules
misc	Miscellaneous system-related modules
sched	Operating system schedulers
strmod	System V STREAMS loadable modules
sys	Loadable system calls

x86 Only

mach x86 hardware support

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcar

SEE ALSO

uname(1), **add_drv**(1M), **boot**(1M), **init**(1M), **kadb**(1M), **rem_drv**(1M), **savecore**(1M), **syslogd**(1M), **system**(4), **attributes**(5)

SPARC Only

monitor(1M)

DIAGNOSTICS

The kernel gives various warnings and error messages. If the kernel detects an unrecoverable fault, it will panic or halt.

BUGS

Bugs in the kernel often result in kernel panics.

Reconfiguration boot does not currently remove filesystem entries for devices that have been physically removed from the system.

NAME

keyserv – server for storing private encryption keys

SYNOPSIS

keyserv [**−d**] [**−D**] [**−n**]

DESCRIPTION

keyserv is a daemon that is used for storing the private encryption keys of each user logged into the system. These encryption keys are used for accessing secure network services such as secure NFS and NIS+.

Normally, root's key is read from the file **/etc/.rootkey** when the daemon is started. This is useful during power-fail reboots when no one is around to type a password.

OPTIONS

−d Disable the use of default keys for **nobody**.

−D Run in debugging mode and log all requests to **keyserv**.

−n Root's secret key is not read from **/etc/.rootkey**. Instead, **keyserv** prompts the user for the password to decrypt root's key stored in the **publickey** database and then stores the decrypted key in **/etc/.rootkey** for future use. This option is useful if the **/etc/.rootkey** file ever gets out of date or corrupted.

FILES

/etc/.rootkey

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

keylogin(1), **keylogout**(1), **publickey**(4), **attributes**(5)

NAME

killall – kill all active processes

SYNOPSIS

/usr/sbin/killall [*signal*]

DESCRIPTION

killall is used by **shutdown**(1M) to kill all active processes not directly related to the shutdown procedure.

killall terminates all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

killall sends *signal* (see **kill**(1)) to the active processes. If no *signal* is specified, a default of **15** is used.

The **killall** command can be run only by the super-user.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWcsu

SEE ALSO

kill(1), **ps**(1), **fuser**(1M), **shutdown**(1M), **signal**(3C), **attributes**(5)

NAME

labelit – list or provide labels for file systems

SYNOPSIS

labelit [**–F** *FSType*] [**–V**] *special* [*operands*]

DESCRIPTION

The **labelit** utility is used to write or display labels on unmounted disk file systems.

OPTIONS

The following options are supported:

–F *FSType* Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from **/etc/vfstab** by matching the *special* with an entry in the table. If no matching entry is found, the default file system type specified in **/etc/default/fs** will be used.

–V Echo complete command line. This option may be used to verify and validate the command line. Additional information obtained using a **/etc/vfstab** lookup is included in the output. The command is not executed.

OPERANDS

The following operands are supported. If no operands are specified, **labelit** will display the value of the labels.

special The disk partition (for example, **/dev/rdsk/c0t3d0s6**). The device may not be on a remote machine.

operands *FSType*-specific operands. Consult the manual page of the *FSType*-specific **labelit** command for detailed descriptions.

USAGE

See **largefile**(5) for the description of the behavior of **labelit** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

The following exit values are returned:

0	Write or display of labels was successful.
non-zero	An error occurred.

FILES

/etc/vfstab list of default parameters for each file system

/etc/default/fs default local file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: **LOCAL=ufs**

LOCAL The default partition for a command if no *FSType* is specified.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

volcopy(1M), **vfstab**(4), **attributes**(5), **largefile**(5)
Manual pages for the *FSType*-specific modules of **labelit**

NOTES

This utility may not be supported for all *FSTypes*.

NAME

labelit_hsfs – provide and print labels for hsfs file systems

SYNOPSIS

/usr/sbin/labelit –F hsfs [*generic_options*] [**–o** *specific_options*] *special*

DESCRIPTION

labelit can be used to provide labels for unmounted CD-ROM images (CD-ROMs may not be labeled, as they are read-only media).

generic_options are options supported by the generic **labelit** command.

If no *specific_options* are specified, **labelit** prints the current value of all label fields.

The *special* name should be the physical disk section (for example, **/dev/dsk/c0d0s6**).

OPTIONS

–o Use one or more of the following *name=value* pairs separated by commas (with no intervening spaces) to specify values for specific label fields. According to the ISO 9660 specification, only certain sets of characters may be used to fill in these labels. Thus, ‘‘d-characters’’ below refers to the characters ‘A’ through ‘Z’, the digits ‘0’ through ‘9’, and the ‘_’ (underscore) character. ‘‘a-characters’’ below refers to ‘A’ through ‘Z’, ‘0’ through ‘9’, space, and the following characters: !"%&’()∗+,-. / :;<=>?_.

absfile= Abstract file identifier, d-characters, 37 characters maximum.

applid= Application identifier, d-characters, 128 characters maximum.

bibfile= Bibliographic file identifier, d-characters, 37 characters maximum.

copyfile= Copyright file identifier, d-characters, 128 maximum.

prepid= Data preparer identifier, d-characters, 128 maximum.

pubid= Publisher identifier, d-characters, 128 maximum.

sysid= System identifier, a-characters, 32 maximum.

volid= Volume identifier, d-characters, 32 maximum.

volsetid= Volume set identifier, d-characters, 128 maximum.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

labelit(1M), **volcopy**(1M), **attributes**(5)

NAME

labelit_ufs – provide and print labels for ufs file systems

SYNOPSIS

labelit –F ufs [*generic_options*] *special* [*fsname volume*]

DESCRIPTION

labelit is used to write labels on unmounted disk file systems. Such labels may be used to uniquely identify volumes and are used by volume-oriented programs such as **volcopy**(1M).

OPTIONS

The following option is supported:

generic_options options supported by the generic **labelit** command. See **labelit**(1M).

OPERANDS

The following operands are supported:

special name should be the physical disk section (for example, **/dev/dsk/c0d0s6**). The device may not be on a remote machine.

fsname represents the mount point (for example, **root**, **u1**, and so on) of the file system.

volume may be used to represent the physical volume name.

If *fsname* and *volume* are not specified, **labelit** prints the current values of these labels. Both *fsname* and *volume* are limited to six or fewer characters.

EXIT STATUS

The following exit values are returned:

0 Write or display of labels was successful.

non-zero An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

labelit(1M), **volcopy**(1M), **fs_ufs**(4), **attributes**(5)

NAME

leoconfig – initialize the Leo (ZX and TZX) Graphics Accelerator and download microcode

SYNOPSIS

/usr/sbin/leoconfig [**–c**] [**–h**] [**–i**] [**–v**] [**–d** *device_filename*] [**–f** *filename*]
[**–g** *gamma_value*] [**–I** *microcode_directory*] [**–l** *leofloat_enable*]
[[**–degamma8 on** | **off**] | [**–DEGAMMA8 on** | **off**]] [**–m** *monitor_type*]
/usr/sbin/leoconfig **–G** *gamma_value* [**–v**]
/usr/sbin/leoconfig **–M** *monitor_type* [**–v**]

DESCRIPTION

The **leoconfig** command initializes the Leo (both ZX and TZX) Graphics Accelerator and downloads microcode from the host. **leoconfig** is normally run as a part of the **/etc/init.d/leoconfig** script to download the Leo microcode file and to complete Leo initialization. The standard version of the file **/etc/init.d/leoconfig** runs **leoconfig** with the following command line options:

```
/usr/sbin/leoconfig ${MONTYPE} –d $devpath/$inst –I /usr/lib
```

If another configuration is desired, edit the leoconfig script in the **/etc/init.d/leoconfig** file.

It is inadvisable to run the **leoconfig** command while the configured device is being used; unpredictable results may occur. To run **leoconfig**, first bring the window system down.

Only **root** can run this command.

File names may be either relative or absolute path names. Relative path names are prepended with the path specified by **–I**, or the default path **/usr/lib**.

OPTIONS

The following options are supported:

–c Checks that the microcode has loaded correctly.

–d *device_filename*
Specifies the Leo special file. The default is **/dev/fbs/leo0**.

–degamma8 on | **off**
Specifies automatic inverse gamma correction of 8-bit indexed color maps. This allows color maps with built-in gamma correction to work properly on Leo. The default is **on**.

–DEGAMMA8 on | **off**
Like the **–degamma8** option but only specifies automatic inverse gamma correction; does not otherwise initialize Leo or download microcode.

–f *filename* Specifies the microcode file. The default is **leo.ucode**.

–g *gamma_value*
Specifies the gamma correction value. The default is **2.22**.

–G *gamma_value*
Loads the gamma correction table only; does not initialize Leo or download microcode. Any other option besides **–v** will cause an invalid option message to be displayed.

–h Display help menu of valid command arguments.

–i Initialize the Leo system.

–I *microcode_directory*
Specifies the directory containing the microcode files. The default is **/usr/lib**.

–l *leofloat_enable*
Specifies the LeoFloat enable mask in decimal form (one bit per float). The default for model ZX is **15**. The default for model TZX is **255**.

–m *monitor_type*
Specifies the monitor type, where *monitor_type* must be one of the following:

1280_76	1280 × 1024 @ 76 Hz, non-interlaced.
1280_67	1280 × 1024 @ 67 Hz, non-interlaced.
1152_76	1152 × 900 @ 76 Hz, non-interlaced.
1152_66	1152 × 900 @ 66 Hz, non-interlaced.
1024_76	1024 × 768 @ 76 Hz, non-interlaced.
1024_60	1024 × 768 @ 60 Hz, non-interlaced.
stereo_108	960 × 680 @ 108 Hz, non-interlaced, stereo.
stereo_114	960 × 680 @ 112 Hz, non-interlaced, stereo.
pal	770 × 575 @ 50 Hz, interlaced (PAL).
ntsc	640 × 480 @ 60 Hz, interlaced (NTSC).

–M *monitor_type*
Specifies the monitor type only; does not initialize the Leo or download microcode. Any other option besides –**v** will cause an invalid option message to be displayed. The *monitor_type* value is the same as for the –**m** option.

–v Verbose. Display detailed progress reports.

EXAMPLES

The following example switches the monitor type to the maximum resolution of 1280 × 1024 at 76 Hz:

```
example% /usr/sbin/leoconfig –M 1280_76 –v
```

EXIT STATUS

0 success

1 an error has occurred

FILES

/dev/fbs/leo0	device special file
leo.ucode	Leo microcode file
/usr/lib	directory that normally contains microcode file
/etc/init.d/leoconfig	local Leo initialization script

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWleow

SEE ALSO

mmap(2), **attributes**(5), **fbio**(7I), **leo**(7D)

NAME

link, unlink – link and unlink files and directories

SYNOPSIS

/usr/sbin/link *file1 file2*
/usr/sbin/unlink *file*

DESCRIPTION

The **link** command is used to create a file name that points to another file. *file1* is the existing file, *file2* is the newly-created file. Linked files and directories can be removed by the **unlink** command; however, it is strongly recommended that the **rm**(1) and **rmdir**(1) commands be used instead of the **ln**(1) command.

The only difference between **ln** and **link** and **unlink** is that the latter do exactly what they are told to do, abandoning all error checking. This is because they directly invoke the **link** and **unlink** system calls.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ln(1), **rm**(1), **link**(2), **unlink**(2), **attributes**(5)

NOTES

These commands can be run only by the super-user.

NAME

list_devices – list allocatable devices

SYNOPSIS

list_devices [–**s**] [–**U uid**] –**l** [*device*]

list_devices [–**s**] [–**U uid**] –**n** [*device*]

list_devices [–**s**] [–**U uid**] –**u** [*device*]

DESCRIPTION

list_devices lists the allocatable devices in the system according to specified qualifications.

The *device* and all device special files associated with the device are listed. The device argument is optional and if it is not present, all relevant devices are listed.

OPTIONS

–**l** [*device*]
List the pathname(s) of the device special files associated with the device that are allocatable to the current process. If *device* is given, list only the files associated with the specified device.

–**n** [*device*]
List the pathname(s) of device special files associated with the device that are allocatable to the current process but are not currently allocated. If *device* is given, list only the files associated with that device.

–**s** Silent. Suppresses any diagnostic output.

–**u** [*device*]
List the pathname(s) of device special files, associated with the device that are allocated to the owner of the current process. If *device* is given, list only the files associated with that device.

–**U** *uid* Use the user ID *uid* instead of the real user ID of the current process when performing the **list_devices** operation. Only the super user can use this option.

DIAGNOSTICS

list_devices returns an nonzero exit status in the event of an error.

FILES

/etc/security/device_allocate
/etc/security/device_maps
/etc/security/dev/∗
/usr/security/lib/∗

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

allocate(1M), **bsmconv**(1M), **deallocate**(1M), **device_allocate**(4), **device_maps**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

listdgrp – lists members of a device group

SYNOPSIS

/usr/bin/listdgrp *dgroup* **...**

DESCRIPTION

listdgrp displays the members of the device groups specified by the *dgroup* list.

ERRORS

This command will exit with one of the following values:

0 = successful completion of the task.

1 = command syntax incorrect, invalid option used, or internal error occurred.

2 = device group table could not be opened for reading.

3 = device group *dgroup* could not be found in the device group table.

EXAMPLES

To list the devices that belong to group **partitions**:

```
example% listdgrp partitions
    root
    swap
    usr
```

FILES

/etc/dgroup.tab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

putdgrp(1M), **attributes**(5)

NAME

listen – network listener daemon

SYNOPSIS

/usr/lib/saf/listen [–**m** *devstem*] *net_spec*

DESCRIPTION

The **listen** process ‘‘listens’’ to a network for service requests, accepts requests when they arrive, and invokes servers in response to those service requests. The network listener process may be used with any connection-oriented network (more precisely, with any connection-oriented transport provider) that conforms to the Transport Layer Interface (TLI) Specification.

The listener internally generates a pathname for the minor device for each connection; it is this pathname that is used in the **utmp** entry for a service, if one is created. By default, this pathname is the concatenation of the prefix **/dev/***netspec* with the decimal representation of the minor device number. In either case, the representation of the minor device number will be at least two digits (for example, 05 or 27), or longer when it is necessary to accommodate minor device numbers larger than 99.

SERVER INVOCATION

When a connection indication is received, the listener creates a new transport endpoint and accepts the connection on that endpoint. Before giving the file descriptor for this new connection to the server, any designated STREAMS modules are pushed and the configuration script is executed, (if one exists). This file descriptor is appropriate for use with either TLI (see **t_sync**(3N)) or the sockets interface library.

By default, a new instance of the server is invoked for each connection. When the server is invoked, file descriptor 0 refers to the transport endpoint, and is open for reading and writing. File descriptors 1 and 2 are copies of file descriptor 0; no other file descriptors are open. The service is invoked with the user and group IDs of the user name under which the service was registered with the listener, and with the current directory set to the HOME directory of that user.

Alternatively, a service may be registered so that the listener will pass connections to a standing server process through a FIFO or a named STREAM, instead of invoking the server anew for each connection. In this case, the connection is passed in the form of a file descriptor that refers to the new transport endpoint. Before the file descriptor is sent to the server, the listener interprets any configuration script registered for that service using **doconfig**(3N), although **doconfig** is invoked with both the NORUN and NOASSIGN flags. The server receives the file descriptor for the connection in a **strrecvfd** structure using an I_RECVFD **ioctl**(2).

For more details about the listener and its administration, see **nlsadmin**(1M).

OPTIONS

–**m***devstem* The listener will use *devstem* as the prefix for the pathname.

FILES

/etc/saf/*pmtag***/***

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

nlsadmin(1M), **pmadm**(1M), **sac**(1M), **sacadm**(1M), **ioctl**(2), **doconfig**(3N), **nlsgetcall**(3N), **nlsprovider**(3N), **t_sync**(3N), **attributes**(5), **streamio**(7I)

System Administration Guide

NOTES

When passing a connection to a standing server, the user and group IDs contained in the **strrecvfd** structure will be those for the listener (that is, they will both be 0); the user name under which the service was registered with the listener is not reflected in these IDs.

When operating multiple instances of the listener on a single transport provider, there is a potential race condition in the binding of addresses during initialization of the listeners, if any of their services have dynamically assigned addresses. This condition would appear as an inability of the listener to bind a static-address service to its otherwise valid address, and would result from a dynamic-address service having been bound to that address by a different instance of the listener.

NAME

lockd – network lock daemon

SYNOPSIS

/usr/lib/nfs/lockd [**–g** *graceperiod*] [**–t** *timeout*] [*nthreads*]

DESCRIPTION

The **lockd** utility is part of the NFS lock manager, which supports record locking operations on NFS files. See **fcntl**(2) and **lockf**(3C). The lock manager provides two functions:

- it forwards **fcntl**(2) locking requests for NFS mounted file systems to the lock manager on the NFS server
- it generates local file locking operations in response to requests forwarded from lock managers running on NFS client machines.

State information kept by the lock manager about these locking requests can be lost if the **lockd** is killed or the operating system is rebooted. Some of this information can be recovered as follows. When the server lock manager restarts, it waits for a grace period for all client-site lock managers to submit reclaim requests. Client-site lock managers, on the other hand, are notified by the status monitor daemon, **statd**(1M), of the restart and promptly resubmit previously granted lock requests. If the lock daemon fails to secure a previously granted lock at the server site, then it sends **SIGLOST** to a process.

OPTIONS

–g *graceperiod* Specify the number of seconds that clients have to reclaim locks after the server reboots. The default is 45 seconds.

–t *timeout* Specify the number of seconds to wait before retransmitting a lock request to the remote server. The default value is 15 seconds.

nthreads Specify the maximum number of concurrent threads that the server can handle. This concurrency is achieved by up to *nthreads* threads created as needed in the kernel. *nthreads* should be based on the load expected on this server. If *nthreads* is not specified, the maximum number of concurrent threads will default to 20.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

statd(1M), **fcntl**(2), **lockf**(3C), **attributes**(5)

NAME

lockfs – change or report file system locks

SYNOPSIS

/usr/sbin/lockfs [–**adehnuw**] [–**c** *string*] [*file-system* . . .]

DESCRIPTION

lockfs is used to change and report the status of file system locks. **lockfs** reports the lock status and unlocks the file systems that were improperly left locked by an application such as **ufsdump**(1M). This could occur if **ufsdump**(1M) is killed using **kill**(1).

Using **lockfs** to lock a file system is discouraged because this requires extensive knowledge of SunOS internals to be used effectively and correctly.

When invoked with no arguments, **lockfs** lists the UFS file systems that are locked. If *file-system* is not specified, and –**a** is specified, **lockfs** is run on all mounted, UFS type file systems.

OPTIONS

You must be super-user to use any of the following options, with the exception of –**a**.

–**a** Apply command to all mounted, UFS type file systems. *file-system* is ignored when –**a** is specified.

–**c** *string* Accept a string that is passed as the comment field. The –**c** only takes affect when the lock is being set using the –**d**, –**h**, –**n**, –**u**, or –**w** options.

–**d** delete-lock (dlock) the specified *file-system*. dlock suspends access that could remove directory entries.

–**e** error-lock (elock) the specified *file-system*. elock blocks all local access to the locked file system and returns **EWOULDBLOCK** on all remote access. File systems are elocked by UFS on detection of internal inconsistency. They may only be unlocked after succesful repair by **fsck**, which is usually done automatically (see **mount_ufs**(1M)). elocked file systems can be unmounted.

–**h** Hard-lock (hlock) the specified *file-system*. hlock returns an error on every access to the locked file system, and cannot be unlocked. hlocked file systems can be unmounted.

–**n** Name-lock (nlock) the specified *file-system*. nlock suspends accesses that could change or remove existing directories entries.

–**u** Unlock (ulock) the specified *file-system*. ulock awakens suspended accesses.

–**w** Write-lock (wlock) the specified *file-system*. wlock suspends writes that would modify the file system. Access times are not kept while a file system is write-locked.

OPERANDS

file-system A list of path names separated by white spaces.

USAGE

See **largefile**(5) for the description of the behavior of **lockfs** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

In the following examples, *filesystem* is the pathname of the mounted-on directory (mount point). **Locktype** is one of "*write*," "*name*," "*delete*," "*hard*," or "*unlock*". When enclosed in parenthesis, the lock is being set. **Comment** is a string set by the process that last issued a lock command.

The following example shows the **lockfs** output when only the **–a** option is specified.

```
example# /usr/sbin/lockfs –a
Filesystem    Locktype    Comment
/             unlock
/var          unlock
example#
```

The following example shows the **lockfs** output when the **–w** option is used to write lock the **/var** file system and the comment string is set using the **–c** option. The **–a** option is then specified on a separate command line.

```
example# /usr/sbin/lockfs –w –c "lockfs: write lock example" /var
example# /usr/sbin/lockfs –a
Filesystem    Locktype    Comment
/             unlock
/var          write       lockfs: write lock example
example#
```

The following example shows the **lockfs** output when the **–u** option is used to unlock the **/var** file system and the comment string is set using the **–c** option.

```
example# /usr/sbin/lockfs –uc "lockfs: unlock example" /var
example# /usr/sbin/lockfs /var
Filesystem    Locktype    Comment
/var          unlock      lockfs: unlock example
example#
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

kill(1), **mount_ufs**(1M), **ufsdump**(1M), **fs_ufs**(4), **attributes**(5), **largefile**(5)

System Administration Guide

DIAGNOSTICS

file system: **Not owner**

You must be root to use this command.

file system **:Deadlock condition detected/avoided**

A file is enabled for accounting or swapping, on *file system*.

file system: **Device busy**

Another process is setting the lock on *file system*.

NAME

lockstat – report kernel lock statistics

SYNOPSIS

lockstat [–**ACEH**] [–**e** *event list*] [–**b** | –**t** | –**h** | –**s** *depth*]
[–**n** *nlocks*] [–**l** *lock*[**,***size*]] [–**d** *duration*] [–**T**]
[–**cwWRp**] [–**D** *count*] [–**o** *filename*] *command* [*args*]

DESCRIPTION

lockstat gathers and displays statistics on kernel synchronization objects. **lockstat** allows you to specify which lock events to watch (for example, spin on adaptive mutex, block on read access to rwlock due to waiting writers, and so forth), how much data to gather for each event, and how to display that data. By default, **lockstat** monitors all lock contention events, gathers frequency and timing data about those events, and displays that data in order of decreasing frequency so that the most heavily contended locks appear first.

lockstat gathers kernel locking statistics until the specified command completes. For example, to gather statistics for a fixed time interval, specify **sleep**(1) as the command, as follows:

```
example# lockstat sleep 5
```

lockstat relies on the **lockstat**(7D) driver, an exclusive-access device that modifies the running kernel's text to intercept events of interest. This imposes a small but measurable overhead on all system activity, so access to the **lockstat**(7D) driver is restricted to super-user by default. The system administrator may relax this restriction by changing the permissions on **/dev/lockstat**.

OPTIONS

Event selection options:

–**C**	Watch contention events [on by default].
–**E**	Watch error events [on by default].
–**H**	Watch hold events [off by default].
–**A**	Watch all events. –**A** is equivalent to –**CEH**.
–**e** *event list*	Only watch the specified events. *event list* is a comma-separated list of events or ranges of events, (for example, 1,4-7,35). Run **lockstat** with no arguments to get a brief description of all events.

Data gathering options (mutually exclusive):

–**b**	Basic statistics: lock, caller, number of events.
–**t**	Timing: Basic plus timing for all events [default].
–**h**	Histogram: Timing plus time distribution histograms.
–**s** *depth*	Stack trace: Histogram plus stack traces of events.

Data filtering options:

–**n** *nlocks*	Maximum number of locks to watch.
–**l** *lock[,size]*	Only watch *lock*, which can be specified as a symbolic name or hex address. *size* defaults to the ELF symbol size or **1** if the symbol size is not available.
–**d** *duration*	Only watch events longer than *duration*.

–T Trace (rather than sample) events [off by default].

Data reporting options:

–c Coalesce lock data for lock arrays (for example, pse_mutex[]).

–w Wherever: distinguish events only by lock, not by caller.

–W Whichever: distinguish events only by caller, not by lock.

–R Display rates (events per second) rather than counts.

–p Parsable output format.

–D *count* Only display the top *count* events of each type.

–o *filename* Direct output to *filename*.

EXAMPLES

Descriptions of the fields from the output of **lockstat** are listed below.

Count or **ops/s** Number of times this lock was acquired by this caller, or the rate (times per second) if –**R** was specified.

indv Percentage of all events represented by this individual event.

cuml Cumulative percentage; a running total of the individuals.

rcnt Average reference count. This will always be **1** for exclusive locks (mutexes, spin locks, rwlocks held as writer) but may be greater than **1** for shared locks (rwlocks held as reader, shared pages, counting semaphores).

spin or **nsec** Average number of times caller spun trying to get the lock, or average duration of the events in nanoseconds, as appropriate for the event.

Lock Address of the lock.

Caller Address of the caller.

The following example uses **lockstat** to gather kernel locking statistics for five seconds:

```
example# lockstat sleep 5
Adaptive mutex spin: 513 events

Count indv cuml rcnt spin Lock             Caller
----- ---- ---- ---- ---- ----------       -------------
   99  19%  19% 1.00  194 0x6335e5f4       cv_timedwait+0xac
   95  19%  38% 1.00  192 0x6335e5f4       nfs_async_start+0x7c
   55  11%  49% 1.00  297 0x6335e5f4       nfs_async_readahead+0xf4
   24   5%  53% 1.00   12 rt_callout_state untimeout+0x24
   19   4%  57% 1.00   11 0x61325e3c       nfs3_readahead+0x3c
   16   3%  60% 1.00   26 0x61478554       nfs3_readahead+0x3c
   15   3%  63% 1.00   28 rt_callout_state realtime_timeout+0xc
   15   3%  66% 1.00  286 0x620f0280       cv_wait_sig_swap+0x1d0
   14   3%  69% 1.00    7 0x61325e3c       nfs3_getapage+0xec
   11   2%  71% 1.00  264 pidlock          thread_exit+0x58
   11   2%  73% 1.00   16 0x61478554       nfs_async_start+0x2d0
   10   2%  75% 1.00   58 fpc_mutex+0x8    page_list_add+0xb8
   10   2%  77% 1.00   13 0x61478554       nfs_async_readahead+0xa8
    9   2%  79% 1.00   42 0x61325e3c       nfs_async_start+0x2d0
    8   2%  80% 1.00   52 fpc_mutex+0x48   page_list_add+0xb8
    7   1%  82% 1.00    7 0x61478554       nfs3_getapage+0xec
    6   1%  83% 1.00    3 0x61325e3c       nfs_async_readahead+0xa8
    6   1%  84% 1.00   63 fpc_mutex+0x48   page_get_mnode_freelist+0xa8
    6   1%  85% 1.00   13 pidlock          cv_wait_sig_swap+0x1d0
    5   1%  86% 1.00    7 0x61478554       nfs3_getapage+0x22c
    5   1%  87% 1.00   87 fpc_mutex+0x28   page_list_add+0xb8
    5   1%  88% 1.00   30 0x61325e3c       nfs3_getapage+0x5a4
    5   1%  89% 1.00  319 0x620f0500       cv_wait_sig_swap+0x1d0
    4   1%  90% 1.00   61 fpc_mutex+0x8    page_get_mnode_freelist+0xa8
...
```

FILES

/dev/lockstat lockstat driver

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5), **lockstat**(7D), **mutex**(9F), **rwlock**(9F)

NOTES

The lockstat technology is provided on an as-is basis. The format and content of **lockstat** output reflect the current Solaris kernel implementation and are therefore subject to change in future releases.

NAME

logins – list user and system login information

SYNOPSIS

/usr/bin/logins [–**admopstux**] [–**g** *group...*] [–**l** *login...*]

DESCRIPTION

This command displays information on user and system logins known to the system. Contents of the output is controlled by the command options and can include the following: user or system login, user id number, **passwd** account field value (user name or other information), primary group name, primary group id, multiple group names, multiple group ids, home directory, login shell, and four password aging parameters. The default information is the following: login id, user id, primary group name, primary group id and the account field value. Output is sorted by user id, system logins, followed by user logins.

OPTIONS

–**a** Add two password expiration fields to the display. The fields show how many days a password can remain unused before it automatically becomes inactive, and the date that the password will expire.

–**d** Selects logins with duplicate uids.

–**m** Displays multiple group membership information.

–**o** Formats output into one line of colon-separated fields.

–**p** Selects logins with no passwords.

–**s** Selects all system logins.

–**t** Sorts output by login instead of by uid.

–**u** Selects all user logins.

–**x** Prints an extended set of information about each selected user. The extended information includes home directory, login shell and password aging information, each displayed on a separate line. The password information consists of password status (PS for password, NP for no password or LK for locked). If the login is passworded, status is followed by the date the password was last changed, the number of days required between changes, and the number of days allowed before a change is required. The password aging information shows the time interval that the user will receive a password expiration warning message (when logging on) before the password expires.

–**g** *group* Selects all users belonging to **group**, sorted by login. Multiple groups can be specified as a comma-separated list.

–**l** *login* Selects the requested login. Multiple logins can be specified as a comma-separated list. Depending on the nameservice lookup types set in **/etc/nsswitch.conf**, the information can come from the **/etc/passwd** and **/etc/shadow** files and other nameservices.

NOTES

Options may be used together. If so, any login that matches any criteria will be displayed. When the –**l** and –**g** options are combined, a user will only be listed once, even if the user belongs to more than one of the selected groups.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5)

NAME

lpadmin – configure the LP print service

SYNOPSIS

lpadmin –**p** *printer options*
lpadmin –**x** *dest*
lpadmin –**d** [*dest*]
lpadmin –**S** *print-wheel* –**A** *alert-type* [–**W** *minutes*] [–**Q** *requests*]
lpadmin –**M** –**f** *form-name* [–**a** [–**o** **filebreak**] [–**t** *tray-number*]]

DESCRIPTION

lpadmin configures the LP print service by defining printers and devices. It is used to add and change printers, to remove printers from service, to set or change the system default destination, to define alerts for printer faults, and to mount print wheels.

OPTIONS

Adding or Changing a Printer

The first form of the **lpadmin** command (**lpadmin** –**p** *printer options*) is used to configure a new printer or to change the configuration of an existing printer. When creating a new printer, one of three options (–**v**, –**U**, or –**s**) must be supplied. In addition, only one of the following may be supplied: –**e**, –**i**, or –**m**; if none of these three options is supplied, the model standard is used. The –**h** and –**l** options are mutually exclusive. Printer and class names may be no longer than 14 characters and must consist entirely of the characters **A-Z**, **a-z**, **0-9**, dash (**-**) and underscore (**_**). If –**s** is specified, the following options are invalid: –**A**, –**e**, –**F**, –**h**, –**i**, –**l**, –**M**, –**m**, –**o**, –**U**, –**v**, and –**W**.

The following *printer options* may appear in any order.

–**A** *alert-type* [–**W** *minutes*]

The –**A** option is used to define an alert that informs the administrator when a printer fault is detected, and periodically thereafter, until the printer fault is cleared by the administrator. The *alert-types* are:

mail Send the alert message using mail (see **mail**(1)) to the administrator.

write Write the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is chosen arbitrarily.

quiet Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the fault has been cleared and printing resumes, messages will again be sent when another fault occurs with the printer.

showfault
Attempt to execute a fault handler on each system that has a print job in the queue. The fault handler is **/etc/lp/alerts/printer**. It is invoked with three parameters: *printer_name*, *date*, *file_name*. The *file_name* is the name of a file containing the fault message.

none Do not send messages; any existing alert definition for the printer will be removed. No alert will be sent when the printer faults until a different alert-type (except **quiet**) is used.

shell-command

Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blank spaces embedded in the command, enclose the command in quotes. Note that the **mail** and **write** values for this option are equivalent to the values **mail** *user-name* and **write** *user-name* respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the **su** command to change to another user ID. If the **su** command has been used to change the user ID, then the *user-name* for the new ID is used.

list Display the type of the alert for the printer fault. No change is made to the alert.

The message sent appears as follows:

> **The printer** *printer* **has stopped printing for the reason given below. Fix the problem and bring the printer back on line. Printing has stopped, but will be restarted in a few minutes; issue an enable command if you want to restart sooner. Unless someone issues a change request**
>
> **lp** –**i** *request-id* –**P ...**
>
> **to change the page list to print, the current request will be reprinted from the beginning.**
>
> **The reason(s) it stopped (multiple reasons indicate reprinted attempts):**
>
> *reason*

The LP print service can detect printer faults only through an adequate fast filter and only when the standard interface program or a suitable customized interface program is used. Furthermore, the level of recovery after a fault depends on the capabilities of the filter.

If the *printer* is **all**, the alerting defined in this command applies to all existing printers.

If the –**W** option is not used to arrange fault alerting for *printer*, the default procedure is to mail one message to the administrator of *printer* per fault. This is equivalent to specifying –**W once** or –**W 0**. If *minutes* is a number greater than zero, an alert will be sent at intervals specified by *minutes*.

–**c** *class*

Insert *printer* into the specified *class*. *class* will be created if it does not already exist.

–**D** *comment*

Save this *comment* for display whenever a user asks for a full description of *printer* (see **lpstat**(1)). The LP print service does not interpret this comment.

–**e** *printer*$_1$

Copy the interface program of an existing *printer*$_1$ to be the interface program for *printer*. (Options –**i** and –**m** may not be specified with this option.)

–F *fault-recovery*

This option specifies the recovery to be used for any print request that is stopped because of a printer fault, according to the value of *fault-recovery*:

continue Continue printing on the top of the page where printing stopped. This requires a filter to wait for the fault to clear before automatically continuing.

beginning Start printing the request again from the beginning.

wait Disable printing on *printer* and wait for the administrator or a user to enable printing again.

During the wait, the administrator or the user who submitted the stopped print request can issue a change request that specifies where printing should resume. (See the **–i** option of the **lp** command.) If no change request is made before printing is enabled, printing resumes at the top of the page where stopped, if the filter allows; otherwise, the request is printed from the beginning.

–f allow:*form-list*

–f deny:*form-list*

Allow or deny the forms in *form-list* to be printed on *printer*. By default no forms are allowed on a new printer.

For each printer, the LP print service keeps two lists of forms: an ‘‘allow-list’’ of forms that may be used with the printer, and a ‘‘deny-list’’ of forms that may not be used with the printer. With the **–f allow** option, the forms listed are added to the allow-list and removed from the deny-list. With the **–f deny** option, the forms listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the forms in the list may be used on the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the forms in the deny-list may not be used with the printer. All forms can be excluded from a printer by specifying **–f deny:all**. All forms can be used on a printer (provided the printer can handle all the characteristics of each form) by specifying **–f allow:all**.

The LP print service uses this information as a set of guidelines for determining where a form can be mounted. Administrators, however, are not restricted from mounting a form on any printer. If mounting a form on a particular printer is in disagreement with the information in the allow-list or deny-list, the administrator is warned but the mount is accepted. Nonetheless, if a user attempts to issue a print or change request for a form and printer combination that is in disagreement with the information, the request is accepted only if the form is currently mounted on the printer. If the form is later unmounted before the request can print, the request is canceled and the user is notified by mail.

If the administrator tries to specify a form as acceptable for use on a printer that doesn’t have the capabilities needed by the form, the command is rejected.

Note the other use of **–f**, with the **–M** option, below.

The –**T** option must be invoked first with **lpadmin** to identify the printer type before the –**f** option can be used.

–**h** Indicate that the device associated with the printer is hardwired. If neither of the mutually exclusive options, –**h** and –**l**, is specified, –**h** is assumed.

–**I** *content-type-list*

Allow *printer* to handle print requests with the content types listed in a *content-type-list*. If the list includes names of more than one type, the names must be separated by commas or blank spaces. (If they are separated by blank spaces, the entire list must be enclosed in double quotes.)

The type **simple** is recognized as the default content type for files in the UNIX system. A **simple** type of file is a data stream containing only printable ASCII characters and the following control characters.

Control Character	Octal Value	Meaning
BACKSPACE	10_8	move back one character, except at beginning of line
TAB	11_8	move to next tab stop
LINEFEED (newline)	12_8	move to beginning of next line
FORMFEED	14_8	move to beginning of next page
RETURN	15_8	move to beginning of current line

To prevent the print service from considering **simple** a valid type for the printer, specify either an explicit value (such as the printer type) in the *content-type-list,* or an empty list. If you do want **simple** included along with other types, you must include **simple** in the *content-type-list*.

Except for **simple**, each *content-type* name is freely determined by the administrator. If the printer type is specified by the –**T** option, then the printer type is implicitly considered to be also a valid content type.

–**i** *interface*

Establish a new interface program for *printer*. *interface* is the pathname of the new program. (The –**e** and –**m** options may not be specified with this option.)

–**l** Indicate that the device associated with *printer* is a login terminal. The LP scheduler (**lpsched**) disables all login terminals automatically each time it is started. (The –**h** option may not be specified with this option.)

–**M** –**f** *form-name* [–**a** [–**o filebreak**]] [–**t** *tray-number*]]

Mount the form *form-name* on *printer.* Print requests that need the pre-printed form *form-name* will be printed on *printer.* If more than one printer has the form mounted and the user has specified **any** (with the –**d** option of the **lp** command) as the printer destination, then the print request will be printed on the one printer that also meets the other needs of the request.

The page length and width, and character and line pitches needed by the form are compared with those allowed for the printer, by checking the capabilities in the **terminfo** database for the type of printer. If the form requires attributes that are not available with the printer, the administrator is warned but the mount is accepted. If the form lists a print wheel as mandatory, but the print wheel mounted on the printer is different, the administrator is also warned but the mount is accepted.

If the −**a** option is given, an alignment pattern is printed, preceded by the same initialization of the physical printer that precedes a normal print request, with one exception: no banner page is printed. Printing is assumed to start at the top of the first page of the form. After the pattern is printed, the administrator can adjust the mounted form in the printer and press return for another alignment pattern (no initialization this time), and can continue printing as many alignment patterns as desired. The administrator can quit the printing of alignment patterns by typing **q**.

If the −**o filebreak** option is given, a formfeed is inserted between each copy of the alignment pattern. By default, the alignment pattern is assumed to correctly fill a form, so no formfeed is added.

If the −**t** *tray-number* option is specified, printer tray *tray-number* will used.

A form is ‘‘unmounted’’ either by mounting a new form in its place or by using the −**f none** option. By default, a new printer has no form mounted.

Note the other use of −**f** without the −**M** option above.

−**M** −**S** *print-wheel*

Mount the *print-wheel* on *printer.* Print requests that need the *print-wheel* will be printed on *printer.* If more than one printer has *print-wheel* mounted and the user has specified **any** (with the −**d** option of the **lp** command) as the printer destination, then the print request will be printed on the one printer that also meets the other needs of the request.

If the *print-wheel* is not listed as acceptable for the printer, the administrator is warned but the mount is accepted. If the printer does not take print wheels, the command is rejected.

A print wheel is ‘‘unmounted’’ either by mounting a new print wheel in its place or by using the option −**S none**. By default, a new printer has no print wheel mounted.

Note the other uses of the −**S** option without the −**M** option described below.

−**m** *model*

Select *model* interface program, provided with the LP print service, for the printer. (Options −**e** and −**i** may not be specified with this option.)

−**o** *option*

The −**o** option defines default printer configuration values given to an interface program. The default may be explicitly overwritten for individual requests by the user (see **lp**(1)), or taken from a preprinted form description (see **lpforms**(1M) and **lp**(1)).

There are several options which are pre-defined by the system. In addition, any number of key-value pairs may be defined. Each of the predefined and undefined options are described.

The Predefined Options

The following options are predefined: adjusting printer capabilities, adjusting printer port characteristics, configuring network printers, and controlling the use of banner.

Adjusting Printer Capabilities

length=*scaled-decimal-number*
width=*scaled-decimal-number*
cpi=*scaled-decimal-number*
lpi=*scaled-decimal-number*

The term *scaled-decimal-number* refers to a non-negative number used to indicate a unit of size. The type of unit is shown by a ‘‘trailing’’ letter attached to the number. Three types of *scaled-decimal-numbers* can be used with the LP print service: numbers that show sizes in centimeters (marked with a trailing **c**); numbers that show sizes in inches (marked with a trailing **i**); and numbers that show sizes in units appropriate to use (without a trailing letter), that is, lines, characters, lines per inch, or characters per inch.

The option values must agree with the capabilities of the type of physical printer, as defined in the terminfo database for the printer type. If they do not, the command is rejected.

The defaults are defined in the **terminfo** entry for the specified printer type. The defaults may be reset by:

lpadmin –**p** *printername* –**o length=**
lpadmin –**p** *printername* –**o width=**
lpadmin –**p** *printername* –**o cpi=**
lpadmin –**p** *printername* –**o lpi=**

Adjusting Printer Port Characteristics

stty="’*stty-option-list*’"

The *stty-option-list* is not checked for allowed values, but is passed directly to the stty program by the standard interface program. Any error messages produced by **stty** when a request is processed (by the standard interface program) are mailed to the user submitting the request.

The default for **stty** is:

```
stty="’9600 cs8 –cstopb –parenb ixon
    –ixany opost –olcuc onlcr –ocrnl –onocr
    –onlret –ofill nl0 cr0 tab0 bs0 vt0 ff0’"
```

The default may be reset by:

lpadmin −**p** *printername* −**o** **stty=**

Configuring Network Printers

dest=*string*
protocol=*string*
bsdctrl=*string*
timeout=*non-negative-integer-seconds*

These four options are provided to support network printing. Each option is passed directly to the interface program; any checking for allowed values is done there.

The value of **dest** is the name of the destination for the network printer; the semantics for value **dest** are dependent on the printer and the configuration. There is no default.

The value of option **protocol** sets the over-the-wire protocol to the printer. The default for option **protocol** is **bsd**. The value of option **bsdctrl** sets the print order of control and data files (BSD protocol only); the default for this option is **control file first**. The value of option **timeout** sets the seed value for backoff time when the printer is busy. The default value for the **timeout** option is **10** seconds. The defaults may be reset by:

lpadmin −**p** *printername* −**o** **protocol=**
lpadmin −**p** *printername* −**o** **bsdctrl=**
lpadmin −**p** *printername* −**o** **timeout=**

Controlling the Use of the Banner Page

nobanner Allow a user to submit a print request specifying that no banner page be printed.

banner Force a banner page to be printed with every print request, even when a user asks for no banner page. This is the default. Specify −**o** **nobanner** to allow users to specify −**o** **nobanner** with the **lp** command.

Undefined Options

*key***=***value* Each *key***=***value* is passed directly to the interface program. Any checking for allowed values is done in the interface program.

Any default values for a given *key***=***value* option are defined in the interface program. If a default is provided, it may be reset by typing the key without any value:

lpadmin −**p** *printername* −**o** *key***=**

−**P** *paper-name*
Specify a paper type list that the printer supports.

−**r** *class*

Remove *printer* from the specified *class*. If *printer* is the last member of *class*, then *class* will be removed.

–**S** *list*

Allow either the print wheels or aliases for character sets named in *list* to be used on the printer.

If the printer is a type that takes print wheels, then *list* is a comma or space separated list of print wheel names. (Enclose the list with quotes if it contains blank spaces.) These will be the only print wheels considered mountable on the printer. (You can always force a different print wheel to be mounted.) Until the option is used to specify a list, no print wheels will be considered mountable on the printer, and print requests that ask for a particular print wheel with this printer will be rejected.

If the printer is a type that has selectable character sets, then *list* is a comma or blank separated list of character set name ‘‘mappings’’ or aliases. (Enclose the list with quotes if it contains blank spaces.) Each ‘‘mapping’’ is of the form

known-name=*alias*

The *known-name* is a character set number preceded by **cs** (such as **cs3** for character set three) or a character set name from the **terminfo** database entry **csnm**. See **terminfo**(4). If this option is not used to specify a list, only the names already known from the **terminfo** database or numbers with a prefix of **cs** will be acceptable for the printer.

If *list* is the word **none**, any existing print wheel lists or character set aliases will be removed.

Note the other uses of the –**S** with the –**M** option described above.

The –**T** option must be invoked first with **lpadmin** to identify the printer type before the –**S** option can be used.

–**s** *system-name*[**!***printer-name*]

Make a remote printer (one that must be accessed through another system) accessible to users on your system. *system-name* is the name of the remote system on which the remote printer is located; it must be listed in the systems table (**/etc/lp/Systems**). *printer-name* is the name used on the remote system for that printer. For example, if you want to access $printer_1$ on $system_1$ and you want it called $printer_2$ on your system:

–**p** $printer_2$ –**s** $system_1$!$printer_1$

–**T** *printer-type-list*

Identify the printer as being of one or more *printer-type*s. Each *printer-type* is used to extract data from the **terminfo** database; this information is used to initialize the printer before printing each user’s request. Some filters may also use a *printer-type* to convert content for the printer. If this option is not used, the default *printer-type* will be **unknown**; no information will be extracted from **terminfo** so each user request will be printed without first initializing the printer. Also, this option must be used if the following are to work: –**o cpi**, –**o lpi**, –**o width**, and –**o length** options of the **lpadmin** and **lp** commands, and the –**S** and –**f** options of the **lpadmin** command.

If the *printer-type-list* contains more than one type, then the *content-type-list* of the –**I** option must either be specified as **simple**, as empty (–**I** ""), or not specified at all.

–**t** *number-of-trays*

Specify the number of trays when creating the printer.

–**u allow**:*login-ID-list*

–**u deny**:*login-ID-list*

Allow or deny the users in *login-ID-list* access to the printer. By default all users are allowed on a new printer. The *login-ID-list* argument may include any or all of the following constructs:

login-ID	a user on any system
system-name!*login-ID*	a user on system *system-name*
system-name!**all**	all users on system *system-name*
all!*login-ID*	a user on all systems
all	all users on all systems

For each printer, the LP print service keeps two lists of users: an ‘‘allow-list’’ of people allowed to use the printer, and a ‘‘deny-list’’ of people denied access to the printer. With the –**u allow** option, the users listed are added to the allow-list and removed from the deny-list. With the –**u deny** option, the users listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the users in the list may use the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the users in the deny-list may not use the printer. All users can be denied access to the printer by specifying –**u deny:all**. All users may use the printer by specifying –**u allow**:**all**.

–**U** *dial-info*

The –**U** option allows your print service to access a remote printer. (It does not enable your print service to access a remote printer service.) Specifically, –**U** assigns the ‘‘dialing’’ information *dial-info* to the printer. *dial-info* is used with the **dial** routine to call the printer. Any network connection supported by the Basic Networking Utilities will work. *dial-info* can be either a phone number for a modem connection, or a system name for other kinds of connections. Or, if –**U direct** is given, no dialing will take place, because the name **direct** is reserved for a printer that is directly connected. If a system name is given, it is used to search for connection details from the file **/etc/uucp/Systems** or related files. The Basic Networking Utilities are required to support this option. By default, –**U direct** is assumed.

–**v** *device*

Associate a *device* with *printer*. *device* is the path name of a file that is writable by **lp**. Note that the same *device* can be associated with more than one printer.

Removing a Printer Destination

The –**x** *dest* option removes the destination *dest* (a printer or a class), from the LP print service. If *dest* is a printer and is the only member of a class, then the class will be deleted, too. If *dest* is **all**, all printers and classes are removed. No other *options* are allowed with –**x**.

Setting/Changing the System Default Destination

The –**d** [*dest*] option makes *dest* (an existing printer or class) the new system default destination. If *dest* is not supplied, then there is no system default destination. No other *options* are allowed with –**d**.

Setting an Alert for a Print Wheel

–**S** *print-wheel* –**A** *alert-type* [–**W** *minutes*] [–**Q** *requests*]

The –**S** *print-wheel* option is used with the –**A** *alert-type* option to define an alert to mount the print wheel when there are jobs queued for it. If this command is not used to arrange alerting for a print wheel, no alert will be sent for the print wheel. Note the other use of –**A**, with the –**p** option, above.

The *alert-types* are:

mail Send the alert message using the **mail** command to the administrator.

write Write the message, using the **write** command, to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.

quiet Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the *print-wheel* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests reaches the threshold specified by the –**Q** option.

none Do not send messages until the –**A** option is given again with a different *alert-type* (other than **quiet**).

shell-command

Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blanks embedded in the command, enclose the command in quotes. Note that the **mail** and **write** values for this option are equivalent to the values **mail** *user-name* and **write** *user-name* respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the **su** command to change to another user ID. If the **su** command has been used to change the user ID, then the *user-name* for the new ID is used.

list Display the type of the alert for the print wheel on standard output. No change is made to the alert.

The message sent appears as follows:

> **The print wheel** *print-wheel* **needs to be mounted**
> **on the printer(s):**
> *printer* **(***integer*$_1$ **requests)**
> *integer*$_2$ **print requests await this print wheel.**

The printers listed are those that the administrator had earlier specified were candidates for this print wheel. The number *integer*$_1$ listed next to each printer is the number of requests eligible for the printer. The number *integer*$_2$ shown after the printer list is the total number of requests awaiting the print wheel. It will be less than the sum of the other numbers if some requests can be handled by more than one printer.

If the *print-wheel* is **all**, the alerting defined in this command applies to all print wheels already defined to have an alert.

If the **–W** option is not given, the default procedure is that only one message will be sent per need to mount the print wheel. Not specifying the **–W** option is equivalent to specifying **–W once** or **–W 0**. If *minutes* is a number greater than zero, an alert will be sent at intervals specified by *minutes.*

If the **–Q** option is also given, the alert will be sent when a certain number (specified by the argument *requests*) of print requests that need the print wheel are waiting. If the **–Q** option is not given, or *requests* is 1 or **any** (which are both the default), a message is sent as soon as anyone submits a print request for the print wheel when it is not mounted.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

FILES

/var/spool/lp/*
/etc/lp
/etc/lp/alerts/printer fault handler for **lpadmin**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

enable(1), **lp**(1), **lpstat**(1), **mail**(1), **stty**(1), **accept**(1M), **lpforms**(1M), **lpsched**(1M), **lpsystem**(1M), **dial**(3N), **terminfo**(4), **attributes**(5)

System Administration Guide

NAME

lpfilter – administer filters used with the LP print service

SYNOPSIS

/usr/sbin/lpfilter –f *filter-name* { – | **–i** | **–l** | **–x** | **–F** *pathname* }

DESCRIPTION

The **lpfilter** command is used to add, change, delete, or list a filter used with the LP print service. These filters convert the content of a file to have a content type acceptable to a printer.

OPTIONS

Arguments consist of the **–f** *filter-name* option and exactly one of the arguments appearing within braces (**{}**) in the **SYNOPSIS**.

–f *filter-name* Specifies the *filter-name* of the filter to be added, changed, reset, deleted, or listed. The filter name **all** is a special filter name defined below. The **–f** option is required.

– Adds or changes a filter as specified from standard input. The format of the input is specified below. If **–f all** is specified with the – option, the specified change is made to all existing filters. This is not useful.

–F *pathname* Adds or changes a filter as specified by the contents of the file *pathname*. The format of the file's contents is specified below. If **–f all** is specified with the **–F** option, the specified change is made to all existing filters. This is not useful.

–i Resets a filter to its default settings. Using **–f all** with the **–i** option restores all filters for which predefined settings are available to their original settings.

–x Deletes a filter. Using **–f all** with the **–x** option results in all filters being deleted.

–l Lists a filter description. Using **–f all** with the **–l** option produces a list of all filters.

USAGE

Adding or Changing a Filter

The filter named in the **–f** option is added to the filter table. If the filter already exists, its description is changed to reflect the new information in the input.

When – is specified, standard input supplies the filter description. When **–F** is specified, the file *pathname* supplies the filter description. One of these two options must be specified to add or change a filter.

When an existing filter is changed with the **–F** or – option, lines in the filter description that are not specified in the new information are not changed. When a new filter is added with this command, unspecified lines receive default values. See below.

Filters are used to convert the content of a request from its initial type into a type acceptable to a printer. For a given print request, the LP print service knows the following:

- The content type of the request (specified by **lp –T** or determined implicitly)
- The name of the printer (specified by **lp –d**)

- The printer type (specified by **lpadmin** –**T**)

  The printer type is intended to be a printer model, but some people specify it with a content type even though **lpadmin** –**I** is intended for this purpose.

- The content types acceptable to the printer (specified by **lpadmin** –I**)**

  The values specified by the **lpadmin** –**T** are treated as if they were specified by the –**I** option as well.

- The modes of printing asked for by the originator of the request (specified by various options to **lp**)

The system uses the above information to construct a list of one or more filters that converts the document’s content type into a content type acceptable to the printer and consumes all **lp** arguments that invoke filters (–**y** and –**P**).

The contents of the file (specified by the –**F** option) and the input stream from standard input (specified by –) must consist of a series of lines, such that each line conforms to the syntax specified by one of the seven lines below. All lists are comma or space separated. Each item contains a description.

Input types: *content-type-list*
Output types: *content-type-list*
Printer types: *printer-type-list*
Printers: *printer-list*
Filter type: *filter-type*
Command: *shell-command*
Options: *template-list*

Input types This gives the content types that can be accepted by the filter. The default is **any**. The document content type must be a member of this list for the initial filter in the sequence.

Output types This gives the content types that the filter can produce from any of the input (content) types. The default is **any**. The intersection of the output types of this list and the content types acceptable to the printer (from **lpadmin** –**I** and **lpadmin** –**T**) must be non-null for the last filter in the sequence. For adjacent filters in the sequence, the intersection of output types of one and the input types of the next must be non-null.

Printer types This gives the printer types for which this printer can be used. The LP print service will restrict the use of the filter to these printer types (from **lpadmin** –**T**). The default is **any**.

Printers This gives the names of the printers for which the filter can be used. The LP print service will restrict the use of the filter to just the printers named. The default is **any**.

Filter type This marks the filter as a **slow** filter or a **fast** filter. Slow filters are generally those that take a long time to convert their input (that is, minutes or hours). They are run before the job is scheduled for a printer, to keep the printers from being tied up while the filter is running. If a listed printer is on a remote system, the filter type for it must have the value **slow**. That is, if a client defines a filter, it must be a slow filter. Fast filters are generally those that convert their input quickly (that is, faster than the printer can process the data), or those that must be connected to the printer when run. Fast filters will be given to the interface program to run while connected to the physical printer.

Command This specifies which program to run to invoke the filter. The full program pathname as well as fixed options must be included in the *shell-command*; additional options are constructed, based on the characteristics of each print request and on the **Options** field. A command must be given for each filter. The command must accept a data stream as standard input and produce the converted data stream on its standard output. This allows filter pipelines to be constructed to convert data not handled by a single filter.

Options This is a comma separated list of templates used by the LP print service to construct options to the filter from the characteristics of each print request listed in the table later. The –**y** and –**P** arguments to the **lp** command cause a filter sequence to be built even if there is no need for a conversion of content types.

In general, each template is of the following form:

keyword pattern **=** *replacement*

The *keyword* names the characteristic that the template attempts to map into a filter-specific option; each valid *keyword* is listed in the table below.

A *pattern* is one of the following: a literal pattern of one of the forms listed in the table, a single asterisk (*), or a regular expression. If *pattern* matches the value of the characteristic, the template fits and is used to generate a filter-specific option.

The *replacement* is what will be used as the option.

Regular expressions are the same as those found on the **regexp**(5) manual page. This includes the **\(**...**\)** and \n constructions, which can be used to extract portions of the *pattern* for copying into the *replacement*, and the **&**, which can be used to copy the entire *pattern* into the *replacement*.

The *replacement* can also contain a *; it too, is replaced with the entire *pattern*, just like the **&** of **regexp**(5).

The keywords are:

lp Option	Characteristic	*keyword*	Possible *patterns*
–T	Content type (input)	**INPUT**	*content-type*
not applicable	Content type (output)	**OUTPUT**	*content-type*
not applicable	Printer type	**TERM**	*printer-type*
–d	Printer name	**PRINTER**	*printer-name*
–f, –o cpi=	Character pitch	**CPI**	*integer*
–f, –o lpi=	Line pitch	**LPI**	*integer*
–f, –o length=	Page length	**LENGTH**	*integer*
–f, –o width=	Page width	**WIDTH**	*integer*
–P	Pages to print	**PAGES**	*page-list*
–S	Character set	**CHARSET**	*character-set-name*
	Print wheel	**CHARSET**	*print-wheel-name*
–f	Form name	**FORM**	*form-name*
–y	Modes	**MODES**	*mode*
–n	Number of copies	**COPIES**	*integer*

Large File Behavior

See **largefile**(5) for the description of the behavior of **lpfilter** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

For example, the template

MODES landscape = –l

shows that if a print request is submitted with the **–y landscape** option, the filter will be given the option **–l**.

As another example, the template

TERM * = –T *

shows that the filter will be given the option **–T** *printer-type* for whichever *printer-type* is associated with a print request using the filter.

As a last example, consider the template

MODES prwidth\=\(.*\) = –w\1

Suppose a user gives the command

lp –y prwidth=10

From the table above, the LP print service determines that the **–y** option is handled by a **MODES** template. The **MODES** template here works because the pattern prwidth=) matches the prwidth=10 given by the user. The replacement -w1 causes the LP print service to generate the filter option -w10. If necessary, the LP print service will construct a filter pipeline by concatenating several filters to handle the user's file and all the print options. See **sh**(1) for a description of a pipeline. If the print service constructs a filter pipeline, the **INPUT** and **OUTPUT** values used for each filter in the pipeline are the types of input and output for that filter, not for the entire pipeline.

Resetting a Filter to Defaults

If the filter named is one originally delivered with the LP print service, the –**i** option restores the original filter description.

Deleting a Filter

The –**x** option is used to delete the filter specified in filter-name from the LP filter table.

Listing a Filter Description

The –**l** option is used to list the description of the filter named in filter-name. If the command is successful, the following message is sent to standard output:

Input types: *content-type-list*
Output types: *content-type-list*
Printer types: *printer-type-list*
Printers: *printer-list*
Filter type: *filter-type*
Command: *shell-command*
Options: *template-list*

If the command fails, an error message is sent to standard error.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

SEE ALSO

lp(1), **sh**(1), **lpadmin**(1M), **attributes**(5), **largefile**(5), **regexp**(5)

System Administration Guide

NOTES

If the **lp** command specifies more than one document, the filtering chain is determined by the first document. Other documents may have a different format, but they will print correctly only if the filter chain is able to handle their format.

NAME

lpforms – administer forms used with the LP print service

SYNOPSIS

lpforms –**f** *form-name option*
lpforms –**f** *form-name* –**A** *alert-type* [–**P** *paper-name* [–**d**]] [–**Q** *requests*]
[–**W** *minutes*]

DESCRIPTION

The **lpforms** command administers the use of preprinted forms, such as company letterhead paper, with the LP print service. A form is specified by its *form-name.* Users may specify a form when submitting a print request (see **lp**(1)). The argument **all** can be used instead of *form-name* with either of the command lines shown above. The first command line allows the administrator to add, change, and delete forms, to list the attributes of an existing form, and to allow and deny users access to particular forms. The second command line is used to establish the method by which the administrator is alerted that the form *form-name* must be mounted on a printer.

OPTIONS

–**f** *formname* Specify a form.

The first form of **lpforms** requires that one of the following *option* (–, –**l**, –**F**, –**x**) must be used:

–**F** *pathname* To add or change form *form-name*, as specified by the information in *pathname*.

– To add or change form *form-name*, as specified by the information from standard input.

–**x** To delete form *form-name* (this option must be used separately; it may not be used with any other option).

–**l** To list the attributes of form *form-name*.

The second form of the **lpforms** command requires the –**A** *alert-type* option. The other options are optional.

–**A** *alert-type* Defines an alert to mount the form when there are queued jobs which need it.

–**P** *paper-name* [–**d**]
Specify the paper name when creating the form. If –**d** is specified, this paper is the default.

–**Q** *requests* An alert will be sent when a certain number of print requests that need the form are waiting.

–**W** *minutes* An alert will be sent at intervals specified by minutes.

USAGE

Adding or Changing a Form

The –**F** *pathname* option is used to add a new form, *form-name*, to the LP print service, or to change the attributes of an existing form. The form description is taken from *pathname* if the –**F** option is given, or from the standard input if the – option is used. One of these two options must be used to define or change a form.

pathname is the path name of a file that contains all or any subset of the following information about the form.

Page length: *scaled−decimal−number*$_1$
Page width: *scaled−decimal−number*$_2$
Number of pages: *integer*
Line pitch: *scaled−decimal−number*$_3$
Character pitch: *scaled−decimal−number*$_4$
Character set choice: *character-set/print-wheel* [**mandatory**]
Ribbon color: *ribbon-color*
Comment:
comment
Alignment pattern: [*content-type*]
content

The term ‘‘scaled-decimal-number’’ refers to a non-negative number used to indicate a unit of size. The type of unit is shown by a ‘‘trailing’’ letter attached to the number. Three types of scaled decimal numbers can be used with the LP print service: numbers that show sizes in centimeters (marked with a trailing **c**); numbers that show sizes in inches (marked with a trailing **i**); and numbers that show sizes in units appropriate to use (without a trailing letter); lines, characters, lines per inch, or characters per inch.

Except for the last two lines, the above lines may appear in any order. The **Comment:** and *comment* items must appear in consecutive order but may appear before the other items, and the **Alignment pattern:** and the *content* items must appear in consecutive order at the end of the file. Also, the *comment* item may not contain a line that begins with any of the key phrases above, unless the key phrase is preceded with a **>** sign. Any leading > sign found in the *comment* will be removed when the comment is displayed. There is no case distinction among the key phrases.

When this command is issued, the form specified by *form-name* is added to the list of forms. If the form already exists, its description is changed to reflect the new information. Once added, a form is available for use in a print request, except where access to the form has been restricted, as described under the **−u** option. A form may also be allowed to be used on certain printers only.

A description of each form attribute is below:

Page length and **Page Width**

Before printing the content of a print request needing this form, the generic interface program provided with the LP print service will initialize the physical printer to handle pages *scaled−decimal−number*$_1$ long, and *scaled−decimal−number*$_2$ wide using the printer type as a key into the **terminfo**(4) database.

The page length and page width will also be passed, if possible, to each filter used in a request needing this form.

Number of pages

Each time the alignment pattern is printed, the LP print service will attempt to truncate the *content* to a single form by, if possible, passing to each filter the page subset of 1-*integer*.

Line pitch and **Character pitch**

Before printing the content of a print request needing this form, the interface program provided with the LP print service will initialize the physical printer to handle these pitches, using the printer type as a key into the **terminfo**(4) database. Also, the pitches will be passed, if possible, to each filter used in a request needing this form. $scaled{-}decimal{-}number_3$ is in lines-per-centimeter if a **c** is appended, and lines-per-inch otherwise; similarly, $scaled{-}decimal{-}number_4$ is in characters-per-centimeter if a **c** is appended, and characters-per-inch otherwise. The character pitch can also be given as **elite** (12 characters-per-inch), **pica** (10 characters-per-inch), or **compressed** (as many characters-per-inch as possible).

Character set choice

When the LP print service alerts an administrator to mount this form, it will also mention that the print wheel *print-wheel* should be used on those printers that take print wheels. If printing with this form is to be done on a printer that has selectable or loadable character sets instead of print wheels, the interface programs provided with the LP print service will automatically select or load the correct character set. If **mandatory** is appended, a user is not allowed to select a different character set for use with the form; otherwise, the character set or print wheel named is a suggestion and a default only.

Ribbon color

When the LP print service alerts an administrator to mount this form, it will also mention that the color of the ribbon should be *ribbon-color*.

Comment

The LP print service will display the *comment* unaltered when a user asks about this form (see **lpstat**(1)).

Alignment pattern

When mounting this form, an administrator can ask for the *content* to be printed repeatedly, as an aid in correctly positioning the preprinted form. The optional *content-type* defines the type of printer for which *content* had been generated. If *content-type* is not given, **simple** is assumed. Note that the *content* is stored as given, and will be readable only by the user **lp**.

When an existing form is changed with this command, items missing in the new information are left as they were. When a new form is added with this command, missing items will get the following defaults:

Page Length: **66**
Page Width: **80**
Number of Pages: **1**
Line Pitch: **6**
Character Pitch: **10**
Character Set Choice: **any**
Ribbon Color: **any**

Deleting a Form

LP print service" The –**x** option is used to delete the form *form-name* from the LP print service.

Listing Form Attributes

The –**l** option is used to list the attributes of the existing form *form-name*. The attributes listed are those described under **Adding and Changing a Form,** above. Because of the potentially sensitive nature of the alignment pattern, only the administrator can examine the form with this command. Other people may use the **lpstat**(1) command to examine the non-sensitive part of the form description.

Allowing and Denying Access to a Form

The –**u** option, followed by the argument **allow**:*login-ID-list* or –**u deny**:*login-ID-list* lets you determine which users will be allowed to specify a particular form with a print request. This option can be used with the –**F** or – option, each of which is described above under **Adding or Changing a Form**.

The *login-ID-list* argument may include any or all of the following constructs:

login-ID	A user on any system
system_name!*login-ID*	A user on system *system_name*
system_name!**all**	All users on system *system_name*
all!*login-ID*	A user on all systems
all	All users on all systems

The LP print service keeps two lists of users for each form: an ‘‘allow-list’’ of people allowed to use the form, and a ‘‘deny-list’’ of people that may not use the form. With the –**u allow** option, the users listed are added to the allow-list and removed from the deny-list. With the –**u deny** option, the users listed are added to the deny-list and removed from the allow-list. (Both forms of the –**u** option can be run together with the –**F** or the – option.)

If the allow-list is not empty, only the users in the list are allowed access to the form, regardless of the content of the deny-list. If the allow-list is empty but the deny-list is not, the users in the deny-list may not use the form, (but all others may use it). All users can be denied access to a form by specifying –**f deny:all**. All users can be allowed access to a form by specifying –**f allow:all**. (This is the default.)

Setting an Alert to Mount a Form

The −**f** *form-name* option is used with the −**A** *alert-type* option to define an alert to mount the form when there are queued jobs which need it. If this option is not used to arrange alerting for a form, no alert will be sent for that form.

The method by which the alert is sent depends on the value of the *alert-type* argument specified with the −**A** option. The *alert-types* are:

mail Send the alert message using the **mail** command to the administrator.

write Write the message, using the **write** command, to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.

quiet Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form *form-name* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests reaches the threshold specified by the −**Q** option.

showfault
Attempt to execute a form alert handler on each system that has a print job for that form in the queue. The fault handler is **/etc/lp/alerts/form**. It is invoked with three parameters: *form_name*, *date*, *file_name*. *file_name* is the name of a file containing the form alert message.

none Do not send messages until the −**A** option is given again with a different *alert-type* (other than **quiet**).

shell-command
Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blank spaces embedded in the command, enclose the command in quotes. Note that the **mail** and **write** values for this option are equivalent to the values **mail** *login-ID* and **write** *login-ID* respectively, where *login-ID* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the **su** command to change to another login-ID. If the **su** command has been used to change the user ID, then the *user-name* for the new ID is used.

list Display the type of the alert for the form on standard output. No change is made to the alert.

The message sent appears as follows:

The form *form-name* **needs to be mounted**
on the printer(s):
printer **(***integer*$_1$ **requests).**
integer$_2$ **print requests await this form.**
Use the *ribbon-color* **ribbon.**
Use the *print-wheel* **print wheel, if appropriate.**

The printers listed are those that the administrator has specified as candidates for this form. The number $integer_1$ listed next to each printer is the number of requests eligible for the printer. The number $integer_2$ shown after the list of printers is the total number of requests awaiting the form. It will be less than the sum of the other numbers if some requests can be handled by more than one printer. The *ribbon-color* and *print-wheel* are those specified in the form description. The last line in the message is always sent, even if none of the printers listed use print wheels, because the administrator may choose to mount the form on a printer that does use a print wheel.

Where any color ribbon or any print wheel can be used, the statements above will read:

Use any ribbon.
Use any print-wheel.

If *form-name* is **any**, the *alert-type* defined in this command applies to any form for which an alert has not yet been defined. If *form-name* is **all**, the *alert-type* defined in this command applies to all forms.

If the –**W** *minutes* option is not given, the default procedure is that only one message will be sent per need to mount the form. Not specifying the –**W** option is equivalent to specifying –**W once** or –**W 0**. If *minutes* is a number greater than **0**, an alert will be sent at intervals specified by *minutes.*

If the –**Q** *requests* option is also given, the alert will be sent when a certain number (specified by the argument *requests*) of print requests that need the form are waiting. If the –**Q** option is not given, or the value of *requests* is **1** or **any** (which are both the default), a message is sent as soon as anyone submits a print request for the form when it is not mounted.

Listing the Current Alert

The –**f** option, followed by the –**A** option and the argument **list** is used to list the *alert-type* that has been defined for the specified form *form-name.* No change is made to the alert. If *form-name* is recognized by the LP print service, one of the following lines is sent to the standard output, depending on the type of alert for the form.

- **When** *requests* **requests are queued:**
  alert with *shell-command* **every** *minutes* **minutes**
- **When** *requests* **requests are queued:**
  write to *user-name* **every** *minutes* **minutes**
- **When** *requests* **requests are queued:**
  mail to *user-name* **every** *minutes* **minutes**
- **No alert**

The phrase **every** *minutes* **minutes** is replaced with **once** if *minutes* (–**W** *minutes)* is 0.

Terminating an Active Alert

The –**A quiet** option is used to stop messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form has been mounted and then unmounted, messages will again be sent when the number of print requests reaches the threshold *requests.*

Removing an Alert Definition

No messages will be sent after the −**A none** option is used until the −**A** option is given again with a different *alert-type*. This can be used to permanently stop further messages from being sent as any existing alert definition for the form will be removed.

Large File Behavior

See **largefile**(5) for the description of the behavior of **lpforms** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

FILES

/etc/lp/alerts/form	Fault handler for **lpform**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

SEE ALSO

lp(1), **lpstat**(1), **lpadmin**(1M), **terminfo**(4), **attributes**(5), **largefile**(5)

System Administration Guide

NAME

lpget – get printing configuration

SYNOPSIS

lpget [**–k** *key*] [*destination* ... | **list**]

DESCRIPTION

The **lpget** utility reads printing configuration information from the configuration databases in **/etc/printers.conf** and **$HOME/.printers** and displays the information (called a *configuration report*) to the standard output. See **printers**(4) and **printers.conf**(4) for information about the printer configuration databases.

lpget displays a configuration report for all keys for the specified destination or destinations by default. Use the **–k** option to display a configuration report for specific keys. Use the **list** operand to display a configuration report for all configured destinations.

OPTIONS

The following options are supported:

–k *key* Displays a configuration report for *key*. See **printers.conf**(4) for information about specifying *key*.

OPERANDS

The following operands are supported:

destination Displays a configuration report for *destination*. Destination can be either a printer of a class of printers, (see **lpadmin**(1M)). Specify *destination* using atomic, POSIX-style (*server*:*destination*), or Federated Naming Service (FNS) (.../**service**/**printer**/...) names. See **printers.conf**(4) for information regarding the naming conventions for atomic and FNS names, and **standards**(5) for information concerning POSIX.

list Displays a configuration report for all configured destinations.

EXAMPLES

The following example displays a configuration report for the **bsdaddr** key for printer **catalpa**.

```
example% lpget –k bsdaddr catalpa
```

The following example displays a configuration report for all keys for all configured destinations.

```
example% lpget list
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

FILES

/etc/printers.conf	System printer configuration database.
$HOME/.printers	User-configurable printer database.
printers.conf.byname	NIS version of **/etc/printers.conf**.
fns.ctx_dir.*domain*	NIS+ version of **/etc/printers.conf**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

lp(1), **lpc**(1B), **lpq**(1B), **lpr**(1B), **lpstat**(1), **lpadmin**(1M), **lpset**(1M), **printers**(4), **printers.conf**(4), **attributes**(5), **standards**(5)

NAME

lpmove – move print requests

SYNOPSIS

lpmove *request-ID destination*
lpmove *destination1 destination2*

DESCRIPTION

The **lpmove** command moves print requests queued by **lp**(1) or **lpr**(1B) between destinations. Only use **lpmove** to move jobs on the local system.

The first form of **lpmove** moves specific print requests (*request-ID*) to a specific (*destination*).

The second form of the **lpmove** command moves all print requests from one destination (*destination1*) to another (*destination2*). This form of **lpmove** also rejects new print requests for *destination1*.

When moving requests, **lpmove** does not check the acceptance status of the destination to which the print requests are being moved (see **accept**(1M)). **lpmove** does not move requests that have options (for example, content type or requiring a special form) that cannot be handled by the new destination.

OPERANDS

The following operands are supported.

destination The name of the printer or class of printers (see **lpadmin**(1M)) *to which* **lpmove** moves a *specified* print request. Specify *destination* using atomic, POSIX-style (*server*:*destination*), or Federated Naming Service (FNS) (.../**service**/**printer**/...) names. See **printers.conf**(4) for information regarding the naming conventions for atomic and FNS names.

destination1 The name of the destination *from which* **lpmove** moves *all* print requests. Specify *destination* using atomic, POSIX-style (*server*:*destination*), or Federated Naming Service (FNS) (.../**service**/**printer**/...) names. See **printers.conf**(4) for information regarding the naming conventions for atomic and FNS names, and **standards**(5) for information regarding POSIX.

destination2 The name of the destination *to which* **lpmove** moves *all* print requests. Specify *destination* using atomic, POSIX-style (*server*:*destination*), or Federated Naming Service (FNS) (.../**service**/**printer**/...) names. See **printers.conf**(4) for information regarding the naming conventions for atomic and FNS names.

request-ID The specific print request to be moved. Specify *request-ID* as the identifier associated with a print request as reported by **lpstat**. See **lpstat**(1).

EXIT STATUS

The following exit values are returned:

0 Successful completion.
non-zero An error occurred.

FILES

/var/spool/print/*   LP print queue.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

lp(1), **lpr**(1B), **lpstat**(1), **accept**(1M), **lpadmin**(1M), **lpsched**(1M), **printers.conf**(4), **attributes**(5), **standards**(5)

System Administration Guide

NAME

lpsched, lpshut – start and stop the LP print service

SYNOPSIS

/usr/lib/lp/lpsched

lpshut

DESCRIPTION

The **lpsched** command starts or restarts the LP print service.

The **lpshut** command stops the LP print service.

Printers that are printing when **lpshut** is invoked stop printing. Printers that are restarted using **lpsched** reprint (in their entirety) print requests that were stopped by **lpshut**.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

FILES

/var/spool/lp/*	LP print queue.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

SEE ALSO

lp(1), **lpmove**(1M), **lpstat**(1), **lpadmin**(1M), **attributes**(5)

System Administration Guide

NAME

lpset – set printing configuration in /etc/printers.conf or FNS

SYNOPSIS

lpset [–**n system** | **fns**] [–**x**] [–**a** *key=value*] [–**d** *key*] *destination*

DESCRIPTION

The **lpset** utility sets printing configuration information in the system configuration databases. Use **lpset** to create and update printing configuration in **/etc/printers.conf** or Federated Naming System (FNS). See **printers.conf**(4) and **fns**(5).

Only a superuser or a member of Group 14 may execute **lpset**.

OPTIONS

[–**n system** | **fns**]
Creates or updates the configuration information for the *destination* entry in **/etc/printers.conf** or FNS. **system** specifies that the information is created or updated in **/etc/printers.conf**. **fns** specifies that the information is written using federated naming context.

If –**n** is not specified, **system** is the default.

–**x** Removes all configuration for the *destination* entry in **/etc/printers.conf** or FNS.

–**a** *key*=*value* Configures the specified *key*=*value* pair for the *destination* entry in **/etc/printers.conf** or FNS. See **printers.conf**(4) for information regarding the specification of *key*=*value* pairs.

–**d** *key* Deletes the configuration option specified by *key* for the *destination* entry in **/etc/printers.conf** or FNS. See **printers.conf**(4) for information regarding the specification of *key* and *key*=*value* pairs.

OPERANDS

destination Specifies the entry in **/etc/printers.conf** or FNS in which to create or modify information. Destination names a printer of class of printers (see **lpadmin**(1M)). Each entry in **printers.conf** describes one destination. Specify *destination* using atomic or Federated Naming Service (FNS) (.../**service**/**printer**/...) names. POSIX-style destination names are not acceptable. See **printers.conf**(4) for information regarding the naming conventions for atomic and FNS names, and **standards**(5) for information regarding POSIX.

EXAMPLES

The following example removes all existing printing configuration information for destination **dogs** from **/etc/printers.conf**.

```
example% lpset -x dogs
```

The following example sets the **user-equivalence =true** *key*=*value* pair for destination **tabloid** in FNS context.

```
example% lpset -n fns -a user-equivalence=true tabloid
```

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

FILES

/etc/printers.conf	System configuration database.
$HOME/.printers	User-configurable printer database.
printer.conf.byname (NIS)	NIS version of **/etc/printers.conf**.
fns.ctx_dir.*domain*	NIS+ version of **/etc/printers.conf**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

lp(1), **lpc**(1B), **lpq**(1B), **lpr**(1B), **lpstat**(1), **lpadmin**(1M), **lpget**(1M), **printers**(4), **printers.conf**(4), **attributes**(5), **fns**(5), **standards**(5)

NAME

lpsystem – register remote systems with the print service

DESCRIPTION

The **lpsystem** command is obsolete. The print system no longer uses the information generated by **lpsystem**. See **lpadmin**(1M), **lpusers**(1M) or **printers.conf**(4) for equivalent functionality.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

lpadmin(1M), **lpusers**(1M), **printers.conf**(4), **attributes**(5)

NAME

lpusers – set printing queue priorities

SYNOPSIS

lpusers –**d** *priority-level*
lpusers –**q** *priority-level* –**u** *login-ID-list*
lpusers –**u** *login-ID-list*
lpusers –**q** *priority-level*
lpusers –**l**

DESCRIPTION

The **lpusers** command sets limits to the queue priority level that can be assigned to jobs submitted by users of the LP print service.

The first form of the command (with –**d**) sets the system-wide priority default to *priority-level*, where *priority-level* is a value of 0 to 39, with 0 being the highest priority. If a user does not specify a priority level with a print request (see **lp**(1)), the default priority level is used. Initially, the default priority level is 20.

The second form of the command (with –**q** and –**u**) sets the default highest *priority-level* (0-39) that the users in *login-ID-list* can request when submitting a print request. The *login-ID-list* argument may include any or all of the following constructs:

login-ID	A user on any system
system_name!*login-ID*	A user on the system *system_name*
system_name!**all**	All users on system *system_name*
all!*login-ID*	A user on all systems
all	All users on all systems

Users that have been given a limit cannot submit a print request with a higher priority level than the one assigned, nor can they change a request that has already been submitted to have a higher priority. Any print requests submitted with priority levels higher than allowed will be given the highest priority allowed.

The third form of the command (with –**u**) removes any explicit priority level for the specified users.

The fourth form of the command (with –**q**) sets the default highest priority level for all users not explicitly covered by the use of the second form of this command.

The last form of the command (with –**l**) lists the default priority level and the priority limits assigned to users.

OPTIONS

–**d** *priority-level* Set the system-wide priority default to *priority-level*.

–**q** *priority-level* –**u** *login-ID-list*
Set the default highest *priority-level* that the users in *login-ID-list* can request when submitting a print request.

–u *login-ID-list* Remove any explicit priority level for the specified users.

–q *priority-level* Set the default highest priority level for all users not explicitly covered.

–l List the default priority level and the priority limits assigned to users.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWpsu

SEE ALSO

lp(1), **attributes**(5)

NAME

luxadm – administration program for the PHOTON and SSA subsystems

SYNOPSIS

luxadm [*options* . . .] *subcommand* [*options* . . .] *enclosure_name*[,*dev*] | *pathname* . . .

DESCRIPTION

The **luxadm** program is an administrative command that manages both the PHOTON and SPARCstorage Array subsystems. **luxadm** performs a variety of control and query tasks depending on the command line arguments and options used. (PHOTON will be changed to the official name when it is available.)

The command line must contain a subcommand. It may also contain options, usually at least one enclosure name or pathname, and other parameters depending on the subcommand. You need specify only as many characters as are required to uniquely identify a subcommand.

Specify the device that a subcommand interacts with by entering a "pathname". For the PHOTON subsystem, a disk device or enclosure services controller may instead be specified by entering the World Wide Name (WWN) for the device or a port to the device or by entering the name of the PHOTON enclosure and, optionally, an identifier for the particular device in the enclosure.

Pathname

Either a complete physical path name or a complete logical path name may be entered to specify the device or controller.

For PHOTON, a typical physical path name for a device is:

/devices/sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037000f96,0:a,raw

or

/devices/io-unit@f,e0200000/sbi@0,0/SUNW,socal@2,0/sf@0,0/ssd@34,0:a,raw

For all PHOTON IBs (Interface Boards) on the system, a logical link to the physical paths is kept in the directory **/dev/es**. An example of a logical link is **/dev/es/ses0**.

For PHOTON, the WWN may be used in place of the pathname to select a device or PHOTON subsystem IB. The WWN is a unique 16 hexadecimal digit value that specifies either the port used to access the device or the device itself. A typical WWN value is: **2200002037000f96**.

See **NOTES** for more information on the WWN formats.

For the SPARCstorage Array controller, a typical physical pathname is:

/devices/. . . /. . . /SUNW,soc@3,0/SUNW,pln@*axxxxxxx***,***xxxxxxxx***:ctlr**

In order to make it easier to address the SPARCstorage Array controller, a logical pathname of the form **c***N* is supported, where *N* is the logical controller number. **luxadm** uses the **c***N* name to find an entry in the **/dev/rdsk** directory of a disk that is attached to the SPARCstorage Array controller. The **/dev/rdsk** entry is then used to determine the physical name of the SPARCstorage Array controller.

For a SPARCstorage Array disk, a typical physical pathname is:

/devices/. . . /. . . /SUNW,soc@3,0/SUNW,pln@*axxxxxxx***,***xxxxxxxx***/ssd@0,0:c,raw**

and a typical logical pathname is:

/dev/rdsk/c1t0d0s2

Enclosure

For PHOTON, a device may be identified by its enclosure name and slotname:

box_name[**,f***slot_number*]
box_name[**,r***slot_number*]

box_name is the name of the PHOTON enclosure, as specified by the **enclosure_name** subcommand. When used without the optional *slot_number* parameter, the *box_name* identifies the PHOTON subystem IB.

f or **r** specifies the front or rear slots in the PHOTON enclosure.

slot_number specifies the slot number of the device in the PHOTON enclosure, **0-6** or **0-10**.

See **disks**(1M) and **devlinks**(1M) for more information on logical names for disks and subsystems.

OPTIONS

–v Verbose mode.

–e Expert mode. Note: This option is not recommended for the novice user.

Other options specific to each subcommand are described below.

OPERANDS

enclosure_name The enclosure name of the enclosure(s) specified by the enclosure or pathname.

pathname The SPARCstorage Array controller or a disk in the SPARCstorage Array.

USAGE

Subcommands

display *enclosure*[,*dev*] . . . | *pathname* . . .
display –p *enclosure*[,*dev*] . . . | *pathname* . . .
display –r *enclosure*[,*dev*] . . . | *pathname* . . .

Displays enclosure or device specific data.

Subsystem data consists of enclosure environmental sense information and status for all subsystem devices, including disks.

Disk data consists of inquiry, capacity, and configuration information. If the verbose option **–v** is specified, mode sense data is also displayed.

–p The **–p** option displays performance information for the device or subsystem specified by the enclosure or pathname.

–r The **–r** option displays error information for the device or subsystem specified by the enclosure or pathname.

probe [–**p**]

This subcommand finds and displays information about all attached PHOTON subsystems, including the logical pathname, the WWNs, and enclosure names. This subcommand will warn the user if it finds different PHOTONS with the same enclosure names.

–**p** Use the –**p** option to also display the physical pathname.

download [–**s**] [–**w** *WWN*] [–**f** *filename*] *enclosure*. . . | *pathname*. . .

Download the prom image specified by *filename* to the PHOTON subsystem IBs or the SPARCstorage Array controllers specified by the enclosure or pathname. Note: The SPARCstorage Array must be reset in order to use the downloaded code.

When the PHOTON's download is complete, the PHOTON will be reset and the downloaded code executed. If no filename is specified, the default prom image will be used. The default prom image for the PHOTON is in the directory **/usr/lib/firmware/luxib** and is named **ibfirmware**. The default prom image for the SPARCstorage Array controller is in **/usr/lib/firmware/ssa/ssafirmware**.

–**s** Save. The –**s** option is used to save the downloaded firmware in the FEPROM. If –**s** is not specified, the downloaded firmware will not be saved across power cycles. Note: The –**s** option does not apply to the SPARCstorage Array controller as it *always* writes the downloaded firmware into the FEPROM. When using the –**s** option, the **download** subcommand modifies the FEPROM on the subsystem and should be used with *caution*.

–**w** *WWN*

The –**w** option applies only to the SPARCstorage Array. Change the SPARCstorage Array controller's World Wide Name. *WWN* is a 12-digit hex number; leading zeros are required. The new SPARCstorage Array controller's image will have the least significant 6 bytes of the 8-byte World Wide Name modified to *WWN*.

enclosure_name *####* *enclosure* | *pathname*

This subcommand applies only to the PHOTON. This subcommand is used to change the enclosure name of the enclosure(s) specified by the enclosure or pathname. The new name (*####*) must be 15 or less characters. The only allowed characters are alphabetic and / or numeric digits. Another restriction is that names of the form **c***xxx*, where *xxx* are numeric characters, are not allowed because of conflicts with the SPARCstorage Array names of the form **c***N*.

fast_write [–**s**] –**c** *pathname*
fast_write [–**s**] –**d** *pathname*
fast_write [–**s**] –**e** *pathname*

This subcommand applies only to the SPARCstorage Array. Enable or disable the use of the NVRAM to enhance the performance of writes in the

SPARCstorage Array. *pathname* may refer to the SPARCstorage Array controller or to an individual disk.

–s This option causes the SPARCstorage Array to save the change so it will persist across power-cycles.
–c Enable fast writes for synchronous writes only.
–e Enable fast writes.
–d Disable fast writes.

fc_s_download [**–F**] [**–f** *fcode-file*]

Download the fcode contained in the file **fcode-file** into *all* the FC/S Sbus Cards. This command is interactive and expects user confirmation before downloading the fcode. When invoked without the **–f** *fcode-file* option, the current version of the fcode in each FC/S Sbus card is printed. When the **–F** option is used, the fcode is forcibly downloaded, but the command still expects user confirmation before the download. The version of the FC/S Sbus Cards fcode that was released with this version of the Operating System is kept in the directory **usr/lib/firmware/fc_s** and is named **fc_s_fcode**.

Use **fc_s_download** *only* in single-user mode. Using **fc_s_download** to update a host adapter while there is I/O activity through that adapter *will* cause the adapter to reset.

fcal_s_download [**–f** *fcal_s_fcode*]

Download the fcode contained in the file **fcal_s_fcode** into *all* the SOC+/S Sbus Cards. This command is interactive and expects user confirmation before downloading the fcode. When invoked without the **–f** *fcal_s_fcode* option, the current version of the fcode in each PHOTON/S Sbus card is printed. The version of the SOC+/S Sbus Cards fcode that was released with this version of the Operating System is kept in the directory **usr/lib/firmware/fc_s** and is named **fcal_s_fcode**.

Use **fcal_s_download** *only* in single-user mode. Using **fcal_s_download** to update a host adapter while there is I/O activity through that adapter *will* cause the adapter to reset.

fpm_download [**–f** *filename*] *enclosure*. . . | *pathname*. . .

This subcommand is used to download a new language module for the PHOTON subsystem to use when interfacing with the Front Panel Module or when it downloads an image to be displayed by the FPM. The filename is the path of the new Front Panel Module image. This file is downloaded to the PHOTON subsystem specified by the enclosure or pathname. If no filename is specified, the default language module for the current locale will be used, if available. The default language module is in the **/usr/lib/locale/C/LC_MESSAGES** directory and is named **lux_fpm_code**. This command verifies that the module specified is a valid FPM module before downloading it. The **LANG** environment variable is used to find the the directory that contains the **LC_MESSAGES** directory. The default directory is "**C**".

inquiry *enclosure*[,*dev*] . . . | *pathname* . . .

Display the inquiry information for the selected device specified by the enclosure or pathname.

insert_device [*enclosure*,*dev* . . .]

This subcommand applies only to the PHOTON. This subcommand assists the user in the hot insertion of a new device or a chain of new devices. Refer to **NOTES** for limitations on hotplug operations.

If more than one enclosure has been specified, concurrent hot insertions on multiple busses can be performed. With no arguments to the subcommand, entire enclosures can be inserted.

This subcommand guides the user interactively through the hot insertion steps of a new device or chain of devices:

- Informs the user that the device(s) can be safely inserted.
- Requests confirmation from the user that the list(s) is / are as expected.
- Creates the logical device names for the new devices.
- Displays the logical path name for the devices.

led *enclosure*,*dev* . . . | *pathname*. . .

Display the current state of the LED for the disk specified by the enclosure or pathname.

led_blink *enclosure*,*dev* . . . | *pathname* . . .

The **led_blink** subcommand requests the subsystem to start blinking the yellow LED associated with the disk specified by the enclosure or pathname. This subcommand only applies to subsystems that support this functionality.

led_off *enclosure*,*dev* . . . | *pathname* . . .

The **led_off** subcommand requests the subsystem to disable (turn off) the yellow LED associated with the disk specified by the enclosure or pathname. Note: On a PHOTON subsystem, this may or may not cause the yellow LED to turn off or stop blinking depending on the state of the PHOTON subsystem. Refer to the *PHOTON Array Installation and Service Manual* (p / n 802-7573).

led_on *enclosure*,*dev* . . . | *pathname* . . .

The **led_on** subcommand requests the subsystem to enable (turn on) the LED associated with the disk specified by the enclosure or pathname.

nvram_data *pathname*

Display the amount of fast write data in the NVRAM for the specified disk. This command can only be used for an individual disk.

password *enclosure* | *pathname*

Change the FPM (Front Panel Module) password of the subsystem associated with the IB addressed by the enclosure or pathname. The password must be a 1-4 digit number, for example, **1**. The user is prompted for the password then again to verify it was entered correctly. The *enclosure* or *pathname* must specify the IB.

perf_statistics –d *pathname*
perf_statistics –e *pathname*

Enable or disable the accumulation of performance statistics for the specified SPARCstorage Array controller. The accumulation of performance statistics must be enabled before using the **display –p** subcommand. This subcommand can be issued only to the SPARCstorage Array controller.

–e Enable the accumulation of performance statistics.
–d Disable the accumulation of performance statistics.

power_off *enclosure* . . . | *pathname*[*enclosure-port*] . . .

When a PHOTON is addressed, this subcommand causes the PHOTON subsystem to go into the power-save mode. Note: The PHOTON drives are not available when in the power-save mode. When an Enclosure Services card within the SPARCstorage Array is addressed, the enclosure is powered down.

power_on *enclosure* . . . | *pathname* . . .

This subcommand causes the PHOTON subsystem to go out of the power-save mode. There is no programmatic way to power on the SPARCstorage Array enclosure.

purge *pathname*

This subcommand applies only to the SPARCstorage Array. Purge any fast write data from NVRAM for one disk, or all disks if the controller is specified. This option should be used with caution, usually only when a drive has failed.

release *enclosure*[,*dev*] . . . | *pathname* . . .]

Release a reservation held on the specified disk(s). If the enclosure or pathname is of the PHOTON IB, then all disks in the PHOTON subsystem that are on the same loop as the IB will be released. If the pathname is of the SPARCstorage Array controller, then all of the disks in the SPARCstorage Array will be released.

remove_device *enclosure*[,*dev*] . . . | *pathname* . . .

This subcommand applies only to the PHOTON. It assists the user in hot removing a device or a chain of devices. This subcommand can also be used to remove entire enclosures. Refer to **NOTES** for limitations on hotplug operations.

This subcommand guides the user interactively through the hot removal of a device or devices:

- Checks whether the device is busy; if so, warns the user.
- Offlines the device, which will fail if the disk is open.
- Informs the user that device(s) can be safely removed.
- Informs the user which device to remove by blinking the activity LED on the enclosure.
- Requests confirmation from the user that the list(s) is / are as expected.
- Removes the logical device names for the device that was removed.

reserve *enclosure*[,*dev*] . . . | *pathname* . . .

Reserve the specified disk(s) for exclusive use by the issuing host. If the enclosure or pathname is of the PHOTON IB, then all disks in the PHOTON subsystem that are on the same loop as the IB will be reserved. If the pathname is of the SPARCstorage Array controller, then all of the disks in the SPARCstorage Array will be reserved.

set_boot_dev [−**y**] *pathname*

Set the boot-device variable in the system PROM to the physical device name specified by *pathname*, which can be a block special device or a mount-point. The command normally runs interactively requesting confirmation for setting the default boot-device in the PROM. The −**y** option can be used to run it non-interactively, in which case no confirmation is requested or required.

start [−**t** *tray-number*] *enclosure*[,*dev*] . . . | *pathname* . . .

Spin up the specified disk(s). If *pathname* specifies the SPARCstorage Array controller, this action applies to all disks in the SPARCstorage Array.

−**t** Spin up all disks in the tray specified by tray-number. *pathname* must specify the SPARCstorage Array controller.

stop [−**t** *tray-number*] *enclosure*[,*dev*] . . . | *pathname* . . .

Spin down the specified disk(s). If *pathname* specifies the SPARCstorage Array controller, this action applies to all disks in the SPARCstorage Array.

−**t** Spin down all disks in the tray specified by tray-number. *pathname* must specify the SPARCstorage Array controller.

sync_cache *pathname*

This subcommand applies only to the SPARCstorage Array. Flush all outstanding writes for the specified disk(s) from NVRAM to the media. If *pathname* specifies the controller, this action applies to all disks in the SPARCstorage Array.

Enclosure Services Card Subcommands

The **env_display** and **alarm**∗ subcommands apply only to an Enclosure Services Card in a SPARCstorage Array:

alarm_off *pathname* [*enclosure-port*]

Disable the audible alarm for this enclosure. When invoked without an option, the current state of audible alarm is printed.

alarm_on *pathname* [*enclosure-port*]

Enable the audible alarm for this enclosure. When invoked without an option, the current state of audible alarm is printed.

alarm_set *controller-pathname* [*enclosure-port*] [*seconds*]

Set the audible alarm setting to *seconds*.

env_display *pathname* [*enclosure-port*]

Display the environmental information fo the specified unit.

Expert Mode Subcommands

Note: The following subcommands are for expert use only, and are applicable only to the PHOTON subsystem. They should only be used by users that are knowledgeable about the PHOTON subsystem and Fiber Channel Loops.

drive_off *enclosure*[,*dev*] . . . | *pathname* . . .

Set the drive (or if the enclosure or pathname specifies the PHOTON IB, set all disks in the enclosure) to the "drive off" state. The PHOTON subsystem has the capability to set each drive to the "drive off / unmated" state. In the "drive off / unmated" state, the drive is spun down (stopped) and in the bypass mode.

drive_on *enclosure*[,*dev*] . . . | *pathname* . . .

Set the drive (or if the enclosure or pathname specifies the PHOTON IB, set all disks in the enclosure) to its normal start-up state.

forcelip *enclosure*[,*dev*] . . . | *pathname* . . .

Force the link to reinitialize, using the Loop Initialization Primitive (LIP) Sequence. The enclosure or pathname can specify any device on the loop.

Note: This is an expert only command and should be used with *caution*. It will reset all ports on the loop and may cause the device addresses to change.

rdls *enclosure*[,*dev*] . . . | *pathname*. . .

Read the link error status block from the device specified by *enclosure* or *pathname*. This will display the link error status information for the device specified by the enclosure or pathname as well as for the Host Adapter associated with this enclosure or pathname, if available.

EXAMPLES

An example of how to find all of the PHOTONs on a system is:

```
array% luxadm probe
```

To get a display of an SSA:

```
array% luxadm display c1
```

To get a display of a PHOTON:

```
array% luxadm display /dev/es/ses0
```

To get a display of two subsystems using the enclosure names:

```
array% luxadm display BOB system1
```

To display information about the first disk in the front of the enclosure named **BOB** (to specify the front disks, use "**f**"; to specify the rear disks, use "**r**"):

```
array% luxadm display BOB,f1
```

To display information about a PHOTON disk or enclosure with the port WWN of **2200002037001246**:

```
array% luxadm display 2200002037001246
```

This is an example of using only as many characters as are required to uniquely identify a subcommand:

```
array% luxadm disp BOB
```

To get error information about the loop that the enclosure **BOB** is on:

```
array% luxadm display –r   BOB
```

To download new firmware into the Interface Board in the enclosure named **BOB** (note that this is using the default path for the file to download):

```
array% luxadm download BOB
```

To display information from the SCSI Inquiry command from all individual disks on the system, using only as many characters as necessary to uniquely identify the inquiry subcommand:

```
array% luxadm inq /dev/rdsk/c?t?d?s2
```

To set up the password for the Front Panel Module for the enclosure named **BOB**:

```
array% luxadm password BOB
Changing FPM password for system BOB
New Password:
Re-enter new password:
```

To hotplug a new drive into the first slot in the front of the enclosure named **BOB**:

```
array% luxadm insert_device BOB,f1
```

To run an expert subcommand, this subcommand will force a loop initialization on the loop that the enclosure **BOB** is on:

```
array% luxadm –e forcelip BOB
```

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **luxadm**:

LANG

EXIT STATUS

The following exit values are returned:

0 Successful completion.

–1 An error occurred.

FILES

usr/lib/firmware/luxib/ibfirmware

usr/lib/firmware/fc_s/fcal_s_fcode

usr/lib/firmware/fc_s/fc_s_fcode

usr/lib/firmware/ssa/ssafirmware

usr/lib/locale/C/LC_MESSAGES/lux_fpm_code

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

usr/sbin

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluxop

SEE ALSO

devlinks(1M), **disks**(1M), **ssaadm**(1M), **attributes**(5), **environ**(5)

Snively, R., *Tutorial for SCSI use of IEEE company_ID*, X3T10/97-101r2, February 25, 1996

PHOTON Array Installation and Service Manual (p/n 802-7573).

NOTES

For more information on the PHOTON, see the *PHOTON Array Installation and Service Manual*. For a definition of the IEEE extended WWN, refer to *Tutorial for SCSI use of IEEE company_ID,* R. Snively. Refer to **SEE ALSO** above.

NAME

mail.local – store mail in a mailbox

SYNOPSIS

/usr/lib/mail.local [**–f** *sender*] [**–d**] *recipient*

DESCRIPTION

mail.local reads the standard input up to an end-of-file and appends it to each user's mail file (mailbox). This program is intended to be used by **sendmail**(1M) as a mail delivery agent for local mail. It is not a user interface agent.

Messages are appended to the user's mail file in the **/var/mail** directory. The user must be a valid user name.

Each delivered mail message in the mailbox are preceded by a "Unix From line" with the following format:

From *sender_address time_stamp*

The sender_address is extracted from the SMTP envelope address (the envelope address is specified with the –**f** option).

A trailing blank line is also added to the end of each message.

The mail files are locked with a **.lock** file while mail is appended.

The mail files are created will mode **660**, owner is set to recipient and group is set to **mail**. If the ‘‘biff’’ service is returned by **getservbyname**(3N), the biff server is notified of delivered mail. This program also computes the **Content-Length:** header which will be used by the mailbox reader to mark message boundary.

OPTIONS

–**f** *sender* specified the "envelope from address" of the message. This flag is technically optional, but should be used.

–**d** specified the recipient of the message, this flag is also optional and is supported here for backward compatibility. (i.e **mail.local** *recipient* is the same as **mail.local** –**d** *recipient*)

ENVIRONMENT

TZ Used to set the appropriate time zone on the timestamp.

EXIT CODES

0 on success.

>0 if an error occurs.

FILES

/tmp/local.*XXXXXX*	temporary files
/tmp/lochd.*XXXXXX*	temporary files
/var/mail/*user_name*	user's mail file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mail(1), **comsat**(1M), **sendmail**(1M), **getservbyname**(3N), **attributes**(5)

NAME

makedbm – make a dbm file, or get a text file from a dbm file

SYNOPSIS

makedbm [B –bls] [**–d** *yp_domain_name*] [**–i** *yp_input_file*]
[**–o** *yp_output_name*] [**–m** *yp_master_name*] *infile outfile*

makedbm [**–u** *dbmfilename*]

DESCRIPTION

The **makedbm** utility takes the *infile* and converts it to a pair of files in **ndbm** format (see **dbm_clearerr**(3)), namely *outfile***&.pag** and *outfile***&.dir**. Each line of the input file is converted to a single **dbm** record. All characters up to the first TAB or SPACE form the key, and the rest of the line is the data. If a line ends with '\' (backslash), the data for that record is continued on to the next line. **makedbm** does not treat '#' (pound-sign) as a special character.

Because **makedbm** is mainly used in generating **dbm** files for the NIS name service, it generates a special entry with the key *yp_last_modified*, which is the date of *infile* (or the current time, if *infile* is '–'). The entries that have keys with the prefix *yp_* are interpreted by NIS server utilities.

OPTIONS

The following options are supported:

–b Insert the YP_INTERDOMAIN into the output. This key causes **ypserv**(1M) to use DNS for host name and address lookups for hosts not found in the maps.

–d *yp_domain_name* Create a special entry with the key *yp_domain_name*.

–i *yp_input_file* Create a special entry with the key *yp_input_file*.

–l Lower case. Convert the keys of the given map to lower case, so that, for example, host name matches will succeed independent of upper or lower case distinctions.

–m *yp_master_name* Create a special entry with the key *yp_master_name*. If no master host name is specified, *yp_master_name* will be set to the local host name.

–o *yp_output_name* Create a special entry with the key *yp_output_name*.

–s Secure map. Accept connections from secure NIS networks only.

–u *dbmfilename* Undo a **dbm** file. Prints out the file in text format, one entry per line, with a single space separating keys from values.

OPERANDS

The following operands are supported:

infile Input file for **makedbm**. If *infile* is '–' (dash), the standard input is read.

outfile One of two output files in **ndbm** format: *outfile***&.pag** and *outfile***&.dir**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ypserv(1M), **dbm_clearerr**(3), **attributes**(5)

NAME

mibiisa – Sun SNMP Agent

SYNOPSIS

mibiisa [**–ar**] [**–c** *config-dir*] [**–d** *debug-level*] [**–p** *port*]

DESCRIPTION

The **mibiisa** utility is an RFC 1157-compliant SNMP agent. It supports MIB-II as defined in RFC 1213, with Sun extensions under Sun's enterprise number. The MIB (Management Information Base) is both readable and writable. The **mibiisa** utility supports all SNMP protocol operations including GET-REQUEST, GETNEXT-REQUEST, SET-REQUEST, GET-REPLY, and TRAP.

The **mibiisa** utility supports the coldStart, linkUp, linkDown, and authentication traps. The authentication trap may be disabled by a command-line switch, which itself may be overridden by a management station writing to a MIB variable in the standard SNMP MIB group.

The **mibiisa** utility supports four distinct views of the MIB. The view used for any request is determined by the community string contained in that request.

To enhance security, **mibiisa** supports an option to block all writes to the MIB. You can also limit the set of management stations from which the agent will accept requests in the configuration file used when starting the **mibiisa**. See the **SECURITY** section for more information.

Unless overridden, **mibiisa** uses UDP port 161, the standard SNMP port. The **mibiisa** utility issues traps through the same port on which it receives SNMP requests.

The **mibiisa** utility must run with super-user privileges and is typically started at system startup via **/etc/rc3.d**. **mibiisa** may not be started using **inetd**(1M). When started, **mibiisa** detaches itself from the keyboard, disables all signals except **SIGKILL**, **SIGILL**, **SIGUSR1**, and **SIGUSR2**, and places itself in the background.

OPTIONS

The following options are supported:

–a Disable the generation of authentication traps. However, an SNMP manager may write a value into **snmpEnableAuthenTraps** to enable or disable authentication traps.

fig-dir" Specify a directory where it expects **snmpd.conf** file, on startup. The default directory is **/etc/snmp/conf** for Solaris 2.x.

–d *debug-level* Debug. A value of 0 disables all debug and is the default. Levels 1 through 3 represent increasing levels of debug output. When **mibiisa** receives the signal **SIGUSR1**, it resets the debug-level to 0. When **mibiisa** receives the signal **SIGUSR2**, it increments the debug-level by one.

Debug output is sent to the standard output in effect at the time **mibiisa** is started. No matter what debug level is in effect, certain significant events are logged in the system log.

–p *port* Define an alternative UDP port on which **mibiisa** listens for incoming requests. The default is UDP port 161.

–r Place the MIB into read-only mode.

CONFIGURATION FILE

The **snmpd.conf** file is used for configuration information. Each entry in the file consists of a keyword followed by a parameter string. The keyword must begin in the first position. Parameters are separated from the keyword and from one another by white space. Case in keywords is ignored. Each entry must be contained on a single line. All text following (and including) a pound sign (#) is ignored. Keywords currently supported are:

sysdescr The value to be used to answer queries for sysDescr.

syscontact The value to be used to answer queries for sysContact.

syslocation The value to be used to answer queries for sysLocation.

trap The parameter names one or more hosts to receive traps. Only five hosts may be listed.

system-group-read-community
The community name to get read access to the system group and Sun's extended system group.

system-group-write-community
The community name to get write access to the system group and Sun's extended system group.

read-community
The community name to get read access to the entire MIB.

write-community
The community name to get write access to the entire MIB (implies read access).

trap-community
The community name to be used in traps.

kernel-file The name of the file to use for kernel symbols.

managers The names of hosts that may send SNMP queries. Only five hosts may be listed on any one line. This keyword may be repeated for a total of 32 hosts.

newdevice The additional devices which are not built in SNMPD. The format is as follows:

newdevice type speed name

where *newdevice* is the keyword, *type* is an integer which has to match your schema file, *speed* is the new device's speed, and *name* is this new device's name.

An example **snmpd.conf** file is shown below:

```
sysdescr      Sun SNMP Agent, SPARCstation 10, Company
                  Property Number 123456
syscontact    Cliff Claven
sysLocation   Stool next to Norms at Cheers
#
system-group-read-community    public
system-group-write-community   private
#
read-community  all_public
write-community all_private
#
trap          localhost
trap-community  SNMP-trap
#
#kernel-file   /vmunix
#
managers      lvs golden
managers      swap
```

INSTALLATION

The **mibiisa** utility and its configuration file, **snmpd.conf**, may be placed in any directory. However, **/usr/lib/snmp** for Solaris 2.4, 2.5, and 2.6 is suggested for **mibiisa** itself and **/etc/snmp/conf** (Solaris 2.4, 2.5, and 2.6) for the configuration file. You should modify the configuration file as appropriate. If you make any changes to **snmpd.conf** file keyword values, you must kill and restart **mibiisa** for the changes to take effect.

Your **/etc/services** file (or NIS equivalent) should contain the following entries:

```
snmp         161/udp                  # Simple Network Mgmt Protocol
snmp-trap    162/udp     snmptrap     # SNMP trap (event) messages
```

The following is an example for Solaris 2.*x*:

```
#
# Start the SNMP agent
#

if [ -f /etc/snmp/conf/snmpd.conf -a -x
  /usr/lib/snmp/mibiisa ];
then
/opt/SUNWconn/snm/agents/snmpd
echo 'Starting SNMP-agent.'
```

SECURITY

SNMP, as presently defined, offers relatively little security. The **mibiisa** utility accepts requests from other machines, which can have the effect of disabling the network capabilities of your computer. To limit the risk, the configuration file lets you specify a list of up to 32 manager stations from which **mibiisa** will accept requests. If you do not specify any such manager stations, **mibiisa** accepts requests from anywhere.

The **mibiisa** utility also allows you to mark the MIB as “read-only” by using the **–r** option.

Finally, **mibiisa** supports four different community strings. These strings, however, are visible in the configuration file and within the SNMP packets as they flow on the network.

The configuration file should be owned by, and readable only by super-user. In other words the mode should be:

For Solaris 2.4, 2.5, and 2.6:

```
–rw––––––– 1 root        2090 Oct 17 15:04 /etc/snmp/conf/snmpd.conf
```

MIB

This section discusses some of the differences between the **mibiisa** MIB and the standard MIB-II (as defined in RFC 1213).

The following variables are read-only in the **mibiisa** MIB:

sysName
atIfIndex
ipDefaultTTL

These variables are read-write in the standard MIB-II.

The **mibiisa** MIB Address Translation tables support limited write access: only atPhysAddress may be written, either to change the physical address of an existing entry or to delete an entire ARP table entry.

The **mibiisa** MIB IP Net to Media table supports limited write access: only ipNetToMediaPhysAddress and ipNetToMediaType may be written, either to change the physical address of an existing entry or to delete an entire ARP table entry.

The following variables are read-write in the **mibiisa** MIB; however, these variables have fixed values. Any new values “set” to them are accepted, but have no effect:

ipRoutIfIndex
ipRouteMetric1
ipRouteMetric2
ipRouteMetric3
ipRouteMetric4
ipRouteType
ipRouteAge
ipRouteMask
ipRouteMetric5

The following **mibiisa** MIB variable reflects the actual state of the related table entry. “Sets” are accepted but have no effect:

tcpConnState

The following **mibiisa** MIB variables are readable, but return a fixed value:

icmpInDestUnreachs	Returns 1
icmpInTimeExcds	Returns 1
icmpInParmProbs	Returns 1
icmpInSrcQuenchs	Returns 1
icmpInRedirects	Returns 1
icmpInEchos	Returns 1
icmpInEchoReps	Returns 1
icmpInTimestamps	Returns 1
icmpInTimestampReps	Returns 1
icmpInAddrMasks	Returns 1
icmpInAddrMaskReps	Returns 1
icmpOutDestUnreachs	Returns 1
icmpOutTimeExcds	Returns 1
icmpOutParmProbs	Returns 1
icmpOutSrcQuenchs	Returns 1
icmpOutRedirects	Returns 1
icmpOutEchos	Returns 1
icmpOutEchoReps	Returns 1
icmpOutTimestamps	Returns 1
icmpOutTimestampReps	Returns 1
icmpOutAddrMasks	Returns 1
icmpOutAddrMaskReps	Returns 1
ifInUnknownProtos	Returns 0
ipAdEntBcastAddr	Returns 1
ipAdEntReasmMaxSiz	Returns 65535
ipRouteMetric1	Returns –1
ipRouteMetric2	Returns –1
ipRouteMetric3	Returns –1
ipRouteMetric4	Returns –1
ipRouteAge	Returns 0
ipRouteMetric5	Returns –1
ipNetToMediaType	Returns (3) dynamic
ipRoutingDiscards	Returns 0

The following variables return a fixed value of **0** for drivers not conforming to the GLD framework (see **gld**(7D)), including the old LAN drivers on SPARC machines:

ifInOctets	Returns 0
ifInNUcastPkts	Returns 0
ifInDiscards	Returns 0
ifOutOctets	Returns 0
ifOutNUcastPkts	Returns 0
ifOutDiscards	Returns 0

SCHEMA ATTRIBUTES

The following describes the attributes in the group and table definitions in the **/var/snmp/mib/sun.mib** file.

system

The **system** group reports statistics about a particular system (for example, a workstation or a printer).

sysDescr – A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. This value must only contain printable ASCII characters. (string[255])

sysObjectID – The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining what type of equipment is being managed. For example, if vendor "Flintstones, Inc." was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its "Fred Router." (objectid)

sysUpTime – Time (in hundredths of a second) since the network management portion of the system was last reinitialized. (timeticks)

sysContact – The textual identification of the contact person for this managed node, together with information on how to contact this person. (string[255])

sysName – An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name. (string[255])

sysLocation – The physical location of this node (for example, "telephone closet, 3rd floor" (string[255]))

sysServices – A value indicating the set of services that this entity primarily offers. (int) The value is a sum. This sum initially takes the value zero. Then, for each layer L in the range 1 through 7 for which this node performs transactions, 2 raised to (L - 1) is added to the sum. For example, a node that performs primarily routing functions would have a value of 4 (2**(3-1)). In contrast, a node that is a host offering application services would have a value of 72 (2**(4-1) + 2**(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:

layer	**functionality**
1	physical (such as repeaters)
2	datalink / subnetwork (such as bridges)
3	internet (such as IP gateways)
4	end-to-end (such as IP hosts)
7	applications (such as mail relays)

For systems including OSI protocols, Layers 5 and 6 may also be counted.

interfaces

The **interfaces** group reports the number of interfaces handled by the agent.

ifNumber – The number of network interfaces, regardless of their current state, present on this system. (int)

ifTable

The **ifTable** is a table of interface entries. The number of entries is given by the value of ifNumber.

ifIndex – A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one reinitialization of the entity's network management system to the next reinitialization. (int)

ifDescr – A textual string containing information about the interface. This string should include the name of the manufacturer, the product name, and the version of the hardware interface. (string[255])

ifType – The type of interface, distinguished according to the physical / link protocol(s) immediately below the network layer in the protocol stack. (enum)

ifMtu – The size of the largest datagram that can be sent / received on the interface, specified in octets. For interfaces used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface. (int)

ifSpeed – An estimate of the interface's current bandwidth in bits-per-second. For interfaces that do not vary in bandwidth, or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. (gauge)

if1hysAddress – The interface's address at the protocol layer immediately below the network layer in the protocol stack. For interfaces without such an address (for example, a serial line), this object should contain an octet string of zero length. (octet[128])

ifAdminStatus – The desired state of the interface. The testing(3) state indicates that no operational packets can be passed. (enum)

if OperStatus – The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed. (enum)

ifLastChange – The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last reinitialization of the local network management subsystem, then this object contains a zero value. (timeticks)

ifInOctets – The total number of octets received on the interface, including framing characters. (counter) Returns a fixed value of 0.

ifInUcastPkts – The number of subnetwork-unicast packets delivered to a higher-layer protocol. (counter)

ifInNUcastPkts – The number of non-unicast (that is, subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol. (counter) Returns a fixed value of 0.

ifInDiscards – The number of inbound packets chosen to be discarded, even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (counter) Returns a fixed value of 0.

ifInErrors – The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. (counter)

ifInUnknownProtos – The number of packets received via the interface that were discarded because of an unknown or unsupported protocol. (counter) Returns a fixed value of 0.

ifOutOctets – The total number of octets transmitted out of the interface, including framing characters. (counter) Returns a fixed value of 0.

ifOutUcastPkts – The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. (counter)

ifOutNUcastPkts – The total number of packets that higher-level protocols requested be transmitted to a non- unicast (that is, a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. (counter) Returns a fixed value of 0.

ifOutDiscards – The number of outbound packets that were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. (counter) Returns a fixed value of 0.

ifOutErrors – The number of outbound packets that could not be transmitted because of errors. (counter)

ifOutQLen – The length of the output packet queue (in packets). (gauge)

ifSpecific – A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an Ethernet, then the value of this object refers to a document defining objects specific to Ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier. Any conformant implementation of ASN.1 and BER must be able to generate and recognize this value. (objectid)

atTable

atTable Address Translation tables contain the NetworkAddress to physical address equivalences. Some interfaces do not use translation tables for determining address equivalences (for example, DDN-X.25 has an algorithmic method). If all interfaces are of this type, then the Address Translation table is empty, that is, has zero entries.

atIfIndex – The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

atPhysAddress – The media-dependent physical address. (octet[128]) Setting this object to a null string (one of zero length) has the effect of invaliding the corresponding entry in the atTable object. That is, it effectively dissociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object.

atNetAddress – The NetworkAddress (that is, the IP address) corresponding to the media-dependent physical address. (netaddress)

ip The **ip** group reports statistics about the Internet Protocol (IP) group.

ipForwarding – The indication of whether this entity is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, this entity. IP gateways forward datagrams. IP hosts do not— except those source-routed via the host. (enum)

Note that for some managed nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a “badValue” response if a management station attempts to change this object to an inappropriate value.

ipDefaultTTL – The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol. (int)

ipInReceives – The total number of input datagrams received from interfaces, including those received in error. (counter)

ipInHdrErrors – The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IP options, and so on. (counter)

ipInAddrErrors – The number of input datagrams discarded because the IP address in their IP header’s destination field was not a valid address to be received at this entity. This count includes invalid addresses (for example, 0.0.0.0) and addresses of unsupported Classes (for example, Class E). For entities that are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address. (counter)

ipForwDatagrams – The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities that do not act as IP Gateways, this counter will include only those packets that were Source-Routed via this entity, and the Source- Route option processing was successful. (counter)

ipInUnknownProtos – The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol. (counter)

ipInDiscards – The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded, for example, for lack of buffer space. Note that this counter does not include any datagrams discarded while awaiting reassembly. (counter)

ipInDelivers – The total number of input datagrams successfully delivered to IP user-protocols (including ICMP). (counter)

ipOutRequests – The total number of IP datagrams that local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in ipForwDatagrams. (counter)

ipOutDiscards – The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (for example, for lack of buffer space). Note that this counter would include datagrams counted in ipForwDatagrams if any such packets met this (discretionary) discard criterion. (counter)

ipOutNoRoutes – The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in ipForwDatagrams which meet this “no-route” criterion. Note that this includes any datagrams that a host cannot route because all its default gateways are down. (counter)

ipReasmTimeout – The maximum number of seconds that received fragments are held while they are awaiting reassembly at this entity. (int)

ipReasmReqds – The number of IP fragments received that needed to be reassembled at this entity. (counter)

ipReasmOKs – The number of IP datagrams successfully reassembled. (counter)

ipReasmFails – The number of failures detected by the IP reassembly algorithm, for whatever reason: timed out, errors, and the like. Note that this is not necessarily a count of discarded IP fragments since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received. (counter)

ipFragOKs – The number of IP datagrams that have been successfully fragmented at this entity. (counter)

ipFragFails – The number of IP datagrams that have been discarded because they needed to be fragmented at this entity but could not be, for example, because their “Don’t Fragment” flag was set. (counter)

ipFragCreates – The number of IP datagram fragments that have been generated as a result of fragmentation at this entity. (counter)

ipRoutingDiscards – The number of routing entries that were chosen to be discarded even though they were valid. One possible reason for discarding such an entry could be to free-up buffer space for other routing entries. (counter) Returns a fixed value of 0.

ipAddrTable

ipAddrTable is a table of addressing information relevant to this entity’s IP addresses.

ipAdEntAddr – The IP address to which this entry’s addressing information pertains. (netaddress)

ipAdEntIfIndex – The index value that uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

ipAdEntNetMask – The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1, and all the hosts bits set to 0. (netaddress)

ipAdEntBcastAddr – The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface. (int) Returns a fixed value of 1.

ipAdEntReasmMaxSize – The size of the largest IP datagram that this entity can reassemble from incoming IP fragmented datagrams received on this interface. (int) Returns a fixed value of 65535.

ipRouteTable

The **ipRouteTable** is this entity's IP Routing table.

ipRouteDest – The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table- access mechanisms defined by the network management protocol in use. (netaddress)

ipRouteIfIndex – The index value that uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

ipRouteMetric1 – The primary routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

ipRouteMetric2 – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

ipRouteMetric3 – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

ipRouteMetric4 – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

ipRouteNextHop – The IP address of the next hop of this route. (In the case of a route bound to an interface that is realized via a broadcast media, the value of this field is the agent's IP address on that interface.) (netaddress)

ipRouteType – The type of route. Note that the values direct (3) and indirect (4) refer to the notion of direct and indirect routing in the IP architecture. (enum)

Setting this object to the value invalid (2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively dissociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.

ipRouteProto – The routing mechanism through which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols. (enum)

ipRouteAge – The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of “too old” can be implied except through knowledge of the routing protocol by which the route was learned. (int) Returns a fixed value of 0.

ipRouteMask – Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belongs to a class-A, B, or C network, and then using one of:

mask	**network**
255.0.0.0	class-A
255.255.0.0	class-B
255.255.255.0	class-C

If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism. (netaddress)

ipRouteMetric5 – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route’s ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

ipRouteInfo – A reference to MIB definitions specific to the particular routing protocol responsible for this route, as determined by the value specified in the route’s ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier. Any conformant implementation of ASN.1 and BER must be able to generate and recognize this value. (objectid)

ipNetToMediaTable

The **ipNetToMediaTable** is the IP Address Translation table used for mapping from IP addresses to physical addresses.

ipNetToMediaIfIndex – The interface on which this entry’s equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

ipNetToMediaPhysAddress – The media-dependent physical address. (octet[128])

ipNetToMediaNetAddress – The IpAddress corresponding to the media- dependent physical address. (netaddress)

ipNetToMediaType – The type of mapping. (enum) Returns a fixed value of (3)dynamic. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively dissociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.

icmp

The **icmp** group reports statistics about the ICMP group.

icmpInMsgs – The total number of ICMP messages that the entity received. Note that this counter includes all those counted by icmpInErrors. (counter)

icmpInErrors – The number of ICMP messages that the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, and the like.). (counter)

icmpInDestUnreachs – The number of ICMP Destination Unreachable messages received. (counter)

icmpInTimeExcds – The number of ICMP Time Exceeded messages received. (counter)

icmpInParmProbs – The number of ICMP Parameter Problem messages received. (counter)

icmpInSrcQuenchs – The number of ICMP Source Quench messages received. (counter)

icmpInRedirects – The number of ICMP Redirect messages received. (counter)

icmpInEchos – The number of ICMP Echo (request) messages received. (counter)

icmpInEchoReps – The number of ICMP Echo Reply messages received. (counter)

icmpInTimestamps – The number of ICMP Timestamp (request) messages received. (counter)

icmpInTimestampReps – The number of ICMP Timestamp Reply messages received. (counter)

icmpInAddrMasks – The number of ICMP Address Mask Request messages received. (counter)

icmpInAddrMaskReps – The number of ICMP Address Mask Reply messages received. (counter)

icmpOutMsgs – The total number of ICMP messages that this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors. (counter)

icmpOutErrors – The number of ICMP messages that this entity did not send due to problems discovered within ICMP, such as a lack of buffers. This value should not include errors discovered outside the ICMP layer, such as the inability of IP to route the resultant datagram. In some implementations there may be no types of errors that contribute to this counter's value. (counter)

icmpOutDestUnreachs – The number of ICMP Destination Unreachable messages sent. (counter)

icmpOutTimeExcds – The number of ICMP Time Exceeded messages sent. (counter)

icmpOutParmProbs – The number of ICMP Parameter Problem messages sent. (counter)

icmpOutSrcQuenchs – The number of ICMP Source Quench messages sent. (counter)

icmpOutRedirects – The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects. (counter)

icmpOutEchos – The number of ICMP Echo (request) messages sent. (counter)

icmpOutEchoReps – The number of ICMP Echo Reply messages sent. (counter)

icmpOutTimestamps – The number of ICMP Timestamp (request) messages sent. (counter)

icmpOutTimestampReps – The number of ICMP Timestamp Reply messages sent. (counter)

icmpOutAddrMasks – The number of ICMP Address Mask Request messages sent. (counter)

icmpOutAddrMaskReps – The number of ICMP Address Mask Reply messages sent. (counter)

tcp The **tcp** group reports statistics about the TCP group.

tcpRtoAlgorithm – The algorithm used to determine the timeout value used for retransmitting unacknowledged octets. (enum)

tcpRtoMin – The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793. (int)

tcpRtoMax – The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793. (int)

cpMaxConn – The limit on the total number of TCP connections that the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value –1. (int)

tcpActiveOpens – The number of times that TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state. (counter)

tcpPassiveOpens – The number of times that TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state. (counter)

tcpAttemptFails – The number of times that TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state. (counter)

tcpEstabResets – The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state. (counter)

tcpCurrEstab – The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT. (gauge)

tcpInSegs – The total number of segments received, including those received in error. This count includes segments received on currently established connections. (counter)

tcpOutSegs – The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets. (counter)

tcpRetransSegs – The total number of segments retransmitted - that is, the number of TCP segments transmitted containing one or more previously transmitted octets. (counter)

tcpInErrs – The total number of segments received in error (for example, bad TCP checksums). (counter)

tcpOutRsts – The number of TCP segments sent containing the RST flag. (counter)

tcpConnTable

The **tcpConnTable** is a table containing TCP connection-specific information.

tcpConnState – The state of this TCP connection. (enum)

The only value that may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a “badValue” response if a management station attempts to set this object to any other value.

If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node. This results in immediate termination of the connection.

As an implementation-specific option, an RST segment may be sent from the managed node to the other TCP endpoint. (Note, however, that RST segments are not sent reliably.)

tcpConnLocalAddress – The local IP address for this TCP connection. For a connection in the listen state that is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used. (netaddress)

tcpConnLocalPort – The local port number for this TCP connection. (int)

tcpConnRemAddress – The remote IP address for this TCP connection. (netaddress)

tcpConnRemPort – The remote port number for this TCP connection. (int)

upd

The **udp** group reports statistics about the UDP group.

udpInDatagrams – The total number of UDP datagrams delivered to UDP users. (counter) Returns a fixed value of 0.

udpNoPorts – The total number of received UDP datagrams for which there was no application at the destination port. (counter) Returns a fixed value of 0.

udpInErrors – The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port. (counter)

udpOutDatagrams – The total number of UDP datagrams sent from this entity. (counter) Returns a fixed value of 0.

udpTable

The **udpTable** is a table containing UDP listener information.

udpLocalAddress – The local IP address for this UDP listener. For a UDP listener that is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used. (netaddress)

udpLocalPort – The local port number for this UDP listener. (int)

snmp

The **snmp** group reports statistics about the SNMP group.

snmpInPkts – The total number of Messages delivered to the SNMP entity from the transport service. (counter)

snmpOutPkts – The total number of SNMP Messages passed from the SNMP protocol entity to the transport service. (counter)

snmpInBadVersions – The total number of SNMP Messages delivered to the SNMP protocol entity that were for an unsupported SNMP version. (counter)

snmpInBadCommunityNames – The total number of SNMP Messages delivered to the SNMP protocol entity that used a SNMP community name not known to said entity. (counter)

snmpInBadCommunityUses – The total number of SNMP Messages delivered to the SNMP protocol entity, which represented an SNMP operation not allowed by the SNMP community named in the Message. (counter)

snmpInASNParseErrs – The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP Messages. (counter)

snmpInTooBigs – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “tooBig.” (counter)

snmpInNoSuchNames – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “noSuchName.” (counter)

snmpInBadValues – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “badValue.” (counter)

snmpInReadOnlys – The total number valid SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “readOnly.” It should be noted that it is a protocol error to generate an SNMP PDU that contains the value “readOnly” in the error-status field. This object is provided as a means of detecting incorrect implementations of the SNMP. (counter)

snmpInGenErrs – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “genErr.” (counter)

snmpInTotalReqVars – The total number of MIB objects successfully retrieved by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs. (counter)

snmpInTotalSetVars – The total number of MIB objects successfully altered by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs. (counter)

snmpInGetRequests – The total number of SNMP Get-Request PDUs accepted and processed by the SNMP protocol entity. (counter)

snmpInGetNexts – The total number of SNMP Get-Next PDUs accepted and processed by the SNMP protocol entity. (counter)

snmpInSetRequests – The total number of SNMP Set-Request PDUs accepted and processed by the SNMP protocol entity. (counter)

snmpInGetResponses – The total number of SNMP Get-Response PDUs accepted and processed by the SNMP protocol entity. (counter)

snmpInTraps – The total number of SNMP Trap PDUs accepted and processed by the SNMP protocol entity. (counter)

snmpOutTooBigs – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is “tooBig.” (counter)

snmpOutNoSuchNames – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status is “noSuchName.” (counter)

snmpOutBadValues – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is “badValue.” (counter)

snmpOutGenErrs – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is “genErr.” (counter)

snmpOutGetRequests – The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity. (counter)

snmpOutGetNexts – The total number of SNMP Get-Next PDUs generated by the SNMP protocol entity. (counter)

snmpOutSetRequests – The total number of SNMP Set-Request PDUs generated by the SNMP protocol entity. (counter)

snmpOutGetResponses – The total number of SNMP Get-Response PDUs generated by the SNMP protocol entity. (counter)

snmpOutTraps – The total number of SNMP Trap PDUs generated by the SNMP protocol entity. (counter)

snmpEnableAuthenTraps – Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any configuration information. As such, it provides a means whereby all authentication-failure traps may be disabled. (enum)

Note that this object must be stored in non-volatile memory, so that it remains constant between reinitializations of the network management system.

The following are Sun-specific group and table definitions.

sunSystem

The **sunSystem** group reports general system information.

agentDescr – The SNMP agent’s description of itself. (string[255])

hostID – The unique Sun hardware identifier. The value returned is four byte binary string. (octet[4])

motd – The first line of **/etc/motd**. (string[255])

unixTime – The UNIX system time. Measured in seconds since January 1, 1970 GMT. (counter)

sunProcessTable

The **sunProcessTable** table reports UNIX process table information.

psProcessID – The process identifier for this process. (int)

psParentProcessID – The process identifier of this process's parent. (int)

psProcessSize – The combined size of the data and stack segments (in kilobytes.) (int)

psProcessCpuTime – The CPU time (including both user and system time) consumed so far. (int)

psProcessState – The run-state of the process. (octet[4])

R	Runnable
T	Stopped
P	In page wait
D	Non-interruptable wait
S	Sleeping (less than 20 seconds)
I	Idle (more than 20 seconds)
Z	Zombie

psProcessWaitChannel – Reason process is waiting. (octet[16])

psProcessTTY – Terminal, if any, controlling this process. (octet[16])

psProcessUserName – Name of the user associated with this process. (octet[16])

psProcessUserID – Numeric form of the name of the user associated with this process. (int)

psProcessName – Command name used to invoke this process. (octet[64])

psProcessStatus – Setting this variable will cause a signal of the set value to be sent to the process. (int)

sunHostPerf

The **sunHostPerf** group reports hostperf information.

rsUserProcessTime – Total number of timeticks used by user processes since the last system boot. (counter)

rsNiceModeTime – Total number of timeticks used by "nice" mode since the last system boot. (counter)

rsSystemProcessTime – Total number of timeticks used by system processes since the last system boot. (counter)

rsIdleModeTime – Total number of timeticks in idle mode since the last system boot. (counter)

rsDiskXfer1 – Total number of disk transfers since the last boot for the first of four configured disks. (counter)

rsDiskXfer2 – Total number of disk transfers since the last boot for the second of four configured disks. (counter)

rsDiskXfer3 – Total number of disk transfers since the last boot for the third of four configured disks. (counter)

rsDiskXfer4 – Total number of disk transfers since the last boot for the fourth of four configured disks. (counter)

rsVPagesIn – Number of pages read in from disk. (counter)

rsVPagesOut – Number of pages written to disk. (counter)

rsVSwapIn – Number of pages swapped in. (counter)

rsVSwapOut – Number of pages swapped out. (counter)

rsVIntr – Number of device interrupts. (counter)

rsIfInPackets – Number of input packets. (counter)

rsIfOutPackets – Number of output packets. (counter)

rsIfInErrors – Number of input errors. (counter)

rsIfOutErrors – Number of output errors. (counter)

rsIfCollisions – Number of output collisions. (counter)

FILES

/etc/snmp/conf/snmpd.conf	configuration information
/var/snmp/mib/sun.mib	standard SNMP MIBII file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmibii

SEE ALSO

inetd(1M), **select**(3C), **recvfrom**(3N), **sendto**(3N), **attributes**(5), **gld**(7D),

DIAGNOSTICS

cannot dispatch request

The proxy cannot dispatch the request. The rest of the message indicates the cause of the failure.

select(3C) failed

A **select**(3C) call failed. The rest of the message indicates the cause of the failure.

sendto(3N) failed

A **sendto**(3N) call failed. The rest of the message indicates the cause of the failure.

recvfrom(3N) failed

A **recvfrom**(3N) call failed. The rest of the message indicates the cause of the failure.

no response from system

The SNMP agent on the target system does not respond to SNMP requests. This error might indicate that the SNMP agent is not running on the target system, the target system is down, or the network containing the target system is unreachable.

response too big

The agent could not fit the results of an operation into a single SNMP message. Split large groups or tables into smaller entities.

missing attribute
An attribute is missing from the requested group.

bad attribute type
An object attribute type received from the SNMP agent that does not match the attribute type specified by the proxy agent schema. The rest of the message indicates the expected type and received type.

cannot get sysUpTime
The proxy agent cannot get the variable *sysUpTime* from the SNMP agent.

sysUpTime type bad
The variable *sysUpTime* received from the SNMP agent has the wrong data type.

unknown SNMP error
An unknown SNMP error was received.

bad variable value
The requested specified an incorrect syntax or value for a set operation.

variable is read only
The SNMP agent did not perform the set request because a variable to set may not be written.

general error
A general error was received.

cannot make request PDU
An error occurred building a request PDU.

cannot make request varbind list
An error occurred building a request variable binding list.

cannot parse response PDU
An error occurred parsing a response PDU.

request ID - response ID mismatch
The response ID does not match the request ID.

string contains non-displayable characters
A displayable string contains non-displayable characters.

cannot open schema file
An error occurred opening the proxy agent schema file.

cannot parse schema file
The proxy agent couldn't parse the proxy agent schema file.

cannot open host file
An error occurred opening the file associated with the *na.snmp.hostfile* keyword in **/etc/snmp/conf/snmpd.conf** for Solaris 2.4, 2.5, 2.6.

cannot parse host file
The proxy agent was unable to parse the file associated with the *na.snmp.hostfile* keyword in **/etc/snmp/conf/snm.conf** for Solaris 2.4, 2.5, 2.6.

attribute unavailable for set operations

The set could not be completed because the attribute was not available for set operations.

BUGS

The **mibiisa** utility returns the wrong interface speed for the SBUS FDDI interface (for example, “bf0”).

The **mibiisa** utility does not return a MAC address for the SBUS FDDI interface (for example, “bf0”).

Process names retrieved from **mibiisa** contain a leading blank space.

When you change attribute values in the system group with an SNMP set request, the change is effective only as long as **mibiisa** is running. **mibiisa** does not save the changes to **/etc/snmp/conf/snmpd.conf** for Solaris 2.4, 2.5, and 2.6.

NAME

mk – remake the binary system and commands from source code

DESCRIPTION

All source code for the UNIX system is distributed in the directory **/usr/src**. The directory tree rooted at **/usr/src** includes source code for the operating system, libraries, commands, miscellaneous data files necessary for the system and procedures to transform this source code into an executable system.

Within the **/usr/src** directory are the **cmd**, **lib**, **uts**, **head**, and **stand** directories, as well as commands to remake the parts of the system found under each of these sub-directories. These commands are named **:mk** and **:mk***dir* where *dir* is the name of the directory to be recreated. Each of these **:mk***dir* commands rebuilds all or part of the directory it is responsible for. The **:mk** command runs each of the other commands in order and thus recreates the whole system. The **:mk** command is distributed only to source code licensees.

Each command, with its associated directory, is described below.

:mklib The **lib** directory contains the source code for the system libraries. The most important of these is the C library. Each library is in its own subdirectory. If any arguments are specified on the **:mklib** command line then only the given libraries are rebuilt. The argument * causes it to rebuild all libraries found under the **lib** directory.

:mkhead The **head** directory contains the source code versions of the headers found in the **/usr/include** directory. The **:mkhead** command installs the headers given as arguments. The argument * causes it to install all headers.

:mkuts The **uts** directory contains the source code for the UNIX Operating System. The **:mkuts** command takes no arguments and invokes a series of makefiles that recreates the operating system.

Associated with the operating system is a set of headers that describe the user interface to the operating system. The source for these headers is found in a sub-directory within the **uts** directory tree. The user-accessible versions of these headers are found in the **/usr/include/sys** directory. The **:mksyshead** command installs these headers into the **/usr/include/sys** directory.

:mkstand The **stand** directory contains stand-alone commands and boot programs. The **:mkstand** command rebuilds and installs these programs. Note that these stand-alone programs are only applicable to the DEC processors and are not built for any other machine.

:mkcmd The **cmd** directory contains the source code for all the commands available on the system. There are two types of entries within the **cmd** directory: commands whose source code consists of only one file with one of the following suffixes: **.l**, **.y**, **.c**, **.s**, **.sh**, or a sub-directory that contains the multiple source files that comprise a particular command or subsystem.

Each sub-directory is assumed to have a makefile (see **make**(1S)) with the name *command***.mk** that takes care of creating everything associated with that directory and its sub-directories.

The **:mkcmd** command transforms source code into an executable command based on a set of predefined rules. If the **:mkcmd** command encounters a sub-directory within the **cmd** directory then it runs the makefile found in that sub-directory. If no makefile is found then an error is reported. For single-file commands, the predefined rules are dependent on the file's suffix. C programs (**.c**) are compiled by the C compiler and loaded stripped with shared text. Assembly language programs (**.s**) are assembled and loaded stripped. Yacc programs (**.y**) and lex programs (**.l**) are processed by **yacc**( ) and **lex**( ) respectively, before C compilation. Shell programs (**.sh**) are copied to create the command. Each of these operations leaves a command in the **./cmd** directory which is then installed into a user-accessible directory by using **/usr/sbin/install**.

The arguments to **:mkcmd** are either command names or subsystem names. Some subsystems distributed with the UNIX system are: **acct**, **graf**, **sgs**, **sccs**, and **text**. Prefacing the **:mkcmd** command with an assignment to the shell variable **$ARGS** causes the indicated components of the subsystem to be rebuilt.

For example, the entire **sccs** subsystem can be rebuilt by:

```
/usr/src/:mkcmd sccs
```

while the **delta** component of **sccs** can be rebuilt by:

```
ARGS="delta" /usr/src/:mkcmd sccs
```

The **log** command, which is a part of the **stat** package, which is itself a part of the **graf** package, can be rebuilt by:

```
ARGS="stat log" /usr/src/:mkcmd graf
```

The argument * causes all commands and subsystems to be rebuilt.

Makefiles throughout the system, and particularly in the **cmd** directory, have a standard format. In particular, **:mkcmd** depends on each makefile having target entries for **install** and **clobber**. The **install** target should cause everything over which the makefile has jurisdiction to be built and installed by **/usr/sbin/install**. The **clobber** target should cause a complete cleanup of all unnecessary files resulting from the previous invocation. The commands that use the **CLOBBER** environment variable are **:mkcmd**, **:mklib**, and **:mkuts**. These commands all check the **CLOBBER** variable before executing **make clobber**. If this variable is set to **OFF**, then **make clobber** is *not* performed. If the variable is *not* set or is set to anything other than **OFF**, the **make clobber** *is* performed.

An effort has been made to separate the creation of a command from source and its installation on the running system. The command **/usr/sbin/install** is used by **:mkcmd** and most makefiles to install commands in standard directories on the system. The use of **install** allows maximum flexibility in the administration of the system. The **install** command makes very few assumptions about where a command is located, who owns it, and what modes are in effect. All assumptions may be overridden on invocation of the command, or more permanently by redefining a few variables in **install**. The purpose of **install** is to install a new version of a command in the same place, with the same attributes as the prior version.

In addition, the use of a separate command to perform installation allows for the creation of test systems in other than standard places, easy movement of commands to balance load, and independent maintenance of makefiles.

SEE ALSO

install(1M), **make**(1S)

NAME

mkfifo – make FIFO special file

SYNOPSIS

/usr/bin/mkfifo [**–m** *mode*] *path* . . .

DESCRIPTION

The **mkfifo** command creates the FIFO special files named by its argument list. The arguments are taken sequentially, in the order specified; and each FIFO special file is either created completely or, in the case of an error or signal, not created at all.

If errors are encountered in creating one of the special files, **mkfifo** writes a diagnostic message to the standard error and continues with the remaining arguments, if any.

The **mkfifo** command calls the library routine **mkfifo**(3C), with the *path* argument is passed as the *path* argument from the command line, and *mode* is set to the equivalent of **a=rw**, modified by the current value of the file mode creation mask **umask**(1).

OPTIONS

The following option is supported:

–m *mode* Set the file permission bits of the newly-created FIFO to the specified *mode* value. The *mode* option-argument will be the same as the *mode* operand defined for the **chmod**(1) command. In <*symbolic*mode> strings, the *op* characters **+** and – will be interpreted relative to an assumed initial mode of **a=rw**.

OPERANDS

The following operand is supported:

file A path name of the FIFO special file to be created.

USAGE

See **largefile**(5) for the description of the behavior of **mkfifo** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **mkfifo**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 All the specified FIFO special files were created successfully.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

SEE ALSO

mkfifo(3C), **attributes**(5), **environ**(5), **largefile**(5)

NAME

mkfile – create a file

SYNOPSIS

mkfile [**–nv**] *size*[**k** | **b** | **m**] *filename* . . .

DESCRIPTION

mkfile creates one or more files that are suitable for use as NFS-mounted swap areas, or as local swap areas. When a root user executes **mkfile( )**, the sticky bit is set and the file is padded with zeros by default. When non-root users execute **mkfile( )**, they must manually set the sticky bit using **chmod**(1). The default *size* is in bytes, but it can be flagged as kilobytes, blocks, or megabytes, with the **k**, **b**, or **m** suffixes, respectively.

OPTIONS

–n Create an empty *filename*. The size is noted, but disk blocks are not allocated until data is written to them. Files created with this option cannot be swapped over local UFS mounts.

–v Verbose. Report the names and sizes of created files.

USAGE

See **largefile**(5) for the description of the behavior of **mkfile** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

chmod(1), **swap**(1M), **attributes**(5), **largefile**(5)

NAME

mkfs – construct a file system

SYNOPSIS

mkfs [–**F** *FSType*] [*generic_options*] [–**o** *FSType-specific_options*] *raw_device_file* [*operands*]

DESCRIPTION

mkfs constructs a file system on the the *raw_device_file* by calling the specific **mkfs** module indicated by –**F** *FSType***.**

Note: ufs file systems are normally created with the **newfs**(1M) command.

generic_options are independent of file system type. *FSType-specific_options* is a comma-separated list of *keyword=value* pairs (with no intervening spaces), which are *FSType*-specific. *raw_device_file* specifies the disk partition on which to write the file system. It is required and must be the first argument following the *specific_options* (if any). *operands* are *FSType*-specific. See the *FSType*-specific manual page of **mkfs** (for example, **mkfs_ufs**(1M)) for a detailed description.

OPTIONS

The following are the generic options for **mkfs**:

–**F** Specify the *FSType* to be constructed. If –**F** is not specified, the *FSType* is determined from **/etc/vfstab** by matching the *raw_device_file* with a **vfstab** entry, or by consulting the **/etc/default/fs** file.

–**V** Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided and adding to them information derived from **/etc/vfstab** or **/etc/default/fs**. This option may be used to verify and validate the command line.

–**m** Return the command line which was used to create the file system. The file system must already exist. This option provides a means of determining the command used in constructing the file system.

–**o** Specify *FSType*-specific options. See the manual page for the **mkfs** module specific to the file system type.

USAGE

See **largefile**(5) for the description of the behavior of **mkfs** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/default/fs Default file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: LOCAL=ufs

LOCAL: The default partition for a command if no FSType is specified.

/etc/vfstab List of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

newfs(1M), **vfstab**(4), **attributes**(5), **largefile**(5)
Manual pages for the *FSType*-specific modules of **mkfs**.

NOTES

This command may not be supported for all *FSTypes*.

NAME

mkfs_ufs – construct a ufs file system

SYNOPSIS

mkfs –F ufs [*generic_options*] [**–o** *FSType_specific_options*] *raw_device_file* [*size*]

DESCRIPTION

The ufs-specific module of **mkfs** builds a ufs file system with a root directory and a **lost+found** directory (see **fsck**(1M))**.**

Note: The ufs-specific **mkfs** is rarely run directly; use the **newfs**(1M) command instead.

raw_device_file indicates the disk partition to write on unless the **–o N** option has been specified, or either the **–V** or **–m** generic options are passed from the generic **mkfs** module. *size* specifies the number of sectors in the file system. This argument must follow the *raw_device_file* argument and is required (even with **–o N**), unless the **–V** or **–m** generic options are specified.

generic_options are supported by the generic **mkfs** command. See **mkfs**(1M) for a description of these options.

OPTIONS

–o Use one or more of the following values separated by commas (with no intervening spaces) to specify ufs-specific options:

N Print out the file system parameters without actually creating the file system.

nsect=*n* The number of sectors per track on the disk. The default is **32**.

ntrack=*n* The number of tracks per cylinder on the disk. The default is **16**.

bsize=*n* Logical block size, either **4096** or **8192**. The default is **8192**. (Note: The sun4u architecture does not support the **4096** block size.)

fragsize=*n* The smallest amount of disk space in bytes to allocate to a file. The value must be a power of 2 selected from the range 512 to the logical block size. If logical block size is 4096, legal values are 512, 1024, 2048 and 4096; if logical block size is 8192, 8192 is also a legal value. The default is **1024**.

cgsize=*n* The number of cylinders per cylinder group. The default is **16**.

free=*n* The minimum percentage of free space to maintain in the file system. This space is off-limits to normal users. Once the file system is filled to this threshold, only the superuser can continue writing to the file system. This parameter can be subsequently changed using the **tunefs**(1M) command. The default is **10%**.

rps=*n* The rotational speed of the disk, in revolutions per second. The default is **60**.

nbpi=*n* The number of bytes per inode, which specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the fixed number of inodes to create. It should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used; to create more inodes, a smaller number should be given. The default is **2048**.

opt=*a* Space or time optimization preference; **s** specifies optimization for space, **t** specifies optimization for time. The default is **t**. This parameter may be subsequently changed with the **tunefs**(1M) command.

apc=*n* The number of alternates per cylinder to reserve for bad block replacement (SCSI devices only). The default is **0**.

gap=*n* Rotational delay. The expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. The value is used to decide how much rotational spacing to place between successive blocks in a file. This parameter can be subsequently changed using the **tunefs**(1M) command. The default is disk-type dependent.

nrpos=*n* The number of different rotational positions in which to divide a cylinder group. The default is **8**.

maxcontig=*n* The maximum number of blocks, belonging to one file, that will be allocated contiguously before inserting a rotational delay. For a 4K file system, the default is **14**; for an 8K file system it is **7**. This parameter can be subsequently changed using the **tunefs**(1M) command.

Note: This parameter also controls clustering. Regardless of the value of *gap*, clustering is enabled only when **maxcontig** is greater than 1. Clustering allows higher I/O rates for sequential I/O and is described in **tunefs**(1M).

Alternatively, parameters can be entered as a list of space-separated values (without keywords) whose meaning is positional. In this case, the −**o** option is omitted and the list follows the size operand. This is the way **newfs** passes the parameters to **mkfs**.

OPERANDS

raw_device_file The disk partition on which to write.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fsck(1M), **mkfs**(1M), **newfs**(1M), **tunefs**(1M), **dir_ufs**(4), **fs_ufs**(4), **attributes**(5)

DIAGNOSTICS

Warning: insufficient space in super block for rotational layout tables with nsect *sblock.fs_nsect* **and ntrak** *sblock.fs_ntrak*. **(File system performance may be impaired.)**

Occurs typically on very high density disks. On such disks, the file system structure cannot encode the proper disk layout information, resulting in suboptimal performance.

Warning: inode blocks/cyl group (*grp***) >= data blocks (***num***) in last cylinder**

User request for inodes / byte (with the **nbpi** keyword) and the disk geometry results in a situation in which the last truncated cylinder group cannot contain the correct number of data blocks; some disk space is wasted.

Warning: *num* **sector(s) in last cylinder group unallocated**

User parameters and disk geometry conflict; some disk space is lost. A possible cause is the specified size being smaller than the partition size.

NAME

mknod – make a special file

SYNOPSIS

mknod *name* **b** *major minor*
mknod *name* **c** *major minor*
mknod *name* **p**

DESCRIPTION

mknod makes a directory entry for a special file.

OPTIONS

The following options are supported:

b Create a block-type special file.

c Create a character-type special file.

p Create a FIFO (named pipe).

OPERANDS

The following operands are supported:

major The *major* device number.

minor The *minor* device number; can be either decimal or octal. The assignment of major device numbers is specific to each system. You must be the super-user to use this form of the command.

name A special file to be created.

USAGE

See **largefile**(5) for the description of the behavior of **mknod** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ftp(1), **in.ftpd**(1M), **mknod**(2), **symlink**(2), **attributes**(5), **largefile**(5)

NOTES

If **mknod**(2) is used to create a device, the major and minor device numbers are always interpreted by the kernel running on that machine.

With the advent of physical device naming, it would be preferable to create a symbolic link to the physical name of the device (in the **/devices** subtree) rather than using **mknod**.

NAME

modinfo – display information about loaded kernel modules

SYNOPSIS

/usr/sbin/modinfo [**–i** *module-id*]

DESCRIPTION

modinfo displays information about the loaded modules. The format of the information is as follows:

Id Loadaddr Size Info Rev Module Name

where *Id* is the module ID, *Loadaddr* is the starting text address in hex, *size* is the size of text, data, and bss in hex bytes, *Info* is module specific info, *Rev* is the revision of the loadable modules system, and *Module Name* is the filename and description of the module.

The module specific information is the block and character major numbers for drivers, the system call number for system calls, or, for other module types, the index into the appropriate kernel table:

fmodsw	for streams modules
vfssw	for filesystems
class	for scheduling classes
execsw	for **exec** modules

OPTIONS

–i *module-id* Display information about this module only.

EXAMPLES

The following example displays the status of module **2**:

```
example% modinfo –i 2
Id   Loadaddr    Size   Info   Rev   Module Name
2    ff08e000    1734    –      1    swapgeneric
                                     (root and swap configuration)
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

modload(1M), **modunload**(1M), **attributes**(5)

NAME

modload – load a kernel module

SYNOPSIS

modload [−**p**] [−**e** *exec_file*] *filename*

DESCRIPTION

modload loads the loadable module *filename* into the running system. *filename* is an object file produced by **ld** −**r**. If *filename* is an absolute pathname then the file specified by that absolute path is loaded. If *filename* does not begin with a ’/’ then the path to load *filename* is relative to the current directory unless the −**p** option is specified. The kernel’s modpath variable can be set using the **/etc/system** file. The default value of the kernel’s modpath variable is set to the path where the operating system was loaded. Typically this is **/kernel /usr/kernel**. Hence if you type:

```
example# modload drv/foo
```

Then the kernel will look for **./drv/foo**.

If you type:

```
example# modload −p drv/foo
```

Then the kernel will look for **/kernel/drv/foo** and then **/usr/kernel/drv/foo**.

OPTIONS

−**p** Use the kernel’s internal modpath variable as the search path for the module.

−**e** *exec_file* Specify the name of a shell script or executable image file that is executed after the module is successfully loaded. The first argument passed is the module ID (in decimal). The other argument is module specific. The module specific information is: the block and character major numbers for drivers, the system call number for system calls, or, for other module types, the index into the appropriate kernel table. See **modinfo**(1M)**.**

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ld(1), **add_drv**(1M), **kernel**(1M), **modinfo**(1M), **modunload**(1M), **system**(4), **attributes**(5), **modldrv**(9S), **modlinkage**(9S), **modlstrmod**(9S), **module_info**(9S)

Writing Device Drivers
Solaris 1.x to 2.x Transition Guide

NOTES

Use **add_drv**(1M) to add device drivers, not modload**.** See *Writing Device Drivers* for procedures on adding device drivers.

NAME

modunload – unload a module

SYNOPSIS

modunload **–i** *module_id* [**–e** *exec_file*]

DESCRIPTION

modunload unloads a loadable module from the running system. The *module_id* is the ID of the module as shown by **modinfo**(1M). If ID is **0**, all modules that were autoloaded which are unloadable, are unloaded. Modules loaded by **modload**(1M) are not affected.

OPTIONS

–i *module_id* Specify the module to be unloaded.

–e *exec_file* Specify the name of a shell script or executable image file to be executed before the module is unloaded. The first argument passed is the module id (in decimal). There are two additional arguments that are module specific. For loadable drivers, the second and third arguments are the block major and character major numbers respectively. For loadable system calls, the second argument is the system call number. For loadable exec classes, the second argument is the index into the **execsw** table. For loadable filesystems, the second argument is the index into the **vfssw** table. For loadable streams modules, the second argument is the index into the **fmodsw** table. For loadable scheduling classes, the second argument is the index into the class array. Minus one is passed for an argument that does not apply.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

modinfo(1M), **modload**(1M), **attributes**(5)

NAME

monitor – SPARC system PROM monitor

SYNOPSIS

L1–A
BREAK
initial system power-on
exit from a client program, e.g., the Operating System

DESCRIPTION

The CPU board of a workstation contains one or more EPROMs or EEPROMs. The program which executes from the PROMs is referred to as “the monitor”. Among other things, the monitor performs system initialization at power-on and provides a user interface.

Monitor Prompt

The monitor of earlier workstations was known as the **SunMON** monitor and displayed the **>** for its prompt. See the **SunMON MONITOR USAGE** section for further details.

Existing workstations use a monitor which is known as the OpenBoot monitor. The OpenBoot monitor typically displays **ok** as its prompt, but it may also display the **>** prompt under certain circumstances.

If the **’auto-boot?’** NVRAM parameter is set to **’false’** when the workstation is powered on then the system will not attempt to boot and the monitor will issue its prompt. If **’auto-boot’** is set to **’true’** then the system will initiate the boot sequence. The boot sequence can be aborted by simultaneously pressing two keys on the system’s keyboard: L1 and A (on older keyboards), or Stop and A (on newer keyboards). Note that either a lower case **’a’** or an upper case **’A’** will work for the keyboard abort sequence. If a console has been attached via one of the system’s serial ports then the abort sequence can be accomplished by sending a BREAK – see the **tip**(1) manpage.

When the NVRAM **’security-mode’** parameter has been turned on, or when the value of the **’sunmon-compat?’** parameter is true, then the OpenBoot monitor will display the message:

Type b (boot), c (continue), or n (new command mode)

and the **>** prompt will appear.

OPENBOOT PROM USAGE

Some of the more useful commands that can be issued from OpenBoot’s **ok** prompt are described here. Refer to the *OpenBoot 2.x Command Reference Manual* book for a complete list of commands.

Help

Help for various functional areas of the OpenBoot monitor can be obtained by typing **help**. The help listing will provide a number of other key words which can then be used in the help command to provide further details.

NVRAM Parameters

Each workstation contains one or more NVRAM devices which contains unique system ID information, as well as a set of user-configurable parameters. The NVRAM parameters allow the user a certain level of flexibility in configuring the system to act in a given manner under a specific set of circumstances.

See the **eeprom**(1M) manpage for a description of the parameters. This manpage also describes a way of setting the parameters from the OS level.

The following commands can be used at the OpenBoot monitor to access the NVRAM parameters.

printenv Used to list the NVRAM parameters, along with their default values and current values.

setenv *pn pv* Used to set or modify a parameter. The *pn* represents the parameter name, and *pv* represents the parameter value.

set-default *pn* Used to set an individual parameter back to its default value.

set-defaults Used to reset all parameters to their default values. (Note that **'set-defaults'** only affects parameters that have assigned default values.)

Hardware Checks and Diagnostics

The following commands are available for testing or checking the system's hardware. If the **'diag-switch?'** NVRAM parameter is set to true when the system is powered on, then a Power-On Self Test (POST) diagnostic will be run, if present, sending its results messages to the system's serial port A. Not all of the commands shown are available on all workstations.

test-all Run the diagnostic tests on each device which has provided a self-test.

test floppy Run diagnostics on the system's floppy device.

test /memory Run the main memory tests. If the NVRAM parameter **'diag-switch?'** is set to true, then all of main memory is tested. If the parameter is false then only the amount of memory specified in the **'selftest-#megs'** NVRAM parameter will be tested.

test net Test the network connection for the on-board network controller.

watch-net Monitor the network attached to the on-board net controller.

watch-net-all Monitor the network attached to the on-board net controller, as well as the network controllers installed in SBus slots.

watch-clock Test the system's clock function.

System Information

The following commands are available for displaying information about the system. Not all commands are available on all workstations.

banner Display the power-on banner.
.enet-addr Display the system's Ethernet address.
.idprom Display the formatted contents of the IDPROM.
module-info Display information about the system's processor(s).
probe-scsi Identify the devices attached to the on-board SCSI controller.
probe-scsi-all Identify the devices attached to the on-board SCSI controller as well as those devices which are attached to SBus SCSI controllers.
show-disks Display a list of the device paths for installed SCSI disk controllers.
show-displays Display a list of the device paths for installed display devices.

show-nets Display a list of the device paths for installed Ethernet controllers.
show-sbus Display list of installed SBus devices.
show-tapes Display a list of the device paths for installed SCSI tape controllers.
show-ttys Display a list of the device paths for tty devices.
.traps Display a list of the SPARC trap types.
.version Display the version and date of the OpenBoot PROM.

Emergency Commands

These commands must be typed from the keyboard, they will not work from a console which is attached via the serial ports. With the exception of the Stop-A command, these commands are issued by pressing and holding down the indicated keys on the keyboard immediately after the system has been powered on. The keys must be held down until the monitor has checked their status. The Stop-A command can be issued at any time after the console display begins, and the keys do not need to be held down once they've been pressed. The Stop-D, Stop-F and Stop-N commands are not allowed when one of the security modes has been set. Not all commands are available on all workstations.

Stop (L1) Bypass the Power-On Self Test (POST). This is only effective if the system has been placed into the diagnostic mode.

Stop-A (L1-A) Abort the current operation and return to the monitor's default prompt.

Stop-D (L1-D) Set the system's **'diag-switch?'** NVRAM parameter to **'true'**, which places the system in diagnostic mode. POST diagnostics, if present, will be run, and the messages will be displayed via the system's serial port A.

Stop-F (L1-F) Enter the OpenBoot monitor before the monitor has probed the system for devices. Issue the **'fexit'** command to continue with system initialization.

Stop-N (L1-N) Causes the NVRAM parameters to be reset to their default values. Note that not all parameters have default values.

Line Editor Commands

The following commands can be used while the monitor is displaying the **ok** prompt. Not all of these editing commands are available on all workstations.

CTRL-A Place the cursor at the start of line.
CTRL-B Move the cursor backward one character.
ESC-B Move the cursor backward one word.
CTRL-D Erase the character that the cursor is currently highlighting.
ESC-D Erase the portion of word from the cursor's present position to the end of the word.
CTRL-E Place the cursor at the end of line.
CTRL-F Move the cursor forward one character.
ESC-F Move the cursor forward one word.
CTRL-H Erase the character preceding the cursor (also use Delete or Back Space)
ESC-H Erase the portion of the word which precedes the cursor (use also CTRL-W)

CTRL-K	Erase from the cursor's present position to the end of the line.
CTRL-L	Show the command history list.
CTRL-N	Recall the next command from the command history list
CTRL-P	Recall a previous command from the command history list.
CTRL-Q	Quote the next character (used to type a control character).
CTRL-R	Retype the current line.
CTRL-U	Erase from the cursor's present position to the beginning of the line.
CTRL-Y	Insert the contents of the memory buffer into the line, in front (to the left) of the cursor.

nvramrc

The **nvramrc** is an area of the system's NVRAM where users may store Forth programs. The programs which are stored in the **nvramrc** will executed each time the system is reset, provided that the **'use-nvramrc?'** NVRAM parameter has been set to **'true'**. Refer to the *OpenBoot 2.x Command Reference Manual* book for information on how to edit and use the **nvramrc**.

Restricted Monitor

The command **'old-mode'** is used to move OpenBoot into a restricted monitor mode, causing the **>** prompt to be displayed. Only three commands are allowed while in the restricted monitor; the **'go'** command (to resume a program which was interrupted with the Stop-A command), the **'n'** command (to return to the normal OpenBoot monitor), and boot commands. The restricted monitor's boot commands will approximate the older **SunMON** monitor's boot command syntax. If a **'security-mode'** has been turned on then the restricted monitor becomes the default monitor environment. The restricted monitor may also become the default environment if the **'sunmon-compat?'** NVRAM parameter is set to true. (Note that not all workstations will have the **'sunmon-compat?'** parameter.)

SunMON PROM USAGE

The following commands are available systems with older **SunMON**-based PROM:

+ | **–**
Increment or decrement the current address and display the contents of the new location.

^C *source destination n*
(caret-C) Copy, byte-by-byte, a block of length *n* from the *source* address to the *destination* address.

^I *program*
(caret-I) Display the compilation date and location of *program*.

^T *virtual_address*
(caret-T) Display the physical address to which *virtual_address* is mapped.

b [**!**] [*device* [**(***c***,***u***,***p***)**]] [*pathname*] [*arguments_list*]

b[**?**]
Reset appropriate parts of the system and bootstrap a program. A '**!**' (preceding the *device* argument) prevents the system reset from occurring. Programs can be loaded from various devices (such as a disk, tape, or Ethernet). '**b**' with no arguments will cause a default boot, either from a disk, or from an Ethernet controller. '**b?**' displays all boot devices and their *device*s.

device one of

le	Lance Ethernet
ie	Intel Ethernet
sd	SCSI disk, CDROM
st	SCSI 1/4" or 1/2" tape
fd	Diskette
id	IPI disk
mt	Tape Master 9-track 1/2" tape
xd	Xylogics 7053 disk
xt	Xylogics 1/2" tape
xy	Xylogics 440/450 disk

c A controller number (**0** if only one controller),

u A unit number (**0** if only one driver), and

p A partition.

pathname A pathname for a program such as **/stand/diag**.

arguments_list
A list of up to seven arguments to pass to the program being booted.

c [*virtual_address*] Resume execution of a program. When given, *virtual_address* is the address at which execution will resume. The default is the current PC. Registers are restored to the values shown by the **d**, and **r** commands.

d [*window_number*] Display (dump) the state of the processor. The processor state is observable only after:

- An unexpected trap was encountered.
- A user program dropped into the monitor (by calling *abortent*).
- The user manually entered the monitor by typing L1–A or BREAK.

The display consists of the following:

- The special registers: PSR, PC, nPC, TBR, WIM, and Y
- Eight global registers
- 24 window registers (8 *in*, 8 *local*, and 8 *out*), corresponding to one of the 7 available windows. If a Floating-Point Unit is on board, its status register along with 32 floating-point registers are also shown.

window_number Display the indicated *window_number*, which can be any value between **0** and **6**, inclusive. If no window is specified and the PSR's current window pointer contains a valid window number, registers from the window that was active just prior to entry into the monitor are displayed. Otherwise, registers from window 0 are displayed.

e [*virtual_address*] [*action*] . . .

Open the 16-bit word at *virtual_address* (default zero). The address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

f *virtual_address1 virtual_address2 pattern* [*size*]

Fill the bytes, words, or long words from *virtual_address1* (lower) to *virtual_address2* (higher) with the constant, *pattern*. The *size* argument can take one of the following values:

b	byte format (the default)
w	word format
l	long word format

For example, the following command fills the address block from **0x1000** to **0x2000** with the word pattern, **0xABCD**:

f 1000 2000 ABCD W

g [*vector*] [*argument*]
g [*virtual_address*] [*argument*]

Goto (jump to) a predetermined or default routine (first form), or to a user-specified routine (second form). The value of *argument* is passed to the routine. If the *vector* or *virtual_address* argument is omitted, the value in the PC is used as the address to jump to.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **g** command, set the variable ***romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

%x	hexadecimal
%d	decimal

g0 Force a panic and produce a crash dump when the monitor is running as a result of the system being interrupted,

g4 (Sun-4 systems only) Force a kernel stack trace when the monitor is running as a result of the system being interrupted,

h Display the help menu for monitor commands and their descriptions. To return to the monitor's basic command level, press ESCAPE or **q** before pressing RETURN.

i [*cache_data_offset*] [*action*] . . .
Modify cache data RAM command. Display and / or modify one or more of the cache data addresses. See the **a** command for a description of *action*.

j [*cache_tag_offset*] [*action*] . . .
Modify cache tag RAM command. Display and / or modify the contents of one or more of the cache tag addresses. See the **a** command for a description of *action*.

k [*reset_level*] Reset the system, where *reset_level* is:

0 Reset VMEbus, interrupt registers, video monitor (Sun-4 systems). This is the default.

1 Software reset.

2 Power-on reset. Resets and clears the memory. Runs the EPROM-based diagnostic self test, which can take several minutes, depending upon how much memory is being tested.

kb Display the system banner.

l [*virtual_address*] [*action*] . . .
Open the long word (32 bit) at memory address *virtual_address* (default zero). The address is interpreted in the address space defined by the **s** command (below). See the **a** command for a description of *action*.

m [*virtual_address*] [*action*] . . .
Open the segment map entry that maps *virtual_address* (default zero). The address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

ne

ni Disable, enable, or invalidate the cache, respectively.

o [*virtual_address*] [action*]* . . .
Open the byte location specified by *virtual_address* (default zero). The address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

p [*virtual_address*] [*action*]. . .
Open the page map entry that maps *virtual_address* (default zero) in the address space defined by the **s** command. See the **a** command for a description of *action*.

q [*eeprom_offset*] [*action*]. . .

Open the EEPROM *eeprom_offset* (default zero) in the EEPROM address space. All addresses are referenced from the beginning or base of the EEPROM in physical address space, and a limit check is performed to insure that no address beyond the EEPROM physical space is accessed. This command is used to display or modify configuration parameters, such as: the amount of memory to test during self test, whether to display a standard or custom banner, if a serial port (A or B) is to be the system console, etc. See the **a** command for a description of *action*.

r [*register_number*]
r [*register_type*]
r [*w window_number*]

Display and / or modify one or more of the IU or FPU registers. A hexadecimal *register_number* can be one of:

0x00–0x0f	window(0,i0)–window(0,i7), window(0,i0)—window(0,i7)
0x16–0x1f	window(1,i0)–window(1,i7), window(1,i0)—window(1,i7)
0x20–0x2f	window(2,i0)–window(2,i7), window(2,i0)—window(2,i7)
0x30–0x3f	window(3,i0)–window(3,i7), window(3,i0)—window(3,i7)
0x40–0x4f	window(4,i0)–window(4,i7), window(4,i0)—window(4,i7)
0x50–0x5f	window(5,i0)–window(5,i7), window(5,i0)—window(5,i7)
0x60–0x6f	window(6,i0)–window(6,i7), window(6,i0)—window(6,i7)
0x70–0x77	**g0, g1, g2, g3, g4, g5, g6, g7**
0x78–0x7d	PSR, PC, nPC, WIM, TBR, Y.
0x7e–0x9e	FSR, f0–f31

Register numbers can only be displayed after an unexpected trap, a user program has entered the monitor using the *abortent* function, or the user has entered the monitor by manually typing L1–A or BREAK.

If a *register_type* is given, the first register of the indicated type is displayed. *register_type* can be one of:

f	floating-point
g	global
s	special

If **w** and a *window_number* (**0**—**6**) are given, the first *in*-register within the indicated window is displayed. If *window_number* is omitted, the window that was active just prior to entering the monitor is used. If the PSR's current window pointer is invalid, window 0 is used.

s [*asi*])

Set or display the Address Space Identifier. With no argument, **s** displays the current Address Space Identifier. The *asi* value can be one of:

0x2	control space
0x3	segment table
0x4	Page table
0x8	user instruction
0x9	supervisor instruction
0xa	user data
0xb	supervisor data
0xc	flush segment
0xd	flush page
0xe	flush context
0xf	cache data

u [*echo*]

u [*port*] [*options*] [*baud_rate*]

u [**u**] [*virtual_address*]

With no arguments, display the current I/O device characteristics including: current input device, current output device, baud rates for serial ports A and B, an input-to-output echo indicator, and virtual addresses of mapped UART devices. With arguments, set or configure the current I/O device. With the **u** argument (**uu**...), set the I/O device to be the *virtual_address* of a UART device currently mapped.

echo Can be either **e** to enable input to be echoed to the output device, or **ne**, to indicate that input is not echoed.

port Assign the indicated *port* to be the current I/O device. *port* can be one of:

a	serial port A
b	serial port B
k	the workstation keyboard
s	the workstation screen

baud_rate Any legal baud rate.

options can be any combination of:

i	input
o	output
u	UART
e	echo input to output
ne	do not echo input

r reset indicated serial port (**a** and **b** ports only)

If either **a** or **b** is supplied, and no *options* are given, the serial port is assigned for both input and output. If **k** is supplied with no options, it is assigned for input only. If **s** is supplied with no options, it is assigned for output only.

v *virtual_address1 virtual_address2* [*size*]

Display the contents of *virtual_address1* (lower) *virtual_address2* (higher) in the format specified by *size*:

b byte format (the default)
w word format
l long word format

Enter return to pause for viewing; enter another return character to resume the display. To terminate the display at any time, press the space bar.

For example, the following command displays the contents of virtual address space from address **0x1000** to **0x2000** in word format:

```
v 1000 2000 W
```

w [*virtual_address*] [*argument*]

Set the execution vector to a predetermined or default routine. Pass *virtual_address* and *argument* to that routine.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **w** command, set the variable *****romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

%x hexadecimal
%d decimal

x Display a menu of extended tests. These diagnostics permit additional testing of such things as the I/O port connectors, video memory, workstation memory and keyboard, and boot device paths.

y c *context_number*
y p | s *context_number virtual_address*

Flush the indicated context, context page, or context segment.

c flush context *context_number*
p flush the page beginning at *virtual_address* within context *context_number*
s flush the segment beginning at *virtual_address* within

context *context_number*

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC

SEE ALSO

tip(1), **boot**(1M), **eeprom**(1M), **attributes**(5)

OpenBoot 2.x Command Reference Manual

NAME

mount, umount – mount or unmount file systems and remote resources

SYNOPSIS

mount [**–p** | **–v**]
mount [**–F** *FSType*] [*generic_options*] [**–o** *specific_options*] [**–O**]
special | *mount_point*
mount [**–F** *FSType*] [*generic_options*] [**–o** *specific_options*] [**–O**]
special mount_point
mount -a [**–F** *FSType*] [**–V**] [*current_options*] [**–o** *specific_options*]
[*mount_point. . .*]

umount [**–V**] [**–o** *specific_options*] *special* | *mount_point*

umount -a [**–V**] [**–o** *specific_options*] [*mount_point. . .*]

DESCRIPTION

mount attaches a file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the **mount** operation, these are hidden until the file system is unmounted.

umount unmounts a currently mounted file system, which may be specified either as a *mount_point* or as *special*, the device on which the file system resides.

mount and **umount** maintain a table of mounted file systems in **/etc/mnttab**, which is described in **mnttab**(4). **mount** adds an entry to the mount table; **umount** removes an entry from the table.

When invoked with both the *special* and *mount_point* arguments and the –**F** option, **mount** validates all arguments except for *special* and invokes the appropriate *FSType*-specific **mount** module. If invoked with no arguments, **mount** lists all the mounted file systems recorded in the mount table, **/etc/mnttab**. If invoked with a partial argument list (with only one of *special* or *mount_point*, or with both *special* or *mount_point* specified but not *FSType*), **mount** will search **/etc/vfstab** for an entry that will supply the missing arguments. If no entry is found, and the special argument starts with "/", the default local file system type specified in **/etc/default/fs** will be used. Otherwise the default remote file system type will be used. The default remote file system type is determined by the first entry in the **/etc/dfs/fstypes** file. After filling in missing arguments, **mount** will invoke the *FSType*-specific **mount** module.

Only a super-user can mount or unmount file systems using **mount** and **umount**. However, any user can use **mount** to list mounted file systems and resources.

OPTIONS

–**F** *FSType*
Used to specify the *FSType* on which to operate. The *FSType* must be specified or must be determinable from **/etc/vfstab**, or by consulting **/etc/default/fs** or **/etc/dfs/fstypes.**

-a [*mount_points. . .*]
Perform mount or umount operations in parallel, when possible.

If mount points are not specified, **mount** will mount all file systems whose **/etc/vfstab** "mount at boot" field is "yes". If mount points are specified, then **/etc/vfstab** "mount at boot" field will be ignored.

If mount points are specified, **umount** will only umount those mount points. If none is specified, then **umount** will attempt to umount all filesystems in **/etc/mnttab**, with the exception of certain system required file systems: **/**, **/usr**, **/var**, **/proc**, **/dev/fd**, and **/tmp**.

−p Print the list of mounted file systems in the **/etc/vfstab** format. Must be the only option specified.

−v Print the list of mounted file systems in verbose format. Must be the only option specified.

−V Echo the complete command line, but do not execute the command. **umount** generates a command line by using the options and arguments provided by the user and adding to them information derived from **/etc/mnttab**. This option should be used to verify and validate the command line.

generic_options Options that are commonly supported by most *FSType*-specific command modules. The following options are available:

- **−m** Mount the file system without making an entry in **/etc/mnttab**.
- **−r** Mount the file system read-only.

−o Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command. (See **mount_ufs**(1M))

−O Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error “device busy”.

USAGE

See **largefile**(5) for the description of the behavior of **mount** and **umount** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/mnttab mount table

/etc/default/fs default local file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: **LOCAL=ufs**

LOCAL: The default partition for a command if no *FSType* is specified.

/etc/vfstab list of default parameters for each file system.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mount_cachefs(1M), **mount_hsfs**(1M), **mount_nfs**(1M), **mount_pcfs**(1M), **mount_tmpfs**(1M), **mount_ufs**(1M), **mountall**(1M), **umountall**(1M), **setmnt**(1M), **mnttab**(4), **vfstab**(4), **attributes**(5), **largefile**(5), **lofs**(7FS), **pcfs**(7FS)

NOTES

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

NAME

mountall, umountall – mount, unmount multiple file systems

SYNOPSIS

mountall [–**F** *FSType*] [–**l** | –**r**] [*file_system_table*]

umountall [–**k**] [–**s**] [–**F** *FSType*] [–**l** | –**r**]

umountall [–**k**] [–**s**] [–**h** *host*]

DESCRIPTION

mountall is used to mount file systems specified in a file system table. The file system table must be in **vfstab**(4) format. If no *file_system_table* is specified, **/etc/vfstab** will be used. If '–' is specified as *file_system_table*, **mountall** will read the file system table from the standard input. **mountall** only mounts those file systems with the **mount at boot** field set to **yes** in the *file_system_table*.

Each file system which has an **fsckdev** entry specified in the file system table will be checked using **fsck**(1M) in order to determine if it may be safely mounted. If the file system does not appear mountable, it is fixed using **fsck** before the mount is attempted. File systems with a '–' entry in the **fsckdev** field will be mounted without first being checked.

umountall causes all mounted file systems except **root**, **/proc**, **/var**, and **/usr** to be unmounted. If the *FSType* is specified, **mountall** and **umountall** limit their actions to the *FSType* specified.

OPTIONS

–**F** Specify the *FSType* of the file system to be mounted or unmounted.

–**h** *host* Unmount all file systems listed in **/etc/mnttab** that are remote-mounted from host.

–**k** Use the "**fuser -k** <*mount point*>" command. (See the **fuser**(1M) man page for details.) The –**k** option sends the **SIGKILL** signal to each process using the file. Since this option spawns kills for each process, the kill messages may not show up immediately.

–**l** Limit the action to local file systems.

–**r** Limit the action to remote file system types.

–**s** Do not perform the **umount** operation in parallel.

FILES

/etc/mnttab	mounted file system table.
/etc/vfstab	table of file system defaults.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fsck(1M), **mount**(1M), **mnttab**(4), **vfstab**(4), **attributes**(5)

DIAGNOSTICS

No messages are printed if the file systems are mountable and clean.

Error and warning messages come from **fsck**(1M) and **mount**(1M).

NAME

mount_cachefs – mount CacheFS file systems

SYNOPSIS

mount –**F cachefs** [*generic_options*]
–**o backfstype=***file_system_type* [*other cacheFS options*] *special mount_point*

DESCRIPTION

The CacheFS-specific version of the **mount** command mounts a cached file system; if necessary, it NFS-mounts its back file system. It also provides a number of CacheFS-specific options for controlling the caching process. For more information regarding back file systems, refer to the *System Administration Guide*.

OPTIONS

To mount a CacheFS file system, use the generic **mount** command with the –**F** option followed by the argument **cachefs**. The following generic **mount** options are available:

–**m** Mount the file system without making an entry in the **/etc/mnttab** file.

–**O** Overlay mount. Allows the filesystem to be mounted over an existing mount point, making the underlying filesystem inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, mount will fail with the error:

mount –**F cachefs: mount failed Device busy**.

–**r** Mount the file system read-only.

The following arguments to the –**o** option are specifically for CacheFS mounts. Use commas to separate multiple options. Note: the **backfstype** argument must be specified.

backfstype=*file_system_type*
The file system type of the back file system (for example, **nfs**).

backpath=*path* Specifies where the back file system is already mounted. If this argument is not supplied, CacheFS determines a mount point for the back file system. The back file system must be read-only.

cachedir=*directory*
The name of the cache directory.

cacheid=*ID* *ID* is a string specifying a particular instance of a cache. If you do not specify a cache ID, CacheFS will construct one.

write-around | **non-shared**
Write modes for CacheFS. The **write-around** mode (the default) handles writes the same as NFS does; that is, writes are made to the back file system, and the affected file is purged from the cache. You can use the **non-shared** mode when you are sure that no one else will be writing to the cached file system. In this mode, all writes are made to both the front and the back file system, and the file remains in the cache.

noconst Disables cache consistency checking. By default, periodic consistency checking is enabled. Specify **noconst** only when you know that the back file system will not be modified. Trying to perform cache consistency check using **cfsadmin –s** will result in error. **demandconst** and **noconst** are mutually exclusive.

demandconst Verifies cache consistency only when explicitly requested, rather than the periodic checking that is done by default. A consistency check is requested by using the **–s** option of the **cfsadmin**(1M) command. This option is useful for back file systems that change infrequently, for example, **/usr/openwin**. **demandconst** and **noconst** are mutually exclusive.

local-access Causes the front file system to interpret the mode bits used for access checking instead or having the back file system verify access permissions. Do not use this argument with secure NFS.

purge Purge any cached information for the specified file system.

rw | **ro** Read-write (default) or read-only.

suid | **nosuid** Allow (default) or disallow set-uid execution.

acregmin=*n* Specifies that cached attributes are held for at least *n* seconds after file modification. After *n* seconds, CacheFS checks to see if the file modification time on the back file system has changed. If it has, all information about the file is purged from the cache and new data is retrieved from the back file system. The default value is **30** seconds.

acregmax=*n* Specifies that cached attributes are held for no more than *n* seconds after file modification. After *n* seconds, all file information is purged from the cache. The default value is **30** seconds.

acdirmin=*n* Specifies that cached attributes are held for at least *n* seconds after directory update. After *n* seconds, CacheFS checks to see if the directory modification time on the back file system has changed. If it has, all information about the directory is purged from the cache and new data is retrieved from the back file system. The default value is **30** seconds.

acdirmax=*n* Specifies that cached attributes are held for no more than *n* seconds after directory update. After *n* seconds, all directory information is purged from the cache. The default value is **30** seconds.

actimeo=*n* Sets **acregmin**, **acregmax**, **acdirmin**, and **acdirmax** to *n*.

EXAMPLES

The following example CacheFS-mounts the file system **server1:/user2**, which is already NFS-mounted on **/usr/abc** as **/xyz**.

```
example# mount -F cachefs -o backfstype=nfs,backpath=/usr/abc,
        cachedir=/cache1 server1:/user2 /xyz
```

The lines similar to the following appear in the **/etc/mnttab** file after the **mount** command is executed:

```
server1:/user2    /usr/abc       nfs
/usr/abc          /cache1/xyz    cachefs    backfstype=nfs
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cfsadmin(1M), **fsck_cachefs**(1M), **mount**(1M), **attributes**(5)
System Administration Guide

NAME

mountd – server for NFS mount requests and NFS access checks

SYNOPSIS

/usr/lib/nfs/mountd [**–v**] [**–r**]

DESCRIPTION

mountd is an RPC server that answers requests for NFS access information and file system mount requests. It reads the file **/etc/dfs/sharetab** to determine which file systems are available for mounting by which remote machines. See **sharetab**(4). **nfsd** running on the local server will contact **mountd** the first time an NFS client tries to access the file system to determine whether the client should get read-write, read-only, or no access. This access can be dependent on the security mode used in the remoted procedure call from the client. See **share_nfs**(1M).

The command also provides information as to what file systems are mounted by which clients. This information can be printed using the **showmount**(1M) command.

The **mountd** daemon is automatically invoked in run level 3.

Only super user can run the **mountd** daemon.

OPTIONS

–v Run the command in verbose mode. Each time **mountd** determines what access a client should get, it will log the result to the console, as well as how it got that result.

–r Reject mount requests from clients. Clients that have file systems mounted will not be affected.

FILES

/etc/dfs/sharetab shared file system table

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

nfsd(1M), **share_nfs**(1M), **showmount**(1M), **sharetab**(4), **attributes**(5)

NOTES

If **nfsd** is running, **mountd** must also be running in order to be assured that the NFS server can respond to requests, otherwise, the NFS service can hang.

Some routines that compare hostnames use case-sensitive string comparisons; some do not. If an incoming request fails, verify that the case of the hostname in the file to be parsed matches the case of the hostname called for, and attempt the request again.

NAME

mount_hsfs – mount hsfs file systems

SYNOPSIS

mount –**F hsfs** [*generic_options*] [−**o** *FSType-specific_options*] [−**O**]
special | *mount_point*
mount –**F hsfs** [*generic_options*] [−**o** *FSType-specific_options*] [−**O**]
special mount_point

DESCRIPTION

mount attaches a High Sierra file system (**hsfs**) to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the **mount** operation, these are hidden until the file system is unmounted.

If **mount** is invoked with *special* or *mount_point* as the only arguments, **mount** will search **/etc/vfstab** to fill in the missing arguments, including the *FSType-specific_options*; see **mount**(1M) for more details.

If the file system being mounted contains Rock Ridge extensions, by default they will be used, enabling support of features not normally available under High Sierra file systems such as symbolic links, and special files.

OPTIONS

generic_options
See **mount**(1M) for the list of supported options.

−**o** Specify **hsfs** file system specific options. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

ro Mount the file system read-only. This option is required.

nrr ‘‘no Rock Ridge’’: if Rock Ridge extensions are present in the file system, ignore them; interpret it as a regular High Sierra file system.

notraildot File names on High Sierra file systems consist of a proper name and an extension separated by a ’.’ (dot) character. By default, the separating dot is always considered part of the file’s name for all file access operations, even if there is no extension present. Specifying **notraildot** makes it optional to specify the trailing dot to access a file whose name lacks an extension.

Exceptions:
This option is effective only on file systems for which Rock Ridge extensions are not active, either because they are not present on the CD-ROM, or they are explicitly ignored via the **nrr** option. If Rock Ridge extensions are active, **hsfs** quietly ignores this option.

nomaplcase
File names on High Sierra cdroms with no Rock Ridge extensions present should be uppercase characters only. By default, **hsfs** maps file names read from a non-Rock Ridge disk to all lowercase characters. **nomaplcase** turns off this mapping. The exceptions for

notraildot discused above apply to **nomaplcase**.

nosuid By default the file system is mounted with setuid execution allowed. Specifying **nosuid** causes the file system to be mounted with setuid execution disallowed.

–O Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error “device busy”.

FILES

/etc/mnttab	table of mounted file systems
/etc/vfstab	list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mount(1M), **mountall**(1M), **mount**(2), **mnttab**(4), **vfstab**(4), **attributes**(5)

NOTES

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

NAME

mount_nfs – mount remote NFS resources

SYNOPSIS

mount [–**F nfs**] [–**r**] [–**m**] [–**o** *specific_options*] [–**O**] *resource*
mount [–**F nfs**] [–**r**] [–**m**] [–**o** *specific_options*] [–**O**] *mount_point*
mount [–**F nfs**] [–**r**] [–**m**] [–**o** *specific_options*] [–**O**] *resource mount_point*

DESCRIPTION

The **mount** utility attaches a named *resource* to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the **mount** operation, the contents remain hidden until the *resource* is once again unmounted.

If the resource is listed in the **/etc/vfstab** file, the command line can specify either *resource* or *mount_point*, and **mount** will consult **/etc/vfstab** for more information. If the –**F** option is omitted, **mount** takes the file system type from **/etc/vfstab**.

If the resource is not listed in the **/etc/vfstab** file, then the command line must specify both the *resource* and the *mount_point*.

mount maintains a table of mounted file systems in **/etc/mnttab**, described in **mnttab**(4).

OPTIONS

–**r** Mount the specified file system read-only.

–**m** Do not append an entry to the **/etc/mnttab** table of mounted file systems

–**o** *specific_options*
Set file system specific options according to a comma-separated list chosen from words below.

rw | **ro** *resource* is mounted read-write or read-only. The default is **rw**.

suid | **nosuid** Setuid execution allowed or disallowed. The default is **suid**.

remount If a file system is mounted read-only, remounts the file system read-write.

bg | **fg** If the first attempt fails, retry in the background, or, in the foreground. The default is **fg**.

quota Enables **quota**(1M) to check whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.

noquota Prevent **quota**(1M) from checking whether the user exceeded the quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.

retry=*n* The number of times to retry the mount operation. The default is **10000**.

vers=<*NFS version number*>
By default, the version of NFS protocol used between the client and the server is the highest one available on both systems. If the NFS server does not support NFS Version 3 protocol, then the NFS mount will use NFS Version 2 protocol.

proto=<*netid*>
<*netid*> is a value of network_id field from entry in the **/etc/netconfig** file. By default, the transport protocol used for the NFS mount will be first available connection oriented transport supported on both the client and the server. If no connection oriented transport is found, then the first available connectionless transport is used. This default behavior can be overridden with the **proto=**<*netid*> option.

port=*n*
The server IP port number. The default is **NFS_PORT**.

grpid
By default, the GID associated with a newly created file will obey the System V semantics; that is, the GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis by setting the set-GID bit of the parent directory; in this case, the GID of a newly created file is set to the GID of the parent directory (see **open**(2) and **mkdir**(2)). Files created on file systems that are mounted with the **grpid** option will obey BSD semantics independent of whether the set-GID bit of the parent directory is set; that is, the GID is unconditionally inherited from that of the parent directory.

rsize=*n*
Set the read buffer size to *n* bytes. The default value is **32768** when using Version 3 of the NFS protocol. The default can be negotiated down if the server prefers a smaller transfer size. When using Version 2, the default value is **8192**.

wsize=*n*
Set the write buffer size to *n* bytes. The default value is **32768** when using Version 3 of the NFS protocol. The default can be negotiated down if the server prefers a smaller transfer size. When using Version 2, the default value is **8192**.

timeo=*n*
Set the NFS timeout to *n* tenths of a second. The default value is **11** tenths of a second for connectionless transports, and **600** tenths of a second for connection-oriented transports.

retrans=*n*
Set the number of NFS retransmissions to *n*. The default value is **5**. For connection-oriented transports, this option has no effect because it is assumed that the transport will perform retransmissions on behalf of NFS.

soft | **hard**
Return an error if the server does not respond, or continue the retry request until the server responds. The default value is **hard**.

intr | **nointr**
Allow (do not allow) keyboard interrupts to kill a process that is

hung while waiting for a response on a hard-mounted file system. The default is **intr**, which makes it possible for clients to interrupt applications that may be waiting for a remote mount.

sec=*mode* Set the security *mode* for NFS transactions. If **sec=** is not specified, then the default action is to use **AUTH_SYS** over NFS Version 2 mounts, or to negotiate a *mode* over NFS Version 3 mounts. NFS Version 3 mounts negotiate a security mode when the server returns an array of security modes. The client will pick the first mode in the array that is supported on the client. Only one mode can be specified with the **sec=** option. See **nfssec**(5) for the available *mode* options.

secure This option has been deprecated in favor of the **sec=***dh* option.

posix Request POSIX.1 semantics for the file system. Requires a mount Version 2 **mountd**(1M) on the server. See **standards**(5) for information regarding POSIX.

kerberos This options has been deprecated in favor of the **sec=***krb4* option.

noac Suppress data and attribute caching.

acdirmax=*n* Hold cached attributes for no more than *n* seconds after directory update. The default value is **60**.

acdirmin=*n* Hold cached attributes for at least *n* seconds after directory update. The default value is **30**.

acregmax=*n* Hold cached attributes for no more than *n* seconds after file modification. The default value is **60**.

acregmin=*n* Hold cached attributes for at least *n* seconds after file modification. The default value is **3**.

actimeo=*n* Set *min* and *max* times for regular files and directories to *n* seconds.

−O Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error “device busy.”

NFS FILE SYSTEMS

Background versus Foreground

File systems mounted with the **bg** option indicate that **mount** is to retry in the background if the server’s mount daemon (**mountd**(1M)) does not respond. **mount** retries the request up to the count specified in the **retry=***n* option. Once the file system is mounted, each NFS request made in the kernel waits **timeo=***n* tenths of a second for a response. If no response arrives, the time-out is multiplied by **2** and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=***n* option, a file system mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Hard versus Soft

File systems that are mounted read-write or that contain executable files should always be mounted with the **hard** option. Applications using **soft** mounted file systems may incur unexpected I/O errors.

Authenticated Requests

The server may require authenticated NFS requests from the client. Either **sec=***dh* or **sec=***krb4* authentication may be required. See **nfssec**(5).

Replicated file systems and failover

resource can list multiple read–only file systems to be used to provide data. These file systems should contain equivalent directory structures and identical files. It is also recommended that they be created by a utility such as **rdist**(1). The file systems may be specified either with a comma–separated list of *host:/pathname* entries, or with a comma –separated list of hosts, if all file system names are the same. If multiple file systems are named and the first server in the list is down, failover will use the next alternate server to access files. If the read–only option is not chosen, replication will be disabled. File access will block on the original if NFS locks are active for that file.

File Attributes

To improve NFS read performance, files and file attributes are cached. File modification times get updated whenever a write occurs. However, file access times may be temporarily out-of-date until the cache gets refreshed.

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=***n* sets flush time to *n* seconds for both regular files and directories.

Setting **actimeo=***n* disables attribute caching on the client. This means that every reference to attributes will be satisfied directly from the server though file data will still be cached. While this guarantees that the client always has the latest file attributes from the server, it has an adverse effect on performance through additional latency, network load, and server load.

Setting the **noac** option also disables attribute caching, but has the further effect of disabling client write caching. While this guarantees that data written by an application will be written directly to a server, where it can be viewed immediately by other clients, it has a significant adverse effect on client write performance. Data written into memory-mapped file pages (**mmap**(2)) will not be written directly to this server.

EXAMPLES

To mount an NFS file system:

```
example# mount serv:/usr/src /usr/src
```

To mount an NFS file system read-only with no suid privileges:

```
example# mount –r –o nosuid serv:/usr/src /usr/src
```

To mount an NFS file system over Version 2, with the UDP transport:

```
example# mount -o vers=2,proto=udp serv:/usr/src /usr/src
```

To mount an NFS file system using **AUTH_KERB** authentication:

```
example# mount –o sec=krb4 serv:/usr/src /usr/src
```

To mount a replicated set of NFS file systems with the same pathnames:

```
example# mount serv–a,serv–b,serv–c:/usr/man /usr/man
```

To mount a replicated set of NFS file systems with different pathnames:

```
example# mount serv–x:/usr/man,serv–y:/var/man /usr/man
```

FILES

/etc/mnttab	table of mounted file systems
/etc/dfs/fstypes	default distributed file system type
/etc/vfstab	table of automatically mounted resources

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

rdist(1), **mountall**(1M), **mountd**(1M), **quota**(1M), **mkdir**(2), **mmap**(2), **mount**(2), **open**(2), **umount**(2), **mnttab**(4), **attributes**(5), **nfssec**(5), **standards**(5), **lofs**(7FS)

NOTES

An NFS server should not attempt to mount its own file systems. See **lofs**(7FS).

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on *the directory to which the symbolic link refers*, rather than being mounted on top of the symbolic link itself.

SunOS 4.X used the **biod** maintenance procedure to perform parallel read-ahead and write-behind on NFS clients. SunOS 5.X made **biod** obsolete with multi-threaded processing, which transparently performs parallel read-ahead and write-behind.

NAME

mount_pcfs – mount pcfs file systems

SYNOPSIS

mount –**F pcfs** [*generic_options*] [–**o** *FSType-specific_options*] *special* | *mount_point*
mount –**F pcfs** [*generic_options*] [–**o** *FSType-specific_options*] *special mount_point*

DESCRIPTION

mount attaches an MS-DOS file system (**pcfs**) to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the **mount** operation, these are hidden until the file system is unmounted.

If **mount** is invoked with *special* or *mount_point* as the only arguments, **mount** will search **/etc/vfstab** to fill in the missing arguments, including the *FSType-specific_options*; see **mount**(1M) for more details.

The *special* argument can be one of two special device file types:

- A floppy disk, such as **/dev/diskette0** or **/dev/diskette1**.
- A DOS logical drive on a hard disk expressed as *device-name*:*logical-drive*, where *device-name* specifies the special block device-file for the whole disk and *logical-drive* is either a drive letter (c through z) or a drive number (1 through 24). Examples are **/dev/dsk/c0t0d0p0:***c* and **/dev/dsk/c0t0d0p0:***1*.

The *special* device file type must have a formatted MS-DOS file system with either a 12-bit or a 16-bit File Allocation Table. Regular or BIG-DOS (greater than 32 megabytes in size) partitions can be mounted.

OPTIONS

generic_options See **mount**(1M) for the list of supported options.

–**o** Specify **pcfs** file system specific options. The following options are available:

rw | **ro** Mount the file system read / write or read-only. The default is **rw**.

foldcase | **nofoldcase**
Force uppercase characters in filenames to lowercase when reading them from the filesystem. This is for compatibility with the previous behavior of **pcfs**. The default is **nofoldcase**.

FILES

/etc/mnttab table of mounted file systems
/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

SEE ALSO

mount(1M), **mountall**(1M), **mount**(2), **mnttab**(4), **vfstab**(4), **attributes**(5), **pcfs**(7FS)

NOTES

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

NAME

mount_s5fs – mount s5 file systems

SYNOPSIS

mount –**F s5fs** [*generic_options*] [–**o** *FSType-specific_options*] *special* | *mount_point*
mount –**F s5fs** [*generic_options*] [–**o** *FSType-specific_options*] *special mount_point*

DESCRIPTION

mount attaches a s5 file system (a System V file system used by PC versions of UNIX) to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the **mount** operation, these are hidden until the file system is unmounted.

If **mount** is invoked with *special* or *mount_point* as the only arguments, **mount** will search **/etc/vfstab** to fill in the missing arguments, including the *FSType-specific_options*. See **mount**(1M).

If *special* and *mount_point* are specified without any *FSType-specific_options*, the default is **rw**.

OPTIONS

See **mount**(1M) for the list of supported *generic_options*.

–**o** Specify **s5** file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

f Fake an **/etc/mnttab** entry, but do not actually mount any file systems. Parameters are not verified.

n Mount the file system without making an entry in **/etc/mnttab**.

rw | **ro** Read-write or read-only. Default is **rw**.

remount Used in conjunction with **rw**. A file system mounted read-only can be *remounted* read-write. Fails if the file system is not currently mounted or if the file system is mounted **rw**.

FILES

/etc/mnttab	table of mounted file systems
/etc/vfstab	list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Architecture	x86
Availability	SUNWs53

SEE ALSO

mount(1M), **mountall**(1M), **mount**(2), **mnttab**(4), **vfstab**(4), **attributes**(5)

NOTES

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

NAME

mount_tmpfs – mount tmpfs file systems

SYNOPSIS

mount [**–F tmpfs**] [**–o size=***sz*] [**–O**] *special mount_point*

DESCRIPTION

tmpfs is a memory based file system which uses kernel resources relating to the VM system and page cache as a file system.

mount attaches a **tmpfs** file system to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the **mount** operation, these remain hidden until the file system is once again unmounted. The attributes (mode, owner, and group) of the root of the **tmpfs** filesystem are inherited from the underlying *mount_point*, provided that those attributes are determinable. If not, the root's attributes are set to their default values.

The *special* argument is usually specified as **swap** but is in fact disregarded and assumed to be the virtual memory resources within the system.

OPTIONS

-o size=*sz* The *sz* argument controls the size of this particular **tmpfs** file system. If the argument is has a 'k' suffix, the number will be interpreted as a number of kilobytes. An 'm' suffix will be interpreted as a number of megabytes. No suffix is interpreted as bytes. In all cases, the actual size of the file system is the number of bytes specified, rounded up to the physical pagesize of the system.

–O Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error "device busy".

FILES

/etc/mnttab table of mounted file systems

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mount(1M), **mkdir**(2), **mount**(2), **open**(2), **umount**(2), **mnttab**(4), **attributes**(5), **tmpfs**(7FS)

NOTES

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

NAME

mount_ufs – mount ufs file systems

SYNOPSIS

mount –**F ufs** [*generic_options*] [–**o** *FSType-specific_options*] [–**O**]
special | *mount_point*
mount –**F ufs** [*generic_options*] [–**o** *FSType-specific_options*] [–**O**]
special mount_point

DESCRIPTION

The **mount** utility attaches a ufs file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the **mount** operation, these are hidden until the file system is unmounted.

If **mount** is invoked with *special* or *mount_point* as the only arguments, **mount** will search **/etc/vfstab** to fill in the missing arguments, including the *FSType-specific_options*. See **mount**(1M).

If *special* and *mount_point* are specified without any *FSType-specific_options*, the default is **rw**.

OPTIONS

See **mount**(1M) for the list of supported *generic_options*.

–**o** Specify **ufs** file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

onerror=*action* where *action* = **panic** | **lock** | **umount**. This option specifies the action that UFS should take to recover from an internal inconsistency on a file system. These values cause a forced system shutdown, a file system lock to be applied to the file system, or the file system to be forcibly unmounted, respectively. The default is **panic**.

toosoon=*number*[**s** | **m** | **h** | **d** | **w** | **y**]
This option specifies the minimum time that must elapse between detection of inconsistencies on a file system. If an inconsistency is detected within this time period the system is forced to shut down. This prevents pathologic repairing of a file system which is damaged repeatedly. The optional unit key letter sets the units to be **s**econds, **m**inutes, **h**ours, **d**ays, **w**eeks, or **y**ears, respectively. The default value is **1w** (1 week).

noforcedirectio | **forcedirectio**
If **forcedirectio** is specified and supported by the file system, then for the duration of the mount forced direct I/O will be used. If the filesystem is mounted using **forcedirectio**, then data is transferred directly between user address space and the disk. If the filesystem is mounted using **noforcedirectio**, then data is buffered in kernel address space when data is transferred between

user address space and the disk. **forceddirectio** is a performance option that benefits only from large sequencial data transfers. The default behavior is **noforcedirectio**.

nolargefiles | largefiles
If **nolargefiles** is specified and supported by the file system, then for the duration of the mount it is guaranteed that all regular files in the file system have a size that will fit in the smallest object of type **off_t** supported by the system performing the mount. The mount will fail if there are any files in the file system not meeting this criterion. If **largefiles** is specified, there is no such guarantee. The default behavior is **largefiles**.

If **nolargefiles** is specified, **mount** will fail for **ufs** if the file system to be mounted has contained a large file (a file whose size is greater than or equal to 2 Gbyte) since the last invocation of **fsck** on the file system. The large file need not be present in the file system at the time of the mount for the mount to fail; it could have been created previously and destroyed. Invoking **fsck** (see **fsck_ufs**(1M)) on the file system will reset the file system state if no large files are present. After invoking **fsck**, a successful mount of the file system with **nolargefiles** specified indicates the absence of large files in the file system; an unsuccessful mount attempt indicates the presence of at least one large file.

f Fake an **/etc/mnttab** entry, but do not actually mount any file systems. Parameters are not verified.

m Mount the file system without making an entry in **/etc/mnttab**.

quota Quotas are turned on for the file system.

rw | **ro** Read-write or read-only. Default is **rw**.

rq Read-write with quotas turned on. Equivalent to **rw, quota**.

nosuid By default the file system is mounted with Setuid execution allowed. Specifying **nosuid** causes the file system to be mounted with setuid execution disallowed. **nosuid** can also be used to disallow setuid when mounting devices.

remount Used in conjunction with **rw**. A file system mounted read-only can be *remounted* read-write. Fails if the file system is not currently mounted or if the file system is mounted **rw**.

intr | **nointr** Allow(do not allow) keyboard interrupts to kill a process that is waiting for an operation on a locked file system. The default is **intr**.

–O Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error "device busy".

FILES

/etc/mnttab table of mounted file systems
/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fsck_ufs(1M), **mount**(1M), **mountall**(1M), **mount**(2), **mnttab**(4), **vfstab**(4), **attributes**(5), **largefile**(5)

NOTES

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

NAME

mpstat – report per-processor statistics

SYNOPSIS

/usr/bin/mpstat [*interval* [*count*]]

DESCRIPTION

mpstat reports per-processor statistics in tabular form. Each row of the table represents the activity of one processor. The first table summarizes all activity since boot; each subsequent table summarizes activity for the preceding interval. All values are rates (events per second) unless otherwise noted.

mpstat reports the following information:

CPU	processor ID
minf	minor faults
mjf	major faults
xcal	inter-processor cross-calls
intr	interrupts
ithr	interrupts as threads (not counting clock interrupt)
csw	context switches
icsw	involuntary context switches
migr	thread migrations (to another processor)
smtx	spins on mutexes (lock not acquired on first try)
srw	spins on readers / writer locks (lock not acquired on first try)
syscl	system calls
usr	percent user time
sys	percent system time
wt	percent wait time
idl	percent idle time

OPTIONS

interval Report once each *interval* seconds.

count Only print *count* reports.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

sar(1), **iostat**(1M), **sar**(1M), **vmstat**(1M), **attributes**(5)

NAME

mvdir – move a directory

SYNOPSIS

/usr/sbin/mvdir *dirname name*

DESCRIPTION

mvdir moves directories within a file system. *dirname* must be a directory. If *name* does not exist, it will be created as a directory. If *name* does exist, and is a directory, *dirname* will be created as *name/dirname*. *dirname* and *name* may not be on the same path; that is, one may not be subordinate to the other. For example:

```
example% mvdir x/y x/z
```

is legal, but

```
example% mvdir x/y x/y/z
```

is not.

OPERANDS

dirname The name of the directory that is to be moved to another directory in the filesystem.

name The name of the directory into which *dirname* is to be moved. If *name* does not exist, it will be created. It may not be on the same path as *dirname*.

USAGE

See **largefile**(5) for the description of the behavior of **mvdir** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

0 Successful operation.

>0 Operation failed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mkdir(1), **mv**(1), **attributes**(5), **largefile**(5)

NAME

named-xfer – ancillary agent for inbound zone transfers

SYNOPSIS

named-xfer -z *zone_to_transfer* **-f** *db_file* **-s** *serial_no* **[-d** *debuglevel***] [-l** *debug_log_file***]**
[-t *trace_file***] [-p** *port#* **] [-S]** *nameserver***...**

DESCRIPTION

The **named-xfer** program is an ancillary program executed by **in.named** to perform an inbound zone transfer. It is rarely executed directly, and only by system administrators who are trying to debug a zone transfer problem. See RFC's 1033, 1034, and 1035 for more information on the Internet name-domain system.

OPTIONS

–z Specifies the name of the zone to be transferred.

–f Specifies the name of the file into which the zone should be dumped when it is received from the primary server.

–s Specifies the serial number of the current copy of this zone. If the SOA RR from the primary server does not have a serial number higher than this, the transfer will be aborted.

–d Print debugging information. A number after the ''d'' determines the level of messages printed.

–l Specifies a log file for debugging messages. The default is system- dependent but is usually in */var/tmp* or */usr/tmp*. Note that this only applies if *–d* is also specified.

–t Specifies a trace file which will contain a protocol trace of the zone transfer. This is probably only of interest to those debugging the name server itself.

–p Use a different port number. The default is the standard port number as returned by **getservbyname**(3N) for service ''domain''.

–S Perform a restricted transfer of only the SOA, NS records and glue A records for the zone. The SOA record will not be loaded by named but will be used to determine when to verify the NS records. See the ''stubs'' directive in **in.named**(1M) for more information.

Additional arguments are taken as name server addresses in so-called ''dotted-quad'' syntax only; no host names are allowed. At least one address must be specified. If the first one fails to transfer successfully, the additional addresses will be tried in the order given.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

in.named(1M), **resolver**(3N), **resolv.conf**(4), **hostname**(1), **RFC 882**, **RFC 883**, **RFC 973**, **RFC 974**, **RFC 1033**, **RFC 1034**, **RFC 1035**, **RFC 1123**, *Name Server Operations Guide for BIND*.

NAME

ncheck – generate a list of path names versus i-numbers

SYNOPSIS

ncheck [**–F** *FSType*] [**–V**] [*generic_options*] [**–o** *FSType-specific_options*] [*special* . . .]

DESCRIPTION

ncheck with no options generates a path-name versus i-number list of all files on *special*. If *special* is not specified on the command line the list is generated for all *special*s in **/etc/vfstab** which have a numeric **fsckpass**. *special* is a block special device on which the file system exists.

OPTIONS

–F Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from **/etc/vfstab** by finding an entry in the table that has a numeric **fsckpass** field and an **fsckdev** that matches *special*.

–V Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from **/etc/vfstab**. This option may be used to verify and validate the command line.

generic_options Options that are commonly supported by most FSType-specific command modules. The following options are available:

- **–i** *i-list* Limit the report to the files on the *i-list* that follows. The *i-list* must be separated by commas with no intervening spaces.
- **–a** Print the names “.” and “. .” which are ordinarily suppressed.
- **–s** Report only special files and files with set-user-ID mode. This option may be used to detect violations of security policy.

–o Specify *FSType-specific_options* in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the FSType-specific module of the command.

USAGE

See **largefile**(5) for the description of the behavior of **ncheck** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/vfstab list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

vfstab(4), **attributes**(5), **largefile**(5)
Manual pages for the FSType-specific modules of **ncheck**

NOTES

This command may not be supported for all *FSTypes*.

NAME

ncheck_ufs – generate pathnames versus i-numbers for ufs file systems

SYNOPSIS

ncheck **–F ufs** [*generic_options*] [**–o m**] [*special* ...]

DESCRIPTION

ncheck **–F ufs** generates a pathname versus i-number list of files for the **ufs** file system residing on *special*. Names of directory files are followed by **/.** .

OPTIONS

See **ncheck**(1M) for the list of *generic_options* supported.

–o Specify **ufs** file system specific options. The available option is:

m Print mode information.

DIAGNOSTICS

When the file system structure is improper, '**??**' denotes the "parent" of a parentless file and a pathname beginning with '**. . .**' denotes a loop.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ff(1M), **ncheck**(1M), **attributes**(5)

NAME

ndd – get and set driver configuration parameters

SYNOPSIS

ndd [–**set**] *driver parameter* [*value*]

DESCRIPTION

ndd gets and sets selected configuration parameters in some kernel drivers. Currently, **ndd** only supports the drivers that implement the TCP / IP Internet protocol family. Each driver chooses which parameters to make visible using **ndd**. Since these parameters are usually tightly coupled to the implementation, they are likely to change from release to release. Some parameters may be read-only.

If the –**set** option is omitted, **ndd** queries the named *driver*, retrieves the value associated with the specified *parameter*, and prints it. If the –**set** option is given, **ndd** passes *value*, which must be specified, down to the named *driver* which assigns it to the named *parameter*.

By convention, drivers that support **ndd** also support a special read-only *parameter* named ‘‘**?**’’ which can be used to list the parameters supported by the driver.

EXAMPLES

To see which parameters are supported by the TCP driver, use the following command:

```
example% ndd /dev/tcp \?
```

NOTE: The parameter name ‘‘**?**’’ may need to be escaped with a backslash to prevent its being interpreted as a shell meta character.

The following command sets the value of the parameter **ip_forwarding** in the IP driver to zero. This disables IP packet forwarding.

```
example% ndd –set /dev/ip ip_forwarding 0
```

To view the current IP forwarding table, use the following command:

```
example% ndd /dev/ip ip_ire_status
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ioctl(2), **attributes**(5), **arp**(7P), **ip**(7P), **tcp**(7P), **udp**(7P)

NOTES

The parameters supported by each driver may change from release to release. Like programs that read **/dev/kmem**, user programs or shell scripts that execute **ndd** should be prepared for parameter names to change.

The **ioctl( )** command that **ndd** uses to communicate with drivers is likely to change in a future release. User programs should avoid making dependencies on it.

The meanings of many **ndd** parameters make sense only if you understand how the driver is implemented.

NAME

netstat – show network status

SYNOPSIS

netstat [**–anv**]

netstat [**–g** | **–m** | **–p** | **–s** | **–f** *address_family*] [**–n**] [**–P** *protocol*]

netstat {[**–i**] [**–I** *interface*]} [*interval*]

netstat –r [**–anv**]

netstat –M [**–ns**]

netstat –D [**–I** *interface*]

DESCRIPTION

netstat displays the contents of various network-related data structures in various formats, depending on the options you select.

The first form of the command displays a list of active sockets for each protocol. The second form selects one from among various other network data structures. The third form shows the state of the interfaces. The fourth form displays the routing table, the fifth form displays the multicast routing table, and the sixth form displays the state of DHCP on one or all interfaces.

OPTIONS

–a Show the state of all sockets and all routing table entries; normally, sockets used by server processes are not shown and only interface, host, network, and default routes are shown.

–f *address_family* Limit statistics or address control block reports to those of the specified address_family, which can be one of:

inet	For the **AF_INET** address family
unix	For the **AF_UNIX** address family

–g Show the multicast group memberships for all interfaces.

–i Show the state of the interfaces that are used for TCP/IP traffic. (See **ifconfig**(1M)).

–m Show the STREAMS statistics.

–n Show network addresses as numbers. **netstat** normally displays addresses as symbols. This option may be used with any of the display formats.

–p Show the address resolution (ARP) tables.

–r Show the routing tables.

–s Show per-protocol statistics. When used with the **–M** option, show multicast routing statistics instead.

–v Verbose. Show additional information for the sockets and the routing table.

–I *interface* Show the state of a particular interface. *interface* can be any valid interface such as **ie0** or **le0**.

–M Show the multicast routing tables. When used with the **–s** option, show multicast routing statistics instead.

–P *protocol* Limit display of statistics or state of all sockets to those applicable to *protocol*.

–d Show the state of all interfaces that are under Dynamic Host Configuration Protocol (DHCP) control.

–D Show the status of DHCP configured interfaces.

OPERANDS

interval If *interval* is specified, **netstat** displays interface information over the last *interval* seconds, repeating forever.

DISPLAYS

Active Sockets (First Form)

The display for each active socket shows the local and remote address, the send and receive queue sizes (in bytes), the send and receive windows (in bytes), and the internal state of the protocol.

The symbolic format normally used to display socket addresses is either

hostname.port

when the name of the host is specified, or

network.port

if a socket address specifies a network but no specific host.

The numeric host address or network number associated with the socket is used to look up the corresponding symbolic hostname or network name in the *hosts* or *networks* database.

If the network or hostname for an address is not known (or if the **–n** option is specified), the numerical network address is shown. Unspecified, or "wildcard", addresses and ports appear as "*". For more information regarding the Internet naming conventions, refer to **inet**(7P)**.**

TCP Sockets

The possible state values for TCP sockets are as follows:

CLOSED	Closed. The socket is not being used.
LISTEN	Listening for incoming connections.
SYN_SENT	Actively trying to establish connection.
SYN_RECEIVED	Initial synchronization of the connection under way.
ESTABLISHED	Connection has been established.
CLOSE_WAIT	Remote shutdown; waiting for the socket to close.
FIN_WAIT_1	Socket closed; shutting down connection.
CLOSING	Closed, then remote shutdown; awaiting acknowledgment.

LAST_ACK	Remote shutdown, then closed; awaiting acknowledgment.
FIN_WAIT_2	Socket closed; waiting for shutdown from remote.
TIME_WAIT	Wait after close for remote shutdown retransmission.

Network Data Structures (Second Form)

The form of the display depends upon which of the −**g**, −**m**, −**p**, or −**s** options you select. If you specify more than one of these options, **netstat** displays the information for each one of them.

Interface Status (Third Form)

The interface status display lists information for all current interfaces, one interface per line. If an interface is specified using the −**I** option, it displays information for only the specified interface.

The list consists of the interface name, **mtu** (maximum transmission unit, or maximum packet size)(see **ifconfig**(1M)), the network to which the interface is attached, addresses for each interface, and counter associated with the interface. The counters show the number of input packets, input errors, output packets, output errors, and collisions, respectively. For Point-to-Point interfaces, the Net/Dest field is the name or address on the other side of the link.

If the −**n** option is specified, the list displays the IP address instead of the interface name.

If an optional *interval* is specified, the output will be continuously displayed in *interval* seconds until interrupted by the user.

The input interface is specified using the −**I** option. In this case, the list only displays traffic information in columns; the specified interface is first, the total count is second. This column list has the format of:

```
    input    le0          output             input     (Total)     output
 packets  errs  packets   errs  colls   packets  errs   packets  errs  colls
 227681   0     659471    1     502     261331   0      99597    1     502
 10       0     0         0     0       10       0      0        0     0
 8        0     0         0     0       8        0      0        0     0
 10       0     2         0     0       10       0      2        0     0
```

If the input interface is not specified, the first interface of address family **inet** will be displayed.

Routing Table (Fourth Form)

The routing table display lists the available routes and the status of each. Each route consists of a destination host or network, and a gateway to use in forwarding packets. The *flags* column shows the status of the route (**U** if "up"), whether the route is to a gateway (**G**), and whether the route was created dynamically by a redirect (**D**). If the −**a** option is specified, there will be routing entries with flags for combined routing and address resolution entries (**A**), broadcast addresses (**B**), and the local addresses for the host (**L**).

Interface routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface.

The **refcnt** column gives the current number of routes that share the same link layer address.

The **use** column displays the number of packets sent using a combined routing and address resolution (**A**) or a broadcast (**B**) route. For a local (**L**) route, this count is the number of packets received, and for all other routes it is the number of times the routing entry has been used to create a new combined route and address resolution entry.

The *interface* entry indicates the network interface utilized for the route.

Multicast Routing Tables (Fifth Form)

The multicast routing table consists of the virtual interface table and the actual routing table.

DHCP Interface Information (Sixth Form)

The DHCP interface information consists of the interface name, its current state, lease information (when the lease began, when it will expire, and when renewal begins), and counts of the number of protocol exchanges done on behalf of the interface.

Below is a sample command line and output from a host with five interfaces under DHCP control:

```
# netstat –D

Interface  Status       Sent Received  Rejects
le0       BOUND         1       1       0
        (Began,Expires,Renew) = (12/04/1996 18:08, 12/04/1996 19:08,
12/04/1996 18:38) qe0       BOUND          1       1       0
        (Began,Expires,Renew) = (12/04/1996 18:08, 12/04/1996 19:08,
12/04/1996 18:38) qe1       BOUND          1       1       0
        (Began,Expires,Renew) = (12/04/1996 18:08, 12/04/1996 19:08,
12/04/1996 18:38) qe2       BOUND          1       1       0
        (Began,Expires,Renew) = (12/04/1996 18:08, 12/04/1996 19:08,
12/04/1996 18:38) qe3       SELECTING       4       0       0
#
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

dhcpagent(1M), **ifconfig**(1M), **iostat**(1M), **vmstat**(1M), **hosts**(4), **networks**(4), **protocols**(4), **services**(4), **attributes**(5), **inet**(7P)

NOTES

The kernel’s tables can change while **netstat** is examining them, creating incorrect or partial displays.

NAME

newfs – construct a new UFS file system

SYNOPSIS

newfs [–**Nv**] [*mkfs-options*] *raw-device*

DESCRIPTION

newfs is a "friendly" front-end to the **mkfs**(1M) program for making UFS file systems on disk partitions. **newfs** calculates the appropriate parameters to use and calls **mkfs**.

If run interactively (that is, standard input is a tty), **newfs** will prompt for confirmation before making the file system.

If the –**N** option is not specified and the inodes of the device are not randomized, **newfs** will call **fsirand**(1M).

You must be super-user to use this command, except when creating a UFS file system on a *diskette* (see **EXAMPLES**).

OPTIONS

–**N** Print out the file system parameters that would be used in creating the file system without actually creating the file system. **fsirand**(1M) is not called here.

–**v** Verbose. **newfs** prints out its actions, including the parameters passed to **mkfs**.

mkfs-options
Options that override the default parameters are:

–**a** *apc* The number of alternate blocks per cylinder (SCSI devices only) to reserve for bad block replacement. The default is **0**.

–**b** *bsize* The logical block size of the file system in bytes (either 4096 or 8192). The default is **8192**. (Note: The sun4u architecture does not support the **4096** block size.)

–**c** *cgize* The number of cylinders per cylinder group (ranging from 1 to 32). The default is **16**.

–**d** *gap* Rotational delay. The expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file. This parameter can be subsequently changed using the **tunefs**(1M) command. The default is disk-type dependent.

–**f** *fragsize* The smallest amount of disk space in bytes to allocate to a file. The values must be a power of two selected from the range 512 to the logical block size. If logical block size is 4096, legal values are 512, 1024, 2048 and 4096; if logical block size is 8192, 8192 is also a legal value. The default is **1024**.

–**i** *nbpi* The number of bytes per inode. This specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the fixed number of inodes to create. It should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given. The default is **2048**.

−m *free* The minimum percentage of free space to maintain in the file system. This space is off-limits to normal users. Once the file system is filled to this threshold, only the super-user can continue writing to the file system. This parameter can be subsequently changed using the **tunefs**(1M) command. The default is **10%**.

−n *nrpos* The number of different rotational positions in which to divide a cylinder group. The default is **8**.

−o *opt* (**space** or **time**). The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the minimum free space threshold (as specified by the **−m** option) is less than 10%, space optimization will be used.

−r *rpm* The speed of the disk in revolutions per minute. The default is **3600**.

−s *size* The size of the file system in sectors. The default is to use the entire partition.

−t *ntrack* The number of tracks per cylinder on the disk. The default is taken from the disk label.

−C *maxcontig* The maximum number of blocks, belonging to one file, that will be allocated contiguously before inserting a rotational delay. For a 4K file system, the default is **14**; for an 8K file system, the default is **7**. This parameter can be subsequently changed using the **tunefs**(1M) command.

Note: This parameter also controls clustering. Regardless of the value of *gap*, clustering is enabled only when *maxcontig* is greater than 1. Clustering allows higher I/O rates for sequential I/O and is described in **tunefs**(1M).

OPERANDS

raw-device The name of a raw special device residing in **/dev/rdsk** (for example, **/dev/rdsk/c0t0d0s6**) on which to create the file system.

USAGE

See **largefile**(5) for the description of the behavior of **newfs** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

The following example verbosely displays the parameters for the raw special device, **c0t0d0s6**, but does not actually create a new file system:

```
example# newfs −Nv /dev/rdsk/c0t0d0s6
mkfs −F ufs −o N /dev/rdsk/c0t0d0s6 1112940 54 15 8192 1024 16 10 60
        2048 t 0 −1 8 /dev/rdsk/c0t0d0s6:     1112940 sectors in
        1374 cylinders of 15 tracks, 54 sectors 569.8MB in 86 cyl
        groups (16 c/g, 6.64MB/g, 3072 i/g) super-block backups
```

```
          (for fsck −b #) at:
          32, 13056, 26080, 39104, 52128, 65152, 78176, 91200, 104224, . . .
```

The following example uses the command to create a UFS file system on a diskette that is managed by Volume Manager.

```
example%  newfs /vol/dev/aliases/floppy0
newfs: construct a new file system /vol/dev/aliases/floppy0: (y/n)? y
/vol/dev/aliases/floppy0:    2880 sectors in 80 cylinders of 2 tracks,
          18 sectors   1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
          super-block backups (for fsck −F ufs −o b=#) at:
          32, 640, 1184, 1792, 2336, . . .
```

EXIT STATUS

0 The operation was successful.

1, **10** Usage error or internal error. A message is output to **STDERR** explaining the error.

Other exit values may be returned by **mkfs**(1M), which is called by **newfs**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

fsck(1M), **fsck_ufs**(1M), **fsirand**(1M), **mkfs**(1M), **mkfs_ufs**(1M), **tunefs**(1M), **fs_ufs**(4), **attributes**(5), **largefile**(5)

DIAGNOSTICS

newfs: No such file or directory
The device specified does not exist, or a disk partition was not specified.

special: **cannot open**
You must be superuser to use this command.

NAME

newkey – create a new Diffie-Hellman key pair in the publickey database

SYNOPSIS

newkey –h *hostname* [**–s nisplus** | **nis** | **files**]
newkey –u *username* [**–s nisplus** | **nis** | **files**]

DESCRIPTION

newkey establishes new public keys for users and machines on the network. These keys are needed when using secure RPC or secure NFS service.

newkey prompts for a password for the given *username* or *hostname* and then creates a new public / secret Diffie-Hellman 192 bit key pair for the user or host. The secret key is encrypted with the given password. The key pair can be stored in the **/etc/publickey** file, the NIS **publickey** map, or the NIS+ **cred.org_dir** table.

newkey consults the **publickey** entry in the name service switch configuration file (see **nsswitch.conf**(4)) to determine which naming service is used to store the secure RPC keys. If the **publickey** entry specifies a unique name service, **newkey** will add the key in the specified name service. However, if there are multiple name services listed, **newkey** cannot decide which source to update and will display an error message. The user is required to specify the source explicitly with the **–s** option.

In the case of NIS, **newkey** should be run by the superuser on the master NIS server for that domain. In the case of NIS+, **newkey** should be run by the superuser on a machine which has permission to update the **cred.org_dir** table of the new user / host domain.

In the case of NIS+, **nisaddcred**(1M) should be used to add new keys.

OPTIONS

–h *hostname* Create a new public / secret key pair for the privileged user at the given *hostname*. Prompts for a password for the given *hostname*.

–u *username* Create a new public / secret key pair for the given *username*. Prompts for a password for the given *username*.

–s nisplus
–s nis
–s files Update the database in the specified source: **nisplus** (for NIS+), **nis** (for NIS), or **files**. Other sources may be available in the future.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

chkey(1), **keylogin**(1), **nisaddcred**(1M), **nisclient**(1M), **nsswitch.conf**(4), **publickey**(4), **attributes**(5)

NAME

nfsd – NFS daemon

SYNOPSIS

/usr/lib/nfs/nfsd [**–a**] [**–c** *#_conn*] [**–l** *listen_backlog*] [**–p** *protocol*] [**–t** *device*] [*nservers*]

DESCRIPTION

nfsd is the daemon that handles client file system requests. Only the super-user can run this daemon.

The **nfsd** daemon is automatically invoked in run level 3 with the **–a** option.

By default **nfsd** will start over the tcp and udp transports.

A previously invoked **nfsd** daemon started with or without options must be stopped before invoking another **nfsd** command.

OPTIONS

The following options are supported:

–a Start a NFS daemon over all available connectionless and connection-oriented transports, including udp and tcp.

–c *#_conn* This sets the maximum number of connections allowed to the NFS server over connection-oriented transports. By default, the number of connections is unlimited.

–l Set connection queue length for the NFS TCP over a connection-oriented transport. The default value is 32 entries.

–p *protocol* Start a NFS daemon over the specified protocol.

–t *device* Start a NFS daemon for the transport specified by the given device.

OPERANDS

The following operands are supported:

nservers This sets the maximum number of concurrent NFS requests that the server can handle. This concurrency is achieved by up to *nservers* threads created as needed in the kernel. *nservers* should be based on the load expected on this server. 16 is the usual number of *nservers*. If *nservers* is not specified, the maximum number of concurrent NFS requests will default to 1.

USAGE

If the **NFS_PORTMON** variable is set, then clients are required to use privileged ports (ports < **IPPORT_RESERVED**) in order to get NFS services. This variable is equal to zero by default. This variable has been moved from the "nfs" module to the "nfssrv" module. To set the variable, edit the **/etc/system** file and add this entry:

```
set nfssrv:nfs_portmon = 1
```

EXIT STATUS

0 Daemon started successfully.

1 Daemon failed to start.

FILES

.nfs*XXX*	client machine pointer to an open-but-unlinked file
/etc/init.d/nfs.server	shell script for starting **nfsd**
/etc/system	system configuration information file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ps(1), **mountd**(1M), **sharetab**(4), **system**(4), **attributes**(5)

NFS Administration Guide

NOTES

1. The NFS service uses kernel threads to process all of the NFS requests. Currently, system utilization associated with these threads is not charged to the **nfsd** process. Therefore, **ps**(1) can report 0 cpu time associated with the NFS daemon, even though NFS processing is taking place on the server.
2. Manually starting and restarting **nfsd** is not recommended. If it is necessary to do so, use the NFS server start / stop script (**/etc/init.d/nfs.server**). See *NFS Administration Guide* for more information.

NAME

nfsstat – NFS statistics

SYNOPSIS

nfsstat [**–cmnrsz**]

DESCRIPTION

nfsstat displays statistical information about the NFS and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is

nfsstat –cnrs

That is, display everything, but reinitialize nothing.

OPTIONS

–c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the **–n** and **–r** options to print client NFS or client RPC information only.

–m Display statistics for each NFS mounted file system. This includes the server name and address, mount flags, current read and write sizes, the retransmission count, and the timers used for dynamic retransmission. The **srtt** value contains the smoothed round trip time, the **dev** value contains the estimated deviation, and the **cur** value is the current backed-off retransmission value.

–n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the **–c** and **–s** options to print client or server NFS information only.

–r Display RPC information.

–s Display server information.

–z Zero (reinitialize) statistics. This option is for use by the super user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

DISPLAYS

The server RPC display includes the following fields:

calls The total number of RPC calls received.

badcalls The total number of calls rejected by the RPC layer (the sum of **badlen** and **xdrcall** as defined below).

nullrecv The number of times an RPC call was not available when it was thought to be received.

badlen The number of RPC calls with a length shorter than a minimum-sized RPC call.

xdrcall The number of RPC calls whose header could not be XDR decoded.

dupchecks The number of RPC calls that looked up in the duplicate request cache.

dupreqs The number of RPC calls that were found to be duplicates.

The server NFS display shows the number of NFS calls received (**calls**) and rejected (**badcalls**), and the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

calls The total number of RPC calls made.

badcalls The total number of calls rejected by the RPC layer.

badxids The number of times a reply from a server was received which did not correspond to any outstanding call.

timeouts The number of times a call timed out while waiting for a reply from the server.

newcreds The number of times authentication information had to be refreshed.

badverfs The number of times the call failed due to a bad verifier in the response.

timers The number of times the calculated time-out value was greater than or equal to the minimum specified time-out value for a call.

cantconn The number of times the call failed due to a failure to make a connection to the server.

nomem The number of times the call failed due to a failure to allocate memory.

interrupts The number of times the call was interrupted by a signal before completing.

retrans The number of times a call had to be retransmitted due to a timeout while waiting for a reply from the server. Applicable only to RPC over connection-less transports.

The client NFS display shows the number of calls sent and rejected, as well as the number of times a **CLIENT** handle was received (**clgets**), the number of times the **CLIENT** handle cache had no unused entries (**cltoomany**), as well as a count of the various calls and their respective percentages.

The −**m** option includes information about mount flags set by mount options, mount flags internal to the system, and other mount information. See **mount_nfs**(1M).

The following fields provide failover information:

noresponse How many times servers have failed to respond.

failover How many times a new server has been selected.

remap How may times files have been re-evaluated to the new server.

currserver Which server is currently providing NFS service. See the *NFS Administration Guide* for additional details.

The following mount flags are set by mount options:

sec (one of the following values):

none	No authentication.
unix	UNIX style authentication (UID, GID).
short	Short hand UNIX style authentication.
des	**des** style authentication (encrypted timestamps).

krb4 **kerberos** style authentication.

hard Hard mount.

soft Soft mount.

intr Interrupts allowed on hard mount.

nointr No interrupts allowed on hard mount.

noac Client is not caching attributes.

rsize Read buffer size in bytes.

wsize Write buffer size in bytes.

retrans NFS retransmissions.

nocto No close-to-open consistency.

llock Local locking being used (no lock manager).

grpid System V group id inheritance.

rpctimesync RPC time sync.

The following mount flags are internal to the system:

printed "Not responding" message printed.

down Server is down.

dynamic Dynamic transfer size adjustment.

link Server supports links.

symlink Server supports symbolic links.

readdir Use **readdir** instead of **readdirplus**.

acl Server supports NFS_ACL.

The following flags relate to additional mount information:

vers NFS version.

proto Protocol.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mount_nfs(1M), **attributes**(5)
Solaris Advanced Installation Guide
NFS Administration Guide

NAME

nisaddcred – create NIS+ credentials

SYNOPSIS

nisaddcred [**–p** *principal*] [**–P** *nis_principal*] [**–l** *login_password*] *auth_type* [*domain_name*]

nisaddcred **–r** [*nis_principal*] [*domain_name*]

DESCRIPTION

The **nisaddcred** command is used to create security credentials for NIS+ principals. NIS+ credentials serve two purposes. The first is to provide authentication information to various services; the second is to map the authentication service name into a NIS+ principal name.

When the **nisaddcred** command is run, these credentials get created and stored in a table named **cred.org_dir** in the default NIS+ domain. If *domain_name* is specified, the entries are stored in the **cred.org_dir** of the specified domain. The specified domain must either be the one to which you belong, or one in which you are authenticated and authorized to create credentials, i.e., a subdomain. Note that the credentials of normal users must be stored in the same domain as their passwords.

It is simpler to add credentials using **nisclient**(1M), because it obtains the required information itself. **nispopulate**(1M) is used for “bulk” updates and can also be used to add credentials for entries in the **hosts** and the **passwd** NIS+ tables.

NIS+ principal names are used in specifying clients that have access rights to NIS+ objects. For more details, refer to the “Principal Names” subsection of the **nis+**(1) manual page. See **nischmod**(1), **nischown**(1), **nis_objects**(3N), and **nis_groups**(3N). Various other services can also implement access control based on these principal names.

The **cred.org_dir** table is organized as follows :

cname	auth_type	auth_name	public_data	private_data
user1.foo.com.	LOCAL	2990	10,102,44	
user1.foo.com.	DES	unix.2990@foo.com	098...819	3b8...ab2

The **cname** column contains a canonical representation of the NIS+ principal name. By convention, this name is the login name of a user, or the host name of a machine, followed by a dot (’.’) followed by the fully qualified “home” domain of that principal. For users, the home domain is defined to be the domain where their DES credentials are kept. For hosts, their home domain is defined to be the domain name returned by the **domainname**(1M) command executed on that host.

There are two types of *auth_type* entries in the **cred.org_dir** table. Those with authentication type LOCAL, and those with authentication type DES. *auth_type*, specified on the command line in upper or lower case, should be either *local* or *des*.

Entries of type LOCAL are used by the NIS+ service to determine the correspondence between fully qualified NIS+ principal names and users identified by UIDs in the domain containing the **cred.org_dir** table.

This correspondence is required when associating requests made using the **AUTH_SYS** RPC authentication flavor (see **rpc_clnt_auth**(3N)) to a NIS+ principal name. It is also required for mapping a UID in one domain to its fully qualified NIS+ principal name whose home domain may be elsewhere. The principal's credentials for any authentication flavor may then be sought for within the **cred.org_dir** table in the principal's home domain (extracted from the principal name). The same NIS+ principal may have LOCAL credential entries in more than one domain. Only users, and not machines, have LOCAL credentials. In their home domain, users of NIS+ should have both types of credentials.

The *auth_name* associated with the LOCAL type entry is a UID that is valid for the principal in the domain containing the **cred.org_dir** table. This may differ from that in the principal's home domain. The public information stored in *public_data* for this type contains a list of GIDs for groups in which the user is a member. The GIDs also apply to the domain in which the table resides. There is no private data associated with this type. Neither a UID nor a principal name should appear more than once among the LOCAL entries in any one **cred.org_dir** table.

The DES *auth_type* is used for Secure RPC authentication (see **secure_rpc**(3N)).

The authentication name associated with the DES *auth_type* is a Secure RPC *netname*. A Secure RPC netname has the form **unix.***id@domain.com*, where *domain* must be the same as the domain of the principal. For principals that are users the *id* must be the UID of the principal in the principal's home domain. For principals that are hosts, the *id* is the host's name. In Secure RPC, processes running under effective UID 0 (root) are identified with the host principal. Unlike LOCAL, there cannot be more than one DES credential entry for one NIS+ principal in the NIS+ namespace.

The public information in an entry of authentication type DES is the public key for the principal. The private information in this entry is the private key of the principal encrypted by the principal's network password.

User clients of NIS+ should have credentials of both types in their home domain. In addition, a principal must have a LOCAL entry in the **cred.org_dir** table of each domain from which the principal wishes to make authenticated requests. A client of NIS+ that makes a request from a domain in which it does not have a LOCAL entry will be unable to acquire DES credentials. A NIS+ service running at security level 2 or higher will consider such users unauthenticated and assign them the name *nobody* for determining access rights.

This command can only be run by those NIS+ principals who are authorized to add or delete the entries in the **cred** table.

If credentials are being added for the caller itself, **nisaddcred** automatically performs a keylogin for the caller.

You can list the **cred** entries for a particular principal with **nismatch**(1).

OPTIONS

−p *principal* The name *principal* specifies the name of the principal as defined by the naming rules for that specific mechanism. For example, LOCAL credential names are supplied with this option by including a string specifying a UID. For DES credentials, the name should be a Secure RPC netname of the form **unix.***id@domain.com*, as described earlier. If the **−p** option is not specified, the *auth_name* field is constructed from the effective UID of the current process and the name of the local domain.

–P *nis_principal* Use the NIS+ principal name *nis_principal*. This option should be used when creating LOCAL or DES credentials for users whose home domain is different than the local machine's default domain.

Whenever the **–P** option is not specified, **nisaddcred** constructs a principal name for the entry as follows. When it is not creating an entry of type LOCAL, **nisaddcred** calls **nis_local_principal**, which looks for an existing LOCAL entry for the effective UID of the current process in the **cred.org_dir** table and uses the associated principal name for the new entry. When creating an entry of authentication type LOCAL, **nisaddcred** constructs a default NIS+ principal name by taking the login name of the effective UID for its own process, and appending to it a dot ('.') followed by the local machine's default domain. If the caller is a superuser, the machine name is used instead of the login name.

–l *login_password*

Use the *login_password* specified as the password to encrypt the secret key for the credential entry. This overrides the prompting for a password from the shell. This option is intended for administration scripts only. Prompting guarantees not only that no one can see your password on the command line using **ps**(1) but it also checks to make sure you have not made any mistakes. **NOTE:** *login_password* does not really HAVE to be the user's password but if it is, it simplifies logging in.

–r [*nis_principal*]

Remove all credentials associated with the principal *nis_principal* from the **cred.org_dir** table. This option can be used when removing a client or user from the system. If *nis_principal* is not specified the default is to remove credentials for the current *user*. If *domain_name* is not specified, the operation is executed in the default NIS+ domain.

EXIT CODES

This command returns **0** on success and **1** on failure.

EXAMPLES

The following examples illustrate how to add the LOCAL and DES credentials for some user, user1, with a UID of **2990**, who is an NIS+ user principal in the **some.domain.com.** NIS+ domain.

```
example% nisaddcred –p 2990 \
        –P user1.some.domain.com. local
```

Note that credentials are always added in the **cred.org_dir** table in the domain where **nisaddcred** is run, unless *domain_name* is specified as the last parameter on the command line. If credentials are being added from the domain server for its clients, then *domain_name* should be specified. The caller should have adequate permissions to create entries in the **cred.org_dir** table.

The system administrator can add a DES credential for the same user, using the following example:

```
example% nisaddcred –p unix.2990@some.domain.com \
          –P user1.some.domain.com. des
```

Please note that DES credentials can be added only after the LOCAL credentials have been added.

Note that the secure RPC netname does not end with a dot ('.') while the NIS+ principal name (specified with the –**P** option) does. This command should be executed from a machine in the same domain as is the user.

The following example shows how to add a machine's DES credentials in the same domain.

```
example% nisaddcred –p unix.foo@some.domain.com \
          –P foo.some.domain.com. des
```

Please note that no LOCAL credentials are needed in this case.

The following example illustrates how to add a NIS+ workstation's principal DES credential.

```
example% nisaddcred –p unix.host1@sub.some.domain.com \
          –P newhost.sub.some.domain.com. des sub.some.domain.com.
```

This format is particularly useful if you are running this command from a server which is in a higher domain than sub.some.domain.com. Without the last option for domain name, **nisaddcred** would fail because it would attempt to use the default domain of some.domain.com.

The following example illustrates adding DES credentials without being prompted for the root login password.

```
example% nisaddcred –p unix.2990@some.domain.com \
          –P user1.some.domain.com. -l login_password des
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

chkey(1), **keylogin**(1), **nis+**(1), **nischmod**(1), **nischown**(1), **nismatch**(1), **nistbladm**(1), **ps**(1), **domainname**(1M), **nisclient**(1M), **nispopulate**(1M), **nis_groups**(3N), **nis_local_names**(3N), **nis_objects**(3N), **rpc_clnt_auth**(3N), **secure_rpc**(3N), **attributes**(5)

NOTES

The **cred.org_dir** NIS+ table replaces the maps *publickey.byname* and *netid.byname* used in NIS (YP).

NAME

nisaddent – create NIS+ tables from corresponding /etc files or NIS maps

SYNOPSIS

/usr/lib/nis/nisaddent [–**D** *defaults*] [–**Parv**] [–**t** *table*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent [–**D** *defaults*] [–**Paprmv**] –**f** *file* [–**t** *table*] *type* [*nisdomain*]

/usr/lib/nis/nisaddent [–**D** *defaults*] [–**Parmv**] [–**t** *table*] –**y** *ypdomain* [–**Y** *map*]
type [*nisdomain*]

/usr/lib/nis/nisaddent –d [–AMq] [–**t** *table*] *type* [*nisdomain*]

DESCRIPTION

nisaddent creates entries in NIS+ tables from their corresponding **/etc** files and NIS maps. This operation is customized for each of the standard tables that are used in the administration of Solaris systems. The *type* argument specifies the type of the data being processed. Legal values for this type are one of **aliases**, **bootparams**, **ethers**, **group**, **hosts**, **netid**, **netmasks**, **networks**, **passwd**, **protocols**, **publickey**, **rpc**, **services**, **shadow**, or **timezone** for the standard tables, or **key-value** for a generic two-column (key, value) table. For a site specific table, which is not of **key-value** type, one can use **nistbladm**(1) to administer it.

The NIS+ tables should have already been created by **nistbladm**(1), **nissetup**(1M), or **nisserver**(1M).

It is easier to use **nispopulate**(1M) instead of **nisaddent** to populate the system tables.

By default, **nisaddent** reads from the standard input and adds this data to the NIS+ table associated with the *type* specified on the command line. An alternate NIS+ table may be specified with the –**t** option. For type **key-value**, a table specification is required.

Note that the *data* type can be different than the table name (–**t**). For example, the automounter tables have **key-value** as the table type.

Although, there is a *shadow* data type, there is no corresponding *shadow* table. Both the shadow and the passwd data is stored in the **passwd** table itself.

Files may be processed using the –**f** option, and NIS version 2 (YP) maps may be processed using the –**y** option. The merge option is not available when reading data from standard input.

When a *ypdomain* is specified, the **nisaddent** command takes its input from the **dbm** files for the appropriate NIS map (**mail.aliases**, **bootparams**, **ethers.byaddr**, **group.byname**, **hosts.byaddr**, **netid.byname**, **netmasks.byaddr**, **networks.byname**, **passwd.byname**, **protocols.byname**, **publickey.byname**, **rpc.bynumber**, **services.byname**, or **timezone.byname**). An alternate NIS map may be specified with the –**Y** option. For type **key-value**, a map specification is required. The map must be in the **/var/yp/***ypdomain* directory on the local machine. Note that *ypdomain* is case sensitive. **ypxfr**(1M) can be used to get the NIS maps.

If a *nisdomain* is specified, **nisaddent** operates on the NIS+ table in that NIS+ domain, otherwise the default domain is used.

In terms of performance, loading up the tables is fastest when done through the dbm files (−**y**).

OPTIONS

−a Add the file or map to the NIS+ table without deleting any existing entries. This option is the default. Note that this mode only propagates additions and modifications, not deletions.

−p Process the password field when loading password information from a file. By default, the password field is ignored because it is usually not valid (the actual password appears in a shadow file).

−q Dump tables in "quick" mode. The default method for dumping tables processes each entry individually. For some tables (e.g., hosts), multiple entries must be combined into a single line, so extra requests to the server must be made. In "quick" mode, all of the entries for a table are retrieved in one call to the server, so the table can be dumped more quickly. However, for large tables, there is a chance that the process will run out of virtual memory and the table will not be dumped.

−r Replace the file or map in the existing NIS+ table by first deleting any existing entries, and then add the entries from the source (**/etc** files, or NIS+ maps). This option has the same effect as the −**m** option. The use of this option is *strongly* discouraged due to its adverse impact on performance, unless there are a large number of changes.

−m Merge the file or map with the NIS+ table. This is the most efficient way to bring an NIS+ table up to date with a file or NIS map when there are only a small number of changes. This option adds entries that are not already in the database, modifies entries that already exist (if changed), and deletes any entries that are not in the source. Use the −**m** option whenever the database is large and replicated, and the map being loaded differs only in a few entries. This option reduces the number of update messages that have to be sent to the replicas. Also see the −**r** option.

−d Dump the NIS+ table to the standard output in the appropriate format for the given *type*. For tables of type **key-value**, use **niscat**(1) instead. To dump the **cred** table, dump the **publickey** and the **netid** types.

−v Verbose.

−f *file* Specify that *file* should be used as the source of input (instead of the standard input).

−y *ypdomain*
Use the **dbm** files for the appropriate NIS map, from the NIS domain *ypdomain*, as the source of input. The files are expected to be on the local machine in the **/var/yp/***ypdomain* directory. If the machine is not an NIS server, use **ypxfr**(1M) to get a copy of the **dbm** files for the appropriate map.

−Y *map* Use the **dbm** files for *map* as the source of input.

−t *table* Specify that *table* should be the NIS+ table for this operation. This should be a relative name as compared to your default domain or the *domainname* if it has been specified.

–P Follow concatenation path. This option specifies that lookups should follow the concatenation path of a table if the initial search is unsuccessful.

–A All data. This option specifies that the data within the table and all of the data in tables in the initial table's concatenation path be returned.

–M Master server only. This option specifies that lookups should be sent to the master server. This guarantees that the most up-to-date information is seen at the possible expense that the master server may be busy, or that it may be made busy by this operation.

–D *defaults*

This option specifies a different set of defaults to be used during this operation. The *defaults* string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=*time*

This token sets the default time to live for objects that are created by this command. The value *time* is specified in the format as defined by the **nischttl**(1) command. The default is 12 hours.

owner=*ownername*

This token specifies that the NIS+ principal *ownername* should own the created object. The default for this value is the principal who is executing the command.

group=*groupname*

This token specifies that the group *groupname* should be the group owner for the object that is created. The default is **NULL**.

access=*rights*

This token specifies the set of access rights that are to be granted for the given object. The value *rights* is specified in the format as defined by the **nischmod**(1) command. The default is −−−−**rmcdr**−−−**r**−−−.

EXAMPLES

This example adds the contents of **/etc/passwd** to the **passwd.org_dir** table.

```
example% cat /etc/passwd | nisaddent passwd
```

The next example adds the shadow information; note that the table type here is "shadow", not "passwd", even though the actual information is stored in the **passwd** table.

```
example% cat /etc/shadow | nisaddent shadow
```

This example replaces the **hosts.org_dir** table with the contents of **/etc/hosts** (in verbose mode).

```
example% nisaddent –rv –f /etc/hosts hosts
```

This example merges the **passwd** map from **yypdomain** with the **passwd.org_dir.nisdomain** table (in verbose mode). The example assumes that the **/var/yp/myypdomain** directory contains the **yppasswd** map.

```
example% nisaddent –mv –y myypdomain passwd nisdomain
```

This example merges the **auto.master** map from **myypdomain** with the **auto_master.org_dir** table.

```
example% nisaddent –m –y myypdomain –Y auto.master \
          –t auto_master.org_dir key-value
```

This example dumps the **hosts.org_dir** table.

```
example% nisaddent –d hosts
```

ENVIRONMENT

NIS_DEFAULTS This variable contains a default string that will override the NIS+ standard defaults. If the **–D** switch is used, those values will then override both the **NIS_DEFAULTS** variable and the standard defaults. To avoid security accidents, the access rights in the **NIS_DEFAULTS** variable are ignored for the **passwd** table (but access rights specified with **–D** are used).

NIS_PATH If this variable is set, and neither the *nisdomain* nor the *table* are fully qualified, each directory specified in **NIS_PATH** will be searched until the table is found (see **nisdefaults**(1)).

EXIT CODES

0 Success.

1 Failure caused by an error other than parsing.

2 A parsing error occurred on an entry. A parsing error does not cause termination; the invalid entries are simply skipped.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

niscat(1), **nischmod**(1), **nisdefaults**(1), **nistbladm**(1), **nispopulate**(1M), **nisserver**(1M), **nissetup**(1M), **ypxfr**(1M), **hosts**(4), **passwd**(4), **shadow**(4), **attributes**(5)

NAME

nisbackup – backup NIS+ directories

SYNOPSIS

nisbackup [–**v**] *backup-dir directory* **. . .**
nisbackup [–**v**] –**a** *backup-dir*

DESCRIPTION

nisbackup backs up a NIS+ directory object on a NIS+ master server. Updates to the NIS+ database will be temporarily disabled while **nisbackup** is running. The *backup-dir* is a UNIX directory that must exist prior to running **nisbackup**. The **nisbackup** command can be used to backup an individual NIS+ directory object or all (–**a**) of the NIS+ directory objects served by a master server. The NIS+ directory objects being backed up will be placed into subdirectories under the *backup-dir* directory. These subdirectories are named according to the NIS+ directory object they contain. **nisbackup** operates on individual NIS+ directory objects (for example, **org_dir.wiz.com**). This allows an administrator to selectively backup specific directories.

The **rpc.nisd**(1M) process must be running on the master server with a stable NIS+ database for **nisbackup** to complete. **nisbackup** will not attempt to correct any corruption in the NIS+ database, so it is important that backups be done regularly as part of the NIS+ administration.

The first synopsis is used to backup a single NIS+ directory object or a list of NIS+ directory objects. The objects can be partially qualified or fully qualified. The machine on which the command is executing must be the master for the NIS+ directory objects specified.

The second synopsis will backup all of the NIS+ directory objects that are served by this master. The –**a** option is the recommended method of backing up a master server, since it will backup all NIS+ directory objects that are served by this master. If this server is a master server for more than one domain, the backup will include NIS+ directories that belong to all of the domains served. Individual NIS+ directory objects can be selected for restoring from a *backup-dir* created with the –**a** option (see **nisrestore**(1M)).

OPTIONS

–**a** Creates a backup of all NIS+ directory objects for which this server is a master.

–**v** Verbose option. Additional output will be produced and sent to **syslog**(3) upon execution of the command (see **syslog.conf**(4)).

OPERANDS

backup-dir The directory into which the subdirectories containing the backed up objects are placed. This must be created prior to running **nisbackup**.

directory The NIS+ directory object(s) being backed up.

EXAMPLES

To backup the **org_dir** NIS+ directory object of the domain **foo.com** on a master server to a directory named **/backup**:

```
master_server# nisbackup /backup org_dir.foo.com.
```

To backup the entire NIS+ domain **foo.com** to a directory named **/backup**:

```
master_server# nisbackup /backup foo.com. \
            org_dir.foo.com. groups_dir.foo.com. \
            ctx_dir.foo.com.
```

To backup an entire NIS+ database to a backup directory named **/backup**:

```
master_server# nisbackup –a /backup
```

EXIT STATUS

0 Successful completion.

1 An error occurred.

FILES

/backup-dir/backup_list This ascii file contains a list of all the objects contained in this *backup-dir* directory.

/backup-dir/directory-object A subdirectory that is created in the *backup-dir* that contains the NIS+ directory-object backup.

*/backup-dir/directory-object/***data** A subdirectory that contains the data files that are part of the NIS+ directory-object backup.

*/backup-dir/directory-object/***last.upd** This data file contains timestamp information about the directory-object.

*/backup-dir/directory-object/***data.dict** A NIS+ data dictionary for all of the objects contained in the NIS+ directory-object backup.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nis+(1), **nisdefaults**(1), **nisrm**(1), **nisrestore**(1M), **rpc.nisd**(1M), **syslog**(3), **xfn**(3N), **nisfiles**(4), **syslog.conf**(4), **attributes**(5)

NOTES

The –**a** option only includes directory objects for which this server is the master. It is possible, but not recommended, to configure a master server as a replica for other domains. The objects belonging to those replicated domains will not be backed up with the –**a** option. The backup of replicated objects must be run on the master server for those objects.

Do not use the same *backup-dir* to backup different master servers. Each master server must have its own *backup-dir*.

nisbackup will set the **rpc.nisd**(1M) to read only mode, which will disable updates to the NIS+ database. This is neccessary to ensure the consistency of the backup. For this reason, **nisbackup** should not be run while large numbers of updates are being applied to the NIS+ database. Update utilities such as **nisaddent**(1M) should not be run simultaneously with **nisbackup**.

NAME

nis_cachemgr – NIS+ utility to cache location information about NIS+ servers

SYNOPSIS

/usr/sbin/nis_cachemgr [**–i**] [**–n**] [**–v**]

DESCRIPTION

The **nis_cachemgr** daemon maintains a cache of NIS+ directory objects and active servers for domains. It is responsible for locating servers for a domain on behalf of client processes. This improves performance because only one process has to search for servers. The cache contains location information necessary to contact the NIS+ servers. This includes transport addresses, information neeeded to authenticate the server, and a time to live field which gives a hint on how long the directory object can be cached. The cache helps to improve the performance of the clients that are traversing the NIS+ name space. **nis_cachemgr** should be running on all the machines that are using NIS+. However, it is not required that the **nis_cachemgr** program be running in order for NIS+ requests to be serviced.

The cache maintained by this program is shared by all the processes that access NIS+ on a machine. The cache is maintained in a file that is memory mapped (see **mmap**(2)) by all the processes. On start up, **nis_cachemgr** initializes the cache from the cold start file (see **nisinit**(1M)) and preserves unexpired entries that already exist in the cache file. Thus, the cache survives machine reboots.

The **nis_cachemgr** program is normally started from a system startup script. **nisshowcache**(1M) can be used to look at the cached objects and active servers.

The **nisprefadm**(1M) command can be used to control which NIS+ servers the **nis_cachemgr** program will try to select.

Note: The **nis_cachemgr** program makes NIS+ requests under the NIS+ principal name of the host on which it runs. Before running **nis_cachemgr**, security credentials for the host should be added to the **cred.org_dir** table in the host’s domain using **nisaddcred**(1M). Credentials of type DES will be needed if the NIS+ service is operating at security level 2 (see **rpc.nisd**(1M)). See the **WARNINGS** section, below. Additionally, a "**keylogin –r**" should be done on the machine.

OPTIONS

–i Force **nis_cachemgr** to ignore the previous cache file and reinitialize the cache from just the cold start file. By default, the cache manager initializes itself from both the cold start file and the old cache file, thereby maintaining the entries in the cache across machine reboots.

–v This flag sets verbose mode. In this mode, the **nis_cachemgr** program logs not only errors and warnings, but also additional status messages. The additional messages are logged using **syslog**(3) with a priority of **LOG_INFO**.

FILES

/var/nis/NIS_SHARED_DIRCACHE	the shared cache file
/var/nis/NIS_COLD_START	the coldstart file

/etc/init.d/rpc initialization scripts for NIS+

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

keylogin(1), **nisaddcred**(1M), **nisinit**(1M), **nisprefadm**(1M), **nisshowcache**(1M), **rpc.nisd**(1M), **mmap**(2), **rpc**(3N), **syslog**(3), **nisfiles**(4), **attributes**(5)

DIAGNOSTICS

The **nis_cachemgr** daemon logs error messages and warnings using **syslog**(3). Error messages are logged to the **DAEMON** facility with a priority of **LOG_ERR** . Warning messages are logged with a priority of **LOG_WARNING**. Additional status messages can be obtained using the **–v** option.

NAME

nisclient – initialize NIS+ credentials for NIS+ principals

SYNOPSIS

/usr/lib/nis/nisclient -c [**–x**] [**–o**] [**–v**] [**–l** *<network_password>*]
[**–d** *<NIS+_domain>*] *client_name* **. . .**

/usr/lib/nis/nisclient -i [**–x**] [**–v**] **–h** *<NIS+_server_host>*
[**–a** *<NIS+_server_addr>*] [**–d** *<NIS+_domain>*] [**–S** *0* | *2*]

/usr/lib/nis/nisclient -u [**–x**] [**–v**]

/usr/lib/nis/nisclient -r [**–x**]

DESCRIPTION

The nisclient shell script can be used to:

- create NIS+ credentials for hosts and users
- initialize NIS+ hosts and users
- restore the network service environment

NIS+ credentials are used to provide authentication information of NIS+ clients to NIS+ service.

Use the first synopsis (**–c**) to create individual NIS+ credentials for hosts or users. You must be logged in as a NIS+ principal in the domain for which you are creating the new credentials. You must also have write permission to the local "cred" table. The *client_name* argument accepts any valid host or user name in the NIS+ domain (for example, the *client_name* must exist in the hosts or passwd table). **nisclient** verifies each *client_name* against both the **host** and **passwd** tables, then adds the proper NIS+ credentials for hosts or users. Note that if you are creating NIS+ credentials outside of your local domain, the host or user must exist in the **host** or **passwd** tables in both the local and remote domains.

By default, **nisclient** will not overwrite existing entries in the credential table for the hosts and users specified. To overwrite, use the **–o** option. After the credentials have been created, **nisclient** will print the command that must be executed on the client machine to initialize the host or the user. The **–c** option requires a network password for the client which is used to encrypt the secret key for the client. You can either specify it on the command line with the **–l** option or the script will prompt you for it. You can change this network password later with **nispasswd**(1) or **chkey**(1).

nisclient -c is not intended to be used to create NIS+ credentials for all users and hosts which are defined in the passwd and hosts tables. To define credentials for all users and hosts, use **nispopulate**(1M).

Use the second synopsis (**–i**) to initialize a NIS+ client machine. **–i** option can be used to convert machines to use NIS+ or to change the machine's domainname. You must be logged in as super-user on the machine that is to become a NIS+ client. Your administrator must have already created the NIS+ credential for this host by using **nisclient -c** or **nispopulate –C**. You will need the network password your administrator created. **nisclient** will prompt you for the network password to decrypt your secret key and then for this machine's root login password to generate a new set of secret / public keys. If the NIS+ credential was created by your administrator using **nisclient -c**, then you can simply use the initialization command that was printed by the **nisclient** script to initialize this host instead of typing it manually.

To initialize an unauthenticated NIS+ client machine, use the "–**i**" option with "–**S** *0*". With these options, the **nisclient -i** option will not ask for any passwords.

During the client initialization process, files that are being modified are backed up as <files>.no_nisplus. The files that are usually modified during a client initialization are: **/etc/defaultdomain**, **/etc/nsswitch.conf**, **/etc/inet/hosts**, and, if it exists, **/var/nis/NIS_COLD_START**. Note that a file will not be saved if a backup file already exists.

The –**i** option does not set up an NIS+ client to resolve hostnames using DNS. Please refer to the DNS documentation for information on setting up DNS. (See **resolv.conf**(4)).

It is not necessary to initialize either NIS+ root master servers or machines that were installed as NIS+ clients using **suninstall**(1M).

Use the third synopsis (–**u**) to initialize a NIS+ user. You must be logged in as the user on a NIS+ client machine in the domain where your NIS+ credentials have been created. Your administrator should have already created the NIS+ credential for your username using **nisclient -c** or **nispopulate**(1M). You will need the network password your administrator used to create the NIS+ credential for your username. **nisclient** will prompt you for this network password to decrypt your secret key and then for your login password to generate a new set of secret / public keys.

Use the fourth synopsis (–**r**) to restore the network service environment to whatever you were using before **nisclient -i** was executed. You must be logged in as super-user on the machine that is to be restored. The restore will only work if the machine was initialized with **nisclient -i** because it uses the backup files created by the –**i** option.

Reboot the machine after initializing a machine or restoring the network service.

OPTIONS

–**a** <*NIS+_server_addr*> Specifies the IP address for the NIS+ server. This option is used *only* with the –**i** option.

–**c** Adds **DES** credentials for NIS+ principals.

–**d** <*NIS+_domain*> Specifies the NIS+ domain where the credential should be created when used in conjuction with the –**c** option. It specifies the name for the new NIS+ domain when used in conjuction with the –**i** option. The default is your current domainname.

–**h** <*NIS+_server_host*> Specifies the NIS+ server's hostname. This option is used *only* with the –**i** option.

–**i** Initializes an NIS+ client machine.

–**l** <*network_password*> Specifies the network password for the clients. This option is used *only* with the –**c** option. If this option is not specified, the script will prompt you for the network password.

–**o** Overwrite existing credential entries. The default is not to overwrite. This is used *only*with the –**c** option.

–r restores the network service environment.

–S *0|2* Specifies the authentication level for the NIS+ client. Level **0** is for unauthenticated clients and level **2** is for authenticated (**DES**) clients. The default is to set up with level **2** authentication. This is used *only* with the **–i** option. **nisclient** always uses level **2** authentication (**DES**) for both **–c** and **–u** options. There is no need to run **nisclient** with **–u** and **–c** for level **0** authentication.

–u Initializes an NIS+ user.

–v Runs the script in verbose mode.

–x turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.

EXAMPLES

To add the **DES** credential for host *sunws* and user *fred* in the local domain:

```
example% /usr/lib/nis/nisclient –c sunws fred
```

To add the **DES** credential for host *sunws* and user *fred* in domain *xyz.sun.com.*:

```
example% /usr/lib/nis/nisclient –c –d xyz.sun.com. sunws fred
```

To initialize host *sunws* as an NIS+ client in domain *xyz.sun.com.* where *nisplus_server* is a server for the domain *xyz.sun.com.*:

```
example# /usr/lib/nis/nisclient –i –h nisplus_server –d xyz.sun.com.
```

The script will prompt you for the IP address of *nisplus_server* if the server is not found in the **/etc/hosts** file. The **–d** option is needed only if your current domain name is different from the new domain name.

To initialize host sunws as an unauthenticated NIS+ client in domain xyz.sun.com. where nisplus_server is a server for the domain xyz.sun.com.:

```
example# /usr/lib/nis/nisclient -i -S 0 \
              -h nisplus_server -d xyz.sun.com. -a 129.140.44.1
```

To initialize user *fred* as an NIS+ principal, log in as user *fred* on an NIS+ client machine.

```
example% /usr/lib/nis/nisclient –u
```

FILES

/var/nis/NIS_COLD_START
This file contains a list of servers, their transport addresses, and their Secure RPC public keys that serve the machines default domain.

/etc/defaultdomain the system default domainname

/etc/nsswitch.conf configuration file for the name-service switch

/etc/inet/hosts local host name database

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

chkey(1), **keylogin**(1), **nis+**(1), **nispasswd**(1), **keyserv**(1M), **nisaddcred**(1M), **nisinit**(1M), **nispopulate**(1M), **suninstall**(1M), **nsswitch.conf**(4), **resolv.conf**(4), **attributes**(5)

NAME

nisinit – NIS+ client and server initialization utility

SYNOPSIS

nisinit **–r**
nisinit **–p Y | D | N** *parent_domain host . . .*
nisinit **–c –H** *host* | **–B** | **–C** *coldstart*

DESCRIPTION

nisinit initializes a machine to be a NIS+ client or an NIS+ root master server. It may be easier to use **nisclient**(1M) or **nisserver**(1M) to accomplish this same task.

OPTIONS

–r Initialize the machine to be a NIS+ root server. This option creates the file **/var/nis/data/root.object** and initialize it to contain information about this machine. It uses the **sysinfo**(2) system call to retrieve the name of the default domain.

To initialize the machine as an NIS+ root server, it is advisable to use the "**–r**" option of **nisserver**(1M), instead of using "**nisinit -r**".

–p Y | D | N *parent_domain host . . .*
This option is used on a root server to initialize a **/var/nis/data/parent.object** to make this domain a part of the namespace above it. Only root servers can have parent objects. A parent object describes the namespace ''above'' the NIS+ root. If this is an isolated domain, this option should not be used. The argument to this option tells the command what type of name server is serving the domain above the NIS+ domain. When clients attempt to resolve a name that is outside of the NIS+ namespace, this object is returned with the error **NIS_FOREIGNNS** indicating that a name space boundary has been reached. It is up to the client to continue the name resolution process.

The parameter *parent_domain* is the name of the parent domain in a syntax that is native to that type of domain. The list of host names that follow the domain parameter are the names of hosts that serve the parent domain. If there is more than one server for a parent domain, the first host specified should be the master server for that domain.

Y Specifies that the parent directory is a NIS version 2 domain.

D Specifies that the parent directory is a DNS domain.

N Specifies that the parent directory is another NIS+ domain. This option is useful for connecting a pre-existing NIS+ subtree into the global namespace.

Note that in the current implementation, the NIS+ clients do not take advantage of the **–p** feature. Also, since the parent object is currently not replicated on root replica servers, it is recommended that this option not be used.

–c Initializes the machine to be a NIS+ client. There are three initialization options available: initialize by coldstart, initialize by hostname, and initialize by broadcast. The most secure mechanism is to initialize from a trusted coldstart file. The second option is to initialize using a hostname that you specify as a trusted host. The third method is to initialize by broadcast and it is the least secure method.

–C *coldstart*
Causes the file *coldstart* to be used as a prototype coldstart file when initializing a NIS+ client. This coldstart file can be copied from a machine that is already a client of the NIS+ namespace. For maximum security, an administrator can encrypt and encode (with **uuencode**(1C)) the coldstart file and mail it to an administrator bringing up a new machine. The new administrator would then decode (with **uudecode**), decrypt, and then use this file with the **nisinit** command to initialize the machine as an NIS+ client. If the coldstart file is from another client in the same domain, the **nisinit** command may be safely skipped and the file copied into the **/var/nis** directory as **/var/nis/NIS_COLD_START**.

–H *hostname*
Specifies that the host *hostname* should be contacted as a trusted NIS+ server. The **nisinit** command will iterate over each transport in the **NETPATH** environment variable and attempt to contact **rpcbind**(1M) on that machine. This hostname *must* be reachable from the client without the name service running. For IP networks this means that there must be an entry in **/etc/hosts** for this host when **nisinit** is invoked.

–B Specifies that the **nisinit** command should use an IP broadcast to locate a NIS+ server on the local subnet. Any machine that is running the NIS+ service may answer. No guarantees are made that the server that answers is a server of the organization’s namespace. If this option is used, it is advisable to check with your system administrator that the server and domain served are valid. The binding information can be dumped to the standard output using the **nisshowcache**(1M) command.

Note that **nisinit –c** will just enable navigation of the NIS+ name space from this client. To make NIS+ your name service, modify the file **/etc/nsswitch.conf** to reflect that. See **nsswitch.conf**(4) for more details.

RETURN VALUES

nisinit returns **0** on success and **1** on failure.

EXAMPLES

This example initializes the machine as an NIS+ client using the host *freddy* as a trusted server.

```
example# nisinit –cH freddy
```

This example sets up a client using a trusted coldstart file.

```
example# nisinit –cC /tmp/colddata
```

This example sets up a client using an IP broadcast.

```
example# nisinit –cB
```

This example sets up a root server.

```
example# nisinit –r
```

ENVIRONMENT

NETPATH This environment variable may be set to the transports to try when contacting the NIS+ server (see **netconfig**(4)). The client library will only attempt to contact the server using connection oriented transports.

FILES

/var/nis/NIS_COLD_START
This file contains a list of servers, their transport addresses, and their Secure RPC public keys that serve the machine's default domain.

/var/nis/data/root.object
This file describes the root object of the NIS+ namespace. It is a standard XDR-encoded NIS+ directory object that can be modified by authorized clients using the **nis_modify**( ) interface.

/var/nis/data/parent.object
This file describes the namespace that is logically above the NIS+ namespace. The most common type of parent object is a DNS object. This object contains contact information for a server of that domain.

/etc/hosts Internet host table.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nis+(1), **uuencode**(1C), **nisclient**(1M), **nisserver**(1M), **nisshowcache**(1M), **sysinfo**(2), **hosts**(4), **netconfig**(4), **nisfiles**(4), **attributes**(5)

NAME

nislog – display the contents of the NIS+ transaction log

SYNOPSIS

/usr/sbin/nislog [**–h** *num* | **–t** *num*] [**–v**] [*directory* ...]

DESCRIPTION

nislog displays the contents of the NIS+ server transaction log on the standard output. This command can be used to track changes in the namespace. The **/var/nis/trans.log** file contains the transaction log maintained by the NIS+ server. When updates occur, they are logged to this file and then propagated to replicas as log transactions. When the log is checkpointed, updates that have been propagated to the replicas are removed.

The **nislog** command can only be run on an NIS+ server by superuser. It displays the log entries for that server only.

If *directory* is not specified, the entire log is searched. Otherwise, only those logs entries that correspond to the specified directories are displayed.

OPTIONS

–h *num* Display *num* transactions from the ''head'' of the log. If the numeric parameter is **0**, only the log header is displayed.

–t *num* Display *num* transactions from the ''tail'' of the log. If the numeric parameter is **0**, only the log header is displayed.

–v Verbose mode.

FILES

/var/nis/trans.log transaction log

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nis+(1), **rpc.nisd**(1M), **nisfiles**(4), **attributes**(5)

NAME

nisping – send ping to NIS+ servers

SYNOPSIS

/usr/lib/nis/nisping [**–uf**] [**–H** *hostname*] [**–r** | *directory*]
/usr/lib/nis/nisping **–C** [**–a**] [**–H** *hostname*] [*directory*]

DESCRIPTION

In the first **SYNOPSIS** line, the **nisping** command sends a ‘‘ping’’ to all replicas of a NIS+ directory. Once a replica receives a ping, it will check with the master server for the directory to get updates. Prior to pinging the replicas, this command attempts to determine the last update "seen" by a replica and the last update logged by the master. If these two timestamps are the same, the ping is not sent. The **–f** (force) option will override this feature.

Under normal circumstances, NIS+ replica servers get the new information from the master NIS+ server within a short time. Therefore, there should not be any need to use **nisping**.

In the second **SYNOPSIS** line, the **nisping** **–C** command sends a checkpoint request to the servers. If no *directory* is specified, the home domain, as returned by **nisdefaults**(1), is checkpointed. If all directories, served by a given server, have to be checkpointed, then use the **–a** option.

On receiving a checkpoint request, the servers would commit all the updates for the given *directory* from the table log files to the database files. This command, if sent to the master server, will also send updates to the replicas if they are out of date. This option is needed because the database log files for NIS+ are not automatically checkpointed. **nisping** should be used at frequent intervals (such as once a day) to checkpoint the NIS+ database log files. This command can be added to the **crontab**(1) file. If the database log files are not checkpointed, their sizes will continue to grow.

OPTIONS

–a Checkpoint all directories on the server.

–C Send a request to checkpoint, rather than a ping, to each server. The servers schedule to commit all the transactions to stable storage.

–H *hostname* Only the host *hostname* is sent the ping, checked for an update time, or checkpointed.

–f Force a ping, even though the timestamps indicate there is no reason to do so. This option is useful for debugging.

–r This option can be used to update or get status about the root object from the root servers, especially when new root replicas are added or deleted from the list.

If used without **–u** option, **–r** will send a ping request to the servers serving the root domain. When the replicas receive a ping, they will update their root object if needed.

The **–r** option can be used with all other options except with the **–C** option; the root object need not be checkpointed.

−u Display the time of the last update; no servers are sent a ping.

RETURN VALUES

−1 No servers were contacted, or the server specified by the **−H** switch could not be contacted.

0 Success.

1 Some, but not all, servers were successfully contacted.

EXAMPLES

This example pings all replicas of the default domain:

```
example% nisping
```

Note that this example will not ping the the **org_dir** and **groups_dir** subdirectories within this domain.

This example pings the server *example* which is a replica of the *org_dir.foo.com.* directory:

```
example% nisping −H example org_dir.foo.com.
```

This example checkpoints all servers of the *org_dir.bar.com.* directory.

```
example% nisping −C org_dir.bar.com.
```

ENVIRONMENT

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

crontab(1), **nisdefaults**(1), **nislog**(1M), **nisfiles**(4), **attributes**(5)

NOTES

If the server specified by the **−H** option does not serve the directory, then no ping is sent.

NAME

nispopulate – populate the NIS+ tables in a NIS+ domain.

SYNOPSIS

/usr/lib/nis/nispopulate –**Y** [–**x**] [–**f**] [–**n**] [–**u**] [–**v**]
[–**S** *0* | *2*] [–**l** <*network_passwd*>] [–**d** <*NIS+_domain*>]
–**h** <*NIS_server_host*> [–**a** <*NIS_server_addr*>]
–**y** <*NIS_domain*> [*table*] **. . .**

/usr/lib/nis/nispopulate –**F** [–**x**] [–**f**] [–**u**] [–**v**] [–**S** 0 | 2]
[–**d** <*NIS+_domain*>] [–**l** <*network_passwd*>]
[–**p** <*directory_path*>] [*table*] **. . .**

/usr/lib/nis/nispopulate –**C** [–**x**] [–**f**] [–**v**] [–**d** <*NIS+_domain*>]
[–**l** <*network_passwd*>] [*hosts* | *passwd*]]

DESCRIPTION

The **nispopulate** shell script can be used to populate NIS+ tables in a specified domain from their corresponding files or NIS maps. **nispopulate** assumes that the tables have been created either through **nisserver**(1M) or **nissetup**(1M).

The table argument accepts standard names that are used in the administration of Solaris systems and non-standard *key-value* type tables. See **nisaddent**(1M) for more information on *key-value* type tables. If the table argument is not specified, **nispopulate** will automatically populate each of the standard tables. These standard (default) tables are: **auto_master**, **auto_home**, **ethers**, **group**, **hosts**, **networks**, **passwd**, **protocols**, **services**, **rpc**, **netmasks**, **bootparams**, **netgroup**, **aliases** and **shadow**. Note that the **shadow** table is only used when populating from files. The non-standard tables that **nispopulate** accepts are those of *key-value* type. These tables must first be created manually with the **nistbladm**(1) command.

Use the first synopsis (–**Y**) to populate NIS+ tables from NIS maps. **nispopulate** uses **ypxfr**(1M) to transfer the NIS maps from the NIS servers to the **/var/yp/**<*NIS_domain*> directory on the local machine. Then, it uses these files as the input source. Note that <*NIS_domain*> is case sensitive. Make sure there is enough disk space for that directory.

Use the second synopsis (–**F**) to populate NIS+ tables from local files. **nispopulate** will use those files that match the table name as input sources in the current working directory or in the specified directory.

Note that when populating the **hosts and passwd** tables, **nispopulate** will automatically create the NIS+ credentials for all users and hosts which are defined in the **hosts** and **passwd** tables, respectively. A network passwd is required to create these credentials. This network password is used to encrypt the secret key for the new users and hosts. This password can be specified using the –**l** option or it will use the default password, "nisplus". **nispopulate** will not overwrite any existing credential entries in the credential table. Use **nisclient**(1M) to overwrite the entries in the **cred** table. It creates both LOCAL and DES credentials for users, and only DES credentials for hosts. To disable automatic credential creation, specify the "**-S** *0*" option.

The third synopsis (**-C**) is used to populate NIS+ credential table with level 2 authentication (**DES**) from the passwd and hosts tables of the specified domain. The valid table arguments for this operation are passwd and hosts. If this argument is not specified then it will use both passwd and hosts as the input source.

If **nispopulate** was earlier used with "**-S** *0*" option, then no credentials were added for the hosts or the users. If later the site decides to add credentials for all users and hosts, then this (**-C**) option can be used to add credentials.

OPTIONS

–a <*NIS_server_addr*> specifies the IP address for the NIS server. This option is ONLY used with the **–Y** option.

–C populate the NIS+ credential table from passwd and hosts tables using **DES** authentication (security level 2).

–d <*NIS+_domain.*> specifies the NIS+ domain. The default is the local domain.

–F populates NIS+ tables from files.

–f forces the script to populate the NIS+ tables without prompting for confirmation.

–h <*NIS_server_host*> specifies the NIS server hostname from where the NIS maps are copied from. This is ONLY used with the **–Y** option. This host must be already exist in either the NIS+ **hosts** table or **/etc/hosts** file. If the hostname is not defined, the script will prompt you for its IP address, or you can use the **–a** option to specify the address manually.

–l <*network_passwd*> specifies the network password for populating the NIS+ credential table. This is ONLY used when you are populating the **hosts** and **passwd** tables. The default passwd is “nisplus”.

–n does not overwrite local NIS maps in **/var/yp/**<*NISdomain*> directory if they already exist. The default is to overwrite the existing NIS maps in the local **/var/yp/**<*NISdomain*> directory. This is ONLY used with the **–Y** option.

–p <*directory_path*> specifies the directory where the files are stored. This is ONLY used with the **–F** option. The default is the current working directory.

-S *0|2* specifies the authentication level for the NIS+ clients. Level 0 is for unauthenticated clients and no credentials will be created for users and hosts in the specified domain. Level 2 is for authenticated (**DES**) clients and DES credentials will be created for users and hosts in the specified domain. The default is to set up with level 2 authentication (**DES**).
There is no need to run nispopulate with **-C** for level 0 authentication.

–u updates the NIS+ tables (ie., adds, deletes, modifies) from either files or NIS maps. This option should be used to bring an NIS+ table up to date when there are only a small number of changes. The default is to add to the NIS+ tables without deleting any existing entries. Also, see the **–n** option for updating NIS+ tables from existing maps in the **/var/yp** directory.

–v runs the script in verbose mode.

-x turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.

–Y populate the NIS+ tables from NIS maps.

–y <*NIS_domain*> specifies the NIS domain to copy the NIS maps from. This is ONLY used with the **–Y** option. The default domainname is the same as the local domainname.

ENVIRONMENT

TMPDIR

nispopulate normally creates temporary files in the directory **/tmp**. You may specify another directory by setting the environment variable **TMPDIR** to your chosen directory. If **TMPDIR** is not a valid directory, then **nispopulate** will use **/tmp**).

EXAMPLES

To populate all the NIS+ standard tables in the domain *xyz.sun.com.* from NIS maps of the *yp.sun.COM* domain as input source where host *yp_host* is a YP server of *yp.sun.COM*:

```
nis_server# /usr/lib/nis/nispopulate -Y -y yp.sun.COM -h yp_host -d xyz.sun.com.
```

To update all of the NIS+ standard tables from the same NIS domain and hosts shown above:

```
  nis_server# /usr/lib/nis/nispopulate -Y -u -y yp.sun.COM –h yp_host \
              –d xyz.sun.com.
```

To populate the **hosts** table in domain *xyz.sun.com.* from the hosts file in the **/var/nis/files** directory and using "somepasswd" as the network password for key encryption:

```
  nis_server# /usr/lib/nis/nispopulate -F -p /var/nis/files -l somepasswd hosts
```

To populate the passwd table in domain xyz.sun.com. from the passwd file in the **/var/nis/files** directory without automatically creating the NIS+ credentials:

```
 nis_server# /usr/lib/nis/nispopulate -F -p /var/nis/files
             -d xys.sun.com. -S 0 passwd
```

To populate the credential table in domain xyz.sun.com. for all users defined in the passwd table.

```
 nis_server# /usr/lib/nis/nispopulate -C –d xys.sun.com. passwd
```

To create and populate a non-standard key-value type NIS+ table, "private", from the file **/var/nis/files/private**: (nispopulate assumes that the private.org_dirkey-value type table has already been created).

```
nis_server# /usr/bin/nistbladm –D access=og=rmcd,nw=r \
            –c private key=S,nogw= value=,nogw= private.org.dir
nis_server# /usr/lib/nis/nispopulate -F -p /var/nis/files private
```

FILES

/etc/hosts	local host name database
/var/yp	NIS(YP) domain directory
/var/nis	NIS+ domain directory
/tmp	

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nis+(1), **nistbladm**(1), **nisaddcred**(1M), **nisaddent**(1M), **nisclient**(1M), **nisserver**(1M), **nissetup**(1M), **rpc.nisd**(1M), **ypxfr**(1M), **attributes**(5)

NAME

nisprefadm – NIS+ utility to set server preferences for NIS+ clients

SYNOPSIS

/usr/bin/nisprefadm –a { **–L** | **–G** } [**–o** *opt-string*] [**–d** *domain*] [**–C** *client*] *server . . .*

/usr/bin/nisprefadm –m { **–L** | **–G** } [**–o** *opt-string*] [**–d** *domain*] [**–C** *client*]
oldserver=newserver . . .

/usr/bin/nisprefadm –r { **–L** | **–G** } [**–o** *opt-string*] [**–d** *domain*] [**–C** *client*] *server . . .*

/usr/bin/nisprefadm –u { **–L** | **–G** } [**–o** *opt-string*] [**–d** *domain*] [**–C** *client*] *server . . .*

/usr/bin/nisprefadm –x { **–L** | **–G** } [**–d** *domain*] [**–C** *client*]

/usr/bin/nisprefadm –l { **–L** | **–G** } [**–d** *domain*] [**–C** *client*]

/usr/bin/nisprefadm –F

DESCRIPTION

nisprefadm defines which servers are to be preferred by NIS+ clients. This information is used by **nis_cachemgr**(1M) to control the order in which it selects which server to use for a particular domain. On a client system, the cache manager first looks for a local preferred server list in **/var/nis**. If it doesn't find one, it looks for an entry with its host name in the NIS+ table. Finally, if it doesn't find it there, it looks for an entry for its subnet.

By default, **nis_cachemgr** puts all servers that are on the same subnet as the client system (that is, local servers) are on the preferred server list. In some cases this default preferred server list is inadequate. For example, if all of the servers for a domain are remote, but some are *closer* than others, the cache manager should try to select the closer one. Because the cache manager has no reliable way to determine the distance to remote servers, **nisprefadm** is used to provide this information.

The preferred server information is stored either globally in a NIS+ table (with the **–G** option) or locally in a file, **/var/nis/client_info** (with the –L option). It is preferable to store the information globally so that it can be used by all clients on a subnet. The **nis_cachemgr** process on a client machine reloads the preferred server information periodically, depending on the machine's setup. If the local file is used, the information is reloaded every 12 hours. If the global table is used, the information is reloaded based on the TTL value of the client information table. This TTL value can be changed using **nischttl**(1). If you want your changes to take effect immediately, use the **nisprefadm –F** command. When changing local information (**–L**), **nisprefadm** automatically forces **nis_cachemgr** to reload the information.

The cache manager assigns weights to all of the servers on the preferred list. By default, local servers (that is, servers on the same subnet) are given a weight of 0. Other servers are given the weight, “infinite”. This can be changed by using the **nisprefadm** command and giving a weight in parentheses after the server name. When selecting a server for a domain, the cache manager first tries to contact the servers with the lowest weight. If it doesn't get a response, it tries the servers with the next lowest weight, and so on. If it fails to get a response from any of the preferred servers, it tries to contact the non-preferred servers.

The use of weights gives fine control over the server selection process, but care must be given to avoid assigning too many different weights. For example, if weights 0, 1, 2, and 3 are used, but all of the servers with weight 0, 1, and 2, are unavailable, then there will be a noticeable delay in selecting a server. This is because the cache manager waits 5 seconds for a response at each weight level before moving on to the next one. As a general rule, one or two weight levels provides a good balance of server selection control and performance.

When specifying a server name, it is not necessary to fully qualify the name. When the cache manager tries to access a domain, it compares the list of servers for the domain with the list of preferred servers. It will find a match if a preferred server name is a prefix of the name of a server for the domain. If a domain is served by two servers with the same prefix, the preferred server name must include enough of the domain name to distinguish the two.

OPTIONS

In the **SYNOPSIS**, when several options are surrounded by braces (that is, by `{´ and `}´) one of the options must be specified.

–a Add the specified servers to the preferred server list.

–C *client* Store the preferred server information with the key, *client*. The *client* can be either a hostname or a subnet number. When a hostname is specified, the preferred server information applies to that host only. When a subnet is specified, the preferred server information applies to all clients on that subnet. The cache manager searches for host specific entries first. It only searches for subnet entries if no host entry is found. If this option is not specified, then the hostname of the machine on which the command is run is used.

–d *domain* Specify the *domain* to which the command is to apply.

–F Tells **nis_cachemgr**(1M) to refresh its preferred server information. The program periodically does this anyway, but this option forces it to do the refresh immediately. When updating the local information, **nis_cachemgr** automatically refreshes the preferred server information.

This option must be executed as **root**.

–l List the current preferred server information.

–L | –G Store the preferred server information locally in the file, **/var/nis/client_info** (the **–L** option), or globally in a NIS+ table **client.info.org-dir.***domain* (the **–G** option). If the information is stored locally, then it only applies to the system on which the command is run. If it is stored globally then it can apply to all systems on a subnet (depending on the value of the **–C** option).

The **–L** option must be run as **root**.

–m Modify the preferred server list. The server specified by *oldserver* is replaced by *newserver*. This is typically used to change the weight for a server.

–o Specify additional options to control server selection. Currently the only valid option is *pref_type*, which can have a value of either **all** (the default) or **pref_only**. If the value is **all**, then the cache manager tries to contact non-preferred servers if all of the preferred servers fail to respond. If **pref_only** is specified, then it won't try non-preferred servers. The only exception to this is when a domain is not served by any of the preferred servers. In this case, the cache manager ignores the option. This is to avoid requiring that preferred servers be defined for every domain.

–r Remove the specified servers from the preferred server list.

–u Clear the list of preferred servers and then add the specified servers to the preferred server list.

–x Remove the preferred server information completely.

RETURN VALUES

nisprefadm returns the following values:

0 On success.

1 On failure.

EXAMPLES

This command sets the preferred server list for the system on which it is run:

```
example% nisprefadm –L –a srv1 srv2
```

The information is stored in a file, **/var/nis/client_info**, so it will only affect this one system.

The following command has the same effect, but the information is stored in a NIS+ table in the default domain.

```
example% nisprefadm –G –a srv1 srv2
```

As a system administrator, you might want to set the preferred server information for a client system other than the one you are running the command on. The following command sets the preferred server information for a client system named *client1*:

```
example% nisprefadm –G –a –C client1 srv1 srv2
```

It is common for all client systems on a subnet to use the same set of preferred servers. The following command sets a preferred server list that applies to all clients on subnet, 192.85.18.0 :

```
example% nisprefadm –G –a –C 192.85.18.0 srv1 srv2
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

nischttl(1), **nis_cachemgr**(1M), **attributes**(5)

NOTES

The **nis_cachemgr**(1M) process automatically adds local servers (same subnet as the client) to the preferred server list with a weight of 0. Thus, it is not necessary to specify them, though it does no harm.

If you specify a weight for a server, you probably should quote the parentheses to avoid having the shell interpret them. The following command illustrates this:

```
example% nisprefadm –G –a –C client1 "srv1(2)"
```

In general, **nis_cachemgr** does a fairly good job of selecting servers on its own. Therefore, the use of **nisprefadm** is not usually necessary. Some situations in which it is recommended are:

No local servers, many remote servers

In this case, **nis_cachemgr** needs to choose one of the remote servers. Because it doesn't have information on which is closest, it sends a ping to all of them and then selects the one that responds fastest. This may not always select the best server. If some of the servers are closer to the client than the others, they should be listed as preferred servers so that **nis_cachemgr** will try them first. This reduces the amount of network traffic for selecting a server.

Very remote servers

In some networks there are NIS+ servers that are only reachable through very slow network connections. It is usually best to avoid unnecessary traffic over that connection. If the *pref_type*=**pref_only** option is set along with preferred servers, then only the preferred servers are contacted for domains they serve. The non-preferred servers are not tried at all; even if all of the preferred servers are unavailable. For domains that are not served by any of the preferred servers, the **pref_only** option is ignored.

NAME

nisrestore – restore NIS+ directory backup

SYNOPSIS

nisrestore [–**fv**] *backup-dir directory* **. . .**
nisrestore [–**fv**] –**a** *backup-dir*
nisrestore –**t** *backup-dir*

DESCRIPTION

nisrestore restores an existing backup of a NIS+ directory object that was created using **nisbackup**(1M). The *backup-dir* is the UNIX directory that contains the NIS+ backup on the server being restored. The **nisrestore** command can be used to restore a NIS+ directory object or a complete NIS+ database. It also can be used as an "out of band" fast replication for a new replica server being initialized. The **rpc.nisd**(1M) daemon must be stopped before running **nisrestore**.

The first synopsis is used to restore a single directory object or a specified list of directory objects. The directory can be partially qualified or fully qualified. The server being restored will be verified against the list of servers serving the directory. If this server is not configured to serve this object, **nisrestore** will exit with an error. The –**f** option will override this check and force the operation.

The second synopsis will restore all of the directory objects contained in the *backup-dir*. Again, the server will be validated against the serving list for each of the directory objects in the *backup-dir*. If one of the objects in the *backup-dir* are not served by this server, **nisrestore** will exit with an error. The –**f** option will override this check and force the operation.

OPTIONS

–**a** Restores all directory objects included in the *backup-dir* partition.

–**f** Forces the restoration of a directory without the validation of the server in the directory object's serving list.

–**t** Lists all directory objects contained in *backup-dir*.

–**v** Verbose option. Additional output will be produced upon execution of the command.

OPERANDS

backup-dir The UNIX directory that contains the data files for the NIS+ directory objects to be restored.

directory The NIS+ directory object(s) to be restored. This can be a fully or partially qualified name.

EXAMPLES

To restore the **org_dir** directory object of the domain **foo.com** on a replica server from a local ufs partition named **/var/backup**:

```
replica_server# nisrestore /var/backup org_dir.foo.com.
```

To force the restore of an entire backed up NIS+ namespace to a replica server from the backup partition named **/var/backup**:

```
replica_server# nisrestore –f –a /var/backup
```

To restore the subdomain **sub.foo.com** on a master server, from a backup that includes other directory objects:

```
master_server# nisrestore /var/backup sub.foo.com. \
org_dir.sub.foo.com. groups_dir.sub.foo.com.
```

EXIT STATUS

0 Successful completion.

1 An error occurred.

FILES

*/backup-dir/***backup_list** This ascii file contains a list of all the objects contained in this *backup-dir* directory. This information can be displayed with the −**t** option.

/backup-dir/directory-object A subdirectory that is created in the *backup-dir* which contains the directory-object backup.

*/backup-dir/directory-object/***data** A subdirectory that contains the data files that are part of the directory-object backup.

*/backup-dir/directory-object/***last.upd** This data file contains timestamp information about the directory-object.

*/backup-dir/directory-object/***data.dict** A NIS+ data dictionary for all of the objects contained in this directory-object backup.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nis+(1), **nisdefaults**(1), **nisbackup**(1M), **nisserver**(1M), **rpc.nisd**(1M), **share_nfs**(1M), **nisfiles**(4), **attributes**(5)

NOTES

The −**a** option will attempt to restore all NIS+ objects contained in the *backup-dir*. If any of these objects are not served by the server, **nisrestore** will exit with an error. If the *backup-dir* contains objects that are not served by the server, **nisrestore** must be executed without the −**a** option and the specific directory objects listed.

The −**f** option will disable verification of the server being configured to serve the objects being restored. This option should be used with care, as data could be inadvertently restored to a server that doesn't serve the restored data. This option is required in the case of restoring a single server domain (master server only) or if the other NIS+ servers are unavailable for NIS+ lookups.

The combination of options −**f** and −**a** should be used with caution, as no validation of the server serving the restored objects will be done.

New replicas can be quickly added to a namespace with the **nisrestore** command. The steps are as follows.

Configure the new replica on the master server (see **nisserver**(1M)):

```
master# nisserver –R –h replica
```

Kill the **rpc.nisd** server process on the new replica server:

```
replica# kill rpc.nisd-pid
```

Create a backup of the NIS+ database on the master, which will include the new replica information (see **nisbackup**(1M)). The **/backup** will need to be exported (see **share_nfs**(1M)) to the new replica:

```
master# nisbackup –a /backup
```

Restore the backup of the NIS+ database on the new replica. Use the –**f** option if **nisrestore** is unable to lookup the NIS+ objects being restored. The backup should be available through nfs or similar means (see **share_nfs**(1M)):

```
replica# nisrestore –f –a /nfs-mnt/backup
```

Restart the **rpc.nisd**(1M) process on the new replica, and the server will immediately be available for service.

NAME

nisserver – set up NIS+ servers.

SYNOPSIS

/usr/lib/nis/nisserver **–r** [**–x**] [**–f**] [**–v**] [**–Y**] [**–d** *NIS+_domain*]
[**–g** *NIS+_groupname*] [**–l** *network_passwd*]

/usr/lib/nis/nisserver **–M** [**–x**] [**–f**] [**–v**] [**–Y**] **–d** *NIS+_domain*
[**–g** *NIS+_groupname*] [**–h** *NIS+_server_host*]

/usr/lib/nis/nisserver **–R** [**–x**] [**–f**] [**–v**] [**–Y**] [**–d** *NIS+_domain*]
[**–h** *NIS+_server_host*]

DESCRIPTION

The **nisserver** shell script can be used to set up a root master, non-root master, and replica NIS+ server with level 2 security (DES).

When setting up a new domain, this script creates the NIS+ directories (including **groups_dir** and **org_dir**) and system table objects for the domain specified. It does not populate the tables. **nispopulate**(1M) must be used to populate the tables.

OPTIONS

–d *NIS+_domain*
Specifies the name for the NIS+ domain. The default is your local domain.

–f
Forces the NIS+ server setup without prompting for confirmation.

–g *NIS+_groupname*
Specifies the NIS+ group name for the new domain. This option is not valid with **–R** option. The default group is admin.<domain>.

–h *NIS+_server_host*
Specifies the hostname for the NIS+ server. It must be a valid host in the local domain. Use a fully qualified hostname (for example, *hostx.xyz.sun.com.*) to specify a host outside of your local domain. This option is *only* used for setting up non-root master or replica servers. The default for non-root master server setup is to use the same list of servers as the parent domain. The default for replica server setup is the local hostname.

–l *network_password*
Specifies the network password with which to create the credentials for the root master server. This option is *only* used for master root server setup (**–r** option). If this option is not specified, the script prompts you for the login password.

–M
Sets up the specified host as a master server. Make sure that **rpc.nisd**(1M) is running on the new master server before this command is executed.

–R
Sets up the specified host as a replica server. Make sure that **rpc.nisd** is running on the new replica server.

–r
Sets up the server as a root master server. Use the **–R** option to set up a root replica server.

–v	Runs the script in verbose mode.
–x	Turns the echo mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.
–Y	Sets up a NIS+ server with NIS-compatibility mode. The default is to set up the server without NIS-compatibility mode.

USAGE

Use the first synopsis of the command (**–r**) to set up a root master server. To run the command, you must be logged in as super-user on the server machine.

Use the second synopsis of the command (**–M**) to set up a non-root master server for the specified domain. To run the command, you must be logged in as a NIS+ principal on a NIS+ machine and have write permission to the parent directory of the domain that you are setting up. The new non-root master server machine must already be an NIS+ client (see **nisclient**(1M)) and have the **rpc.nisd**(1M) daemon running.

Use the third synopsis of the command (**–R**) to set up a replica server for both root and non-root domains. To run the command, you must be logged in as a NIS+ principal on a NIS+ machine and have write permission to the parent directory of the domain that you are replicating. The new non-root replica server machine must already be an NIS+ client and have the **rpc.nisd** daemon running.

EXAMPLES

To set up a root master server for domain **sun.com.** :

```
root_server# /usr/lib/nis/nisserver –r –d sun.com.
```

For the following examples make sure that the new servers are NIS+ clients and **rpc.nisd** is running on these hosts before executing **nisserver**.

To set up a replica server for domain **sun.com.** on host *sunreplica*:

```
root_server# /usr/lib/nis/nisserver –R –d sun.com. –h sunrep
```

To set up a non-root master server for domain *xyz.sun.com.* on host *sunxyz* with the NIS+ groupname as *admin-mgr.xyz.sun.com.* :

```
root_server# /usr/lib/nis/nisserver –M –d xyz.sun.com. –h sunxyz \
        –g admin-mgr.xyz.sun.com.
```

To set up a non-root replica server for domain *xyz.sun.com.* on host *sunabc*:

```
sunxyz# /usr/lib/nis/nisserver –R –d xyz.sun.com. –h sunabc
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

nis+(1), **nisgrpadm**(1), **nismkdir**(1), **nisaddcred**(1M), **nisclient**(1M), **nisinit**(1M), **nispopulate**(1M), **nisprefadm**(1M), **nissetup**(1M), **rpc.nisd**(1M), **attributes**(5)

NAME

nissetup – initialize a NIS+ domain

SYNOPSIS

/usr/lib/nis/nissetup [**–Y**] [*domain*]

DESCRIPTION

nissetup is a shell script that sets up a NIS+ domain to service clients that wish to store system administration information in a domain named *domain*. This domain should already exist prior to executing this command (see **nismkdir**(1) and **nisinit**(1M)).

A NIS+ domain consists of a NIS+ directory and its subdirectories: **org_dir** and **groups_dir**. **org_dir** stores system administration information and **groups_dir** stores information for group access control.

nissetup creates the subdirectories **org_dir** and **groups_dir** in *domain*. Both subdirectories will be replicated on the same servers as the parent domain. After the subdirectories are created, **nissetup** creates the default tables that NIS+ serves. These are **auto_master**, **auto_home**, **bootparams**, **cred**, **ethers**, **group**, **hosts**, **mail_aliases**, **netmasks**, **networks**, **passwd**, **protocols**, **rpc**, **services**, and **timezone**. The **nissetup** script uses the **nistbladm**(1) command to create these tables. The script can be easily customized to add site specific tables that should be created at setup time.

This command is normally executed just once per domain.

OPTIONS

–Y Specify that the domain will be served as both a NIS+ domain as well as an NIS domain using the backward compatibility flag. This will set up the domain to be less secure by making all the system tables readable by unauthenticated clients as well.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nis+(1), **nismkdir**(1), **nistbladm**(1), **nisaddent**(1M), **nisinit**(1M) **nisserver**(1M), **attributes**(5)

NOTES

While this command creates the default tables, it does not initialize them with data. This is accomplished with the **nisaddent**(1M) command.

It is easier to use the **nisserver**(1M) script to create subdirectories and the default tables.

NAME

nisshowcache – NIS+ utility to print out the contents of the shared cache file

SYNOPSIS

/usr/lib/nis/nisshowcache [**–v**]

DESCRIPTION

nisshowcache prints out the contents of the per-machine NIS+ directory cache that is shared by all processes accessing NIS+ on the machine. By default, **nisshowcache** only prints out the directory names in the cache along with the list of active servers. The shared cache is maintained by **nis_cachemgr**(1M).

OPTIONS

–v Verbose mode. Print out the contents of each directory object, including information on the server name and its universal addresses.

FILES

/var/nis/NIS_SHARED_DIRCACHE

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWcsu

SEE ALSO

nis_cachemgr(1M), **syslogd**(1M), **nisfiles**(4), **attributes**(5)

DIAGNOSTICS

Error messages are sent to the **syslogd**(1M) daemon.

NAME

nisstat – report NIS+ server statistics

SYNOPSIS

/usr/lib/nis/nisstat [**–H** *host*] [*directory*]

DESCRIPTION

The **nisstat** command queries a NIS+ server for various statistics about its operations. These statistics may vary between implementations and from release to release. Not all statistics are available from all servers. Requesting a statistic from a server that does not support that statistic is never fatal, it simply returns "unknown statistic".

By default, statistics are fetched from the server(s) of the NIS+ directory for the default domain. If *directory* is specified, servers for that directory are queried.

Supported statistics for this release are as follows:

root server This reports whether the server is a root server.

NIS compat mode This reports whether the server is running in NIS compat mode.

DNS forwarding in NIS mode
This reports whether the server in NIS compat mode will forward host lookup calls to DNS.

security level This reports the security level of this server.

serves directories This lists the directories served by this server.

Operations This statistic returns results in the form:
OP=*opname***:C=***calls***:E=***errors***:T=***micros*
Where *opname* is replaced by the RPC procedure name or operation, *calls* is the number of calls to this procedure that have been made since the server started running. *errors* is the number of errors that have occurred while processing a call, and *micros* is the average time in microseconds to complete the last 16 calls.

Directory Cache This statistic reports the number of calls to the internal directory object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.

Group Cache This statistic reports the number of calls to the internal NIS+ group object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.

Static Storage This statistic reports the number of bytes the server has allocated for its static storage buffers.

Dynamic Storage This statistic reports the amount of heap the server process is currently using.

Uptime This statistic reports the time since the service has been running.

OPTIONS

–H *host* Normally all servers for the directory are queried. With this option, only the machine named *host* is queried. If the named machine does not serve the directory, no statistics are returned.

ENVIRONMENT

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found (see **nisdefaults**(1)).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nisdefaults(1), **attributes**(5)

NAME

nisupdkeys – update the public keys in a NIS+ directory object

SYNOPSIS

/usr/lib/nis/nisupdkeys [**–a** | **–C**] [**–H** *host*] [*directory*]

/usr/lib/nis/nisupdkeys –s [**–a** | **–C**] **–H** *host*

DESCRIPTION

This command updates the public keys in an NIS+ directory object. When the public key for a NIS+ server is changed, the new key must be propagated to all directory objects that reference that server.

nisupdkeys reads a directory object and attempts to get the public key for each server of that directory. These keys are placed in the directory object and the object is then modified to reflect the new keys.

If *directory* is present, the directory object for that directory is updated. Otherwise the directory object for the default domain is updated.

On the other hand, **nisupdkeys -s** gets a list of all the directories served by *host* and updates those directory objects. This assumes that the caller has adequate permission to change all the associated directory objects. The list of directories being served by a given server can also be obtained by **nisstat**(1M).

Before you do this operation, make sure that the new address / public key has been propagated to all replicas.

OPTIONS

–a Update the universal addresses of the NIS+ servers in the directory object. Currently, this only works for the TCP / IP family of transports. This option should be used when the IP address of the server is changed. The server's new address is resolved using **gethostbyname**(3N) on this machine. The **/etc/nsswitch.conf** file must point to the correct source for the *hosts* entry for this resolution to work.

–C Specify to clear rather than set the public key. Communication with a server that has no public key does not require the use of secure RPC.

–H *host* Limit key changes only to the server named *host*. If the hostname is not a fully qualified NIS+ name, then it is assumed to be a host in the default domain. If the named host does not serve the directory, no action is taken.

-s Update all the NIS+ directory objects served by the specified server. This assumes that the caller has adequate access rights to change all the associated directory objects. If the NIS+ principal making this call does not have adequate permissions to update the directory objects, those particular updates will fail and the caller will be notified. If the **rpc.nisd** on *host* cannot return the list of servers it serves, the command will print an error message. The caller would then have to invoke **nisupdkeys** multiple times (as in the first synopsis), once per NIS+ directory that it serves.

EXAMPLES

The following example updates the keys for servers of the *foo.bar.* domain.

```
example% nisupdkeys foo.bar.
```

This example updates the key for host *fred* which serves the *foo.bar.* domain.

```
example% nisupdkeys –H fred foo.bar.
```

This example clears the public key for host *wilma* in the *foo.bar.* directory.

```
example% nisupdkeys –CH wilma foo.bar.
```

This example updates the public key in all directory objects that are served by the host *wilma.*

```
example% nisupdkeys –s –H wilma
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

chkey(1), **niscat**(1), **nisaddcred**(1M), **gethostbyname**(3N), **nis_objects**(3N), **attributes**(5)

NOTES

The user executing this command must have modify access to the directory object for it to succeed. The existing directory object can be displayed with the **niscat**(1) command using the –**o** option.

This command does not update the directory objects stored in the **NIS_COLD_START** file on the NIS+ clients.

If a server is also the root master server, then **nisupdkeys -s** cannot be used to update the root directory.

NAME

nlsadmin – network listener service administration

SYNOPSIS

/usr/sbin/nlsadmin −x
/usr/sbin/nlsadmin [*options*] *net_spec*
/usr/sbin/nlsadmin [*options*] **−N** *port_monitor_tag*
/usr/sbin/nlsadmin −V
/usr/sbin/nlsadmin −c *cmd* | **−o** *streamname* [**−p** *modules*] [**−A** *address* | **−D**]
[**−R** *prognum*: *versnum*]

DESCRIPTION

nlsadmin is the administrative command for the network listener process(es) on a machine. Each network has at least one instance of the network listener process associated with it; each instance (and thus, each network) is configured separately. The listener process ‘‘listens’’ to the network for service requests, accepts requests when they arrive, and invokes servers in response to those service requests. The network listener process may be used with any network (more precisely, with any connection-oriented transport provider) that conforms to the transport provider specification.

nlsadmin can establish a listener process for a given network, configure the specific attributes of that listener, and start and kill the listener process for that network. **nlsadmin** can also report on the listener processes on a machine, either individually (per network) or collectively.

net_spec represents a particular listener process. Specifically, *net_spec* is the relative path name of the entry under **/dev** for a given network (that is, a transport provider). *address* is a transport address on which to listen and is interpreted using a syntax that allows for a variety of address formats. By default, *address* is interpreted as the symbolic ASCII representation of the transport address. An *address* preceded by **\x** will let you enter an address in hexadecimal notation. Note that *address* must appear as a single word to the shell, thus it must be quoted if it contains any blanks.

Changes to the list of services provided by the listener or the addresses of those services are put into effect immediately.

OPTIONS

nlsadmin may be used with the following combinations of options and arguments:

−x Report the status of all of the listener processes installed on this machine.

net_spec Print the status of the listener process for *net_spec* .

−q *net_spec* Query the status of the listener process for the specified network, and reflects the result of that query in its exit code. If a listener process is active, **nlsadmin** will exit with a status of **0**; if no process is active, the exit code will be **1**; the exit code will be greater than **1** in case of error.

−v *net_spec* Print a verbose report on the servers associated with *net_spec*, giving the service code, status, command, and comment for each. It also specifies the **uid** the server will run as and the list of modules to be pushed, if any, before the server is started.

–**z** *service_code net_spec*

Print a report on the server associated with *net_spec* that has service code *service_code,* giving the same information as in the –**v** option.

–**q** –**z** *service_code net_spec*

Query the status of the service with service code *service_code* on network *net_spec,* and exits with a status of **0** if that service is enabled, **1** if that service is disabled, and greater than **1** in case of error.

–**l** *address net_spec*

Change or set the transport address on which the listener listens (the general listener service). This address can be used by remote processes to access the servers available through this listener (see the –**a** option, below).

If *address* is just a dash (–), **nlsadmin** reports the address currently configured, instead of changing it.

A change of address takes effect immediately.

–**t** *address net_spec*

Change or set the address on which the listener listens for requests for terminal service but is otherwise similar to the –**l** option above. A terminal service address should not be defined unless the appropriate remote login software is available; if such software is available, it must be configured as service code 1 (see the –**a** option, below).

–**i** *net_spec*

Initialize an instance of the listener for the network specified by *net_spec;* that is, create and initialize the files required by the listener as well as starting that instance of the listener. Note that a particular instance of the listener should be initialized only once. The listener must be initialized before assigning addresses or services.

–**a** *service_code* [–**p** *modules*] [–**w** *name*] –**c** *cmd* –**y** *comment net_spec*

Add a new service to the list of services available through the indicated listener. *service_code* is the code for the service, *cmd* is the command to be invoked in response to that service code, comprised of the full path name of the server and its arguments, and *comment* is a brief (free-form) description of the service for use in various reports. Note that *cmd* must appear as a single word to the shell; if arguments are required, the *cmd* and its arguments must be enclosed in quotation marks. The *comment* must also appear as a single word to the shell. When a service is added, it is initially enabled (see the –**e** and –**d** options, below).

Service codes are alphanumeric strings, and are administered by AT&T. The numeric service codes 0 through 100 are reserved for internal use by the listener. Service code 0 is assigned to the nlps server, which is the service invoked on the general listening address. In particular, code 1 is assigned to the remote login service, which is the service automatically invoked for connections to the terminal login address.

If the −**p** option is specified, then *modules* will be interpreted as a list of STREAMS modules for the listener to push before starting the service being added. The modules are pushed in the order they are specified. *modules* should be a comma-separated list of modules, with no white space included.

If the −**w** option is specified, then *name* is interpreted as the user name from **/etc/passwd** that the listener should look up. From the user name, the listener obtains the user ID, the group ID(s), and the home directory for use by the server. If −**w** is not specified, the default is to use the user name **listen.**

A service must explicitly be added to the listener for each network on which that service is to be available. This operation will normally be performed only when the service is installed on a machine, or when populating the list of services for a new network.

−**r** *service_code net_spec*
Remove the entry for the *service_code* from that listener’s list of services. This is normally done only in conjunction with the de-installation of a service from a machine.

−**e** *service_code net_spec*
−**d** *service_code net_spec*
Enable or disable (respectively) the service indicated by *service_code* for the specified network. The service must previously have been added to the listener for that network (see the −**a** option, above). Disabling a service will cause subsequent service requests for that service to be denied, but the processes from any prior service requests that are still running will continue unaffected.

−**s** *net_spec*
−**k** *net_spec* Start and kill (respectively) the listener process for the indicated network. These operations are normally performed as part of the system startup and shutdown procedures. Before a listener can be started for a particular network, it must first have been initialized (see the −**i** option, above). When a listener is killed, processes that are still running as a result of prior service requests will continue unaffected.

Under the Service Access Facility, it is possible to have multiple instances of the listener on a single *net_spec.* In any of the above commands, the option −**N** *port_monitor_tag* may be used in place of the *net_spec* argument. This argument specifies the tag by which an instance of the listener is identified by the Service Access Facility. If the −**N** option is not specified (that is, the *net_spec* is specified in the invocation), then it will be assumed that the last component of the *net_spec* represents the tag of the listener for which the operation is destined. In other words, it is assumed that there is at least one listener on a designated *net_spec*, and that its tag is identical to the last component of the *net_spec*. This listener may be thought of as the primary, or default, listener for a particular *net_spec*.

nlsadmin is also used in conjunction with the Service Access Facility commands. In that capacity, the following combinations of options can be used:

–V Write the current version number of the listener's administrative file to the standard output. It is used as part of the **sacadm** command line when **sacadm** adds a port monitor to the system.

–c *cmd* | **–o** *streamname* [**–p** *modules*] [**–A** *address* | **–D**] [**–R** *prognum* : *versnum*]
Format the port monitor-specific information to be used as an argument to **pmadm**(1M)**.**

The **–c** option specifies the full path name of the server and its arguments. *cmd* must appear as a single word to the shell, and its arguments must therefore be surrounded by quotes.

The **–o** option specifies the full path name of a FIFO or named STREAM through which a standing server is actually receiving the connection.

If the **–p** option is specified, then *modules* will be interpreted as a list of STREAMS modules for the listener to push before starting the service being added. The modules are pushed in the order in which they are specified. *modules* must be a comma-separated list, with no white space included.

If the **–A** option is specified, then *address* will be interpreted as the server's private address. The listener will monitor this address on behalf of the service and will dispatch all calls arriving on this address directly to the designated service. This option may not be used in conjunction with the **–D** option.

If the **–D** option is specified, then the service is assigned a private address dynamically, that is, the listener will have the transport provider select the address each time the listener begins listening on behalf of this service. For RPC services, this option will be often be used in conjunction with the **–R** option to register the dynamically assigned address with the rpcbinder. This option may not be used in conjunction with the **–A** option.

When the **–R** option is specified, the service is an RPC service whose address, program number, and version number should be registered with the rpcbinder for this transport provider. This registration is performed each time the listener begins listening on behalf of the service. *prognum* and *versnum* are the program number and version number, respectively, of the RPC service.

nlsadmin may be invoked by any user to generate reports; all operations that affect a listener's status or configuration may only be run by a super-user.

The options specific to the Service Access Facility may not be used together with any other options.

ERRORS

If successful, **nlsadmin** exits with a status of 0. If **nlsadmin** fails for any reason, it exits with a status greater than or equal to 2. See **-q** option for a return status of 1.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWcsu

SEE ALSO

listen(1M), **pmadm**(1M), **rpcbind**(1M), **sacadm**(1M), **attributes**(5)

System Administration Guide

NOTES

Dynamically assigned addresses are not displayed in reports as statically assigned addresses are.

NAME

nscd – name service cache daemon

SYNOPSIS

/usr/sbin/nscd [**–f** *configuration-file*] [**–g**] [**–e** *cachename*, **yes** | **no**] [**–i** *cachename*]

DESCRIPTION

nscd is a process that provides a cache for the most common name service requests. It is started up during multi-user boot. The default *configuration-file* **/etc/nscd.conf** determines the behavior of the cache daemon. See **nscd.conf**(4).

nscd provides cacheing for the **passwd**(4), **group**(4) and **hosts**(4) databases through standard **libc** interfaces, such as **gethostbyname**(3N), **gethostbyaddr**(3N), and others. Each cache has a separate time-to-live for its data; modifying the local database (**/etc/hosts**, and so forth) causes that cache to become invalidated within ten seconds. Note that the shadow file is specifically not cached. **getspnam**(3C) calls remain uncached as a result.

nscd also acts as its own administration tool. If an instance of **nscd** is already running, commands are passed to the running version transparently.

In order to preserve NIS+ security, the startup script for **nscd** (**/etc/init.d/nscd**) checks the permissions on the **passwd**, **group** and **host** tables if NIS+ is being used. If those tables are not readable by unauthenticated users, then caching is disabled so that each process continues to authenticate itself as before.

nscd does not rescan the **/etc/nsswitch.conf** file; if this file is changed, the machine should be rebooted or **nscd** stopped and restarted, as shown in the EXAMPLES below.

OPTIONS

Several of the options described below require a *cachename* specification. Supported values are **passwd**, **group** and **hosts**.

-f *configuration-file*
Causes **nscd** to read its configuration data from the specified file.

-g Prints current configuration and statistics to standard output. This is the only option executable by non-root users.

-e *cachename***, yes** | **no**
Enables or disables the specified cache.

-i *cachename*
Invalidate the specified cache.

EXAMPLES

Stopping and restarting the **nscd** daemon.

```
example# /etc/init.d/nscd stop
example# /etc/init.d/nscd start
```

FILES

/etc/nscd.conf determines behavior of cache daemon

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

gethostbyname(3N), **group**(4), **hosts**(4), **nscd.conf**(4), **nsswitch.conf**(4), **passwd**(4), **attributes**(5)

WARNINGS

The **nscd** interface is included in this release on an uncommitted basis only, and is subject to change or removal in a future minor release.

NAME

nslookup – query name servers interactively

SYNOPSIS

nslookup [– *option*]. . . *host* [*server*]

nslookup [– *option*]. . . – [*server*]

nslookup

DESCRIPTION

nslookup sends queries to Internet domain name servers. It has two modes: interactive and non-interactive. Interactive mode allows the user to contact servers for information about various hosts and domains or to display a list of hosts in a domain. Non-interactive mode is used to display just the name and requested information for a host or domain.

OPTIONS

–option Set the permissible options, as shown in the following list. These are the same options that the **set** command supports in interactive mode (see **set** in the **Commands** section for more complete descriptions).

all	List the current settings
class=*classname*	Restrict search according to the specified class
d2	Set exhaustive debug mode on
nod2	Set exhaustive debug mode off
debug	Set debug mode on
nodebug	Set debug mode off
defname	Set domain-appending mode on
nodefname	Set domain-appending mode off
domain=*string*	Establish the appendable domain
ignoretc	Set it to ignore packet truncation errors
noignoretc	Set it to acknowledge packet truncation errors

OPERANDS

host Inquires about the specified *host*. In this non-interactive command format, **nslookup** Does not prompt for additional commands.

– Causes **nslookup** to prompt for more information, such as host names, before sending one or more queries.

server Directs inquiries to the name server specified here in the command line rather than the one read from the **/etc/resolv.conf** file (see **resolv.conf**(4)). *server* can be either a name or an Internet address. If the specified host cannot be reached, **nslookup** resorts to using the name server specified in **/etc/resolv.conf**.

USAGE

Non-interactive Mode

Non-interactive mode is selected when the name or Internet address of the host to be looked up is given as the first argument.

Within non-interactive mode, space-separated options can be specified. They must be entered before the host name, to be queried. Each option must be prefixed with a hyphen.

For example, to request extensive host information and to set the timeout to 10 seconds when inquiring about **gypsy**, enter:

```
example% nslookup –query=hinfo –timeout=10 gypsy
```

To avoid repeated entry of an option that you almost always use, place a corresponding **set** command in a **.nslookuprc** file located inside your home directory. (See **Commands** for more information about **set**.) The **.nslookuprc** file can contain several **set** commands if each is followed by a RETURN.

Entering and Leaving Interactive Mode

Interactive mode is selected when

- No arguments are supplied.
- A `–´ (hyphen) character is supplied as the *host* argument.

To exit from an interactive **nslookup** session, type **Control-d** or type the command **exit** followed by RETURN.

Supported Command Interactions

The commands associated with interactive mode are subject to various limitations and run-time conventions.

The maximum length of a command line is 255 characters. When the RETURN key is pressed, command-line execution begins. While a command is running, its execution can be interrupted by typing **Control-c.**

The first word entered on the command line must be the name of a **nslookup** command unless you wish to enter the name of a host to inquire about. Any unrecognized command is handled as a host name to inquire about. To force a command to be treated as a host name to be inquired about, precede it with a backslash character.

Commands

exit Exit the **nslookup** program.

help
? Display a brief summary of commands.

host [*server*] Look up information for *host* using the current default server, or using *server* if it is specified.

If the *host* supplied is an Internet address and the query type is **A** or**P**TR, the name of the host is returned. If the *host* supplied is a name and it does not have a trailing period, the default domain name is appended to the name. (This behavior depends on the state of the **set** options **domain**, **srchlist**, **defname**, and **search**).

To look up a host that is not in the current domain, append a period to the name.

finger [*name*] [>> *filename*]

Connect with the finger server on the current host, which is defined by the most recent successful host lookup.

If no *name* value is specified, a list of login account names on the current host is generated.

Similar to a shell command interpreter, output can be redirected to a file using the usual redirection symbols: > and >>.

ls [*–options*] *domain* [>> *filename*]

List the information available for *domain*, optionally creating or appending to *filename*. The default output contains host names and their Internet addresses.

Output can be redirected to *filename* using the > and >> redirection symbols. When output is directed to a file, hash marks are shown for every 50 records received from the server. The permissible values for *options* are:

a Lists aliases of hosts in the domain. This is a synonym for the command **ls –t CNAME**.

d Lists all records for the domain. This is a synonym for the command **ls –t ANY**.

h Lists CPU and operating system information for the domain. This is a synonym for the command **ls –t HINFO**.

s Lists well-known services of hosts in the domain. This is a synonym for the command **ls –t WKS**.

t *querytype-value*
lists all records of the specified type (see **querytype** within the discussion of the **set** command).

set *token***=***value*
set *keyword* Establish a preferred mode of search operation. Permissible *token* and *keyword* values are:

all Display the current values of frequently-used options. Information about the current default server and host is also displayed.

cl[**ass**]**=***classname*
Limit the search according to the protocol group (*classname*) for which lookup information is desired.
Permissible *classname* values are:

ANY	A wildcard selecting all classes
IN	The Internet class (the default)
CHAOS	The Chaos class.
HESIOD	The MIT Athena Hesiod class.

d2
nod2 Enable or disable exhaustive debugging mode. Essentially all fields of every packet are displayed. By default, this option is disabled.

deb[**ug**]
nodeb[**ug**]
Enable or disable debugging mode. When debugging mode is

enabled, much more information is produced about the packet sent to the server and the resulting answer. By default, this option is disabled.

def[**name**]
nodef[**name**]

Enable or disable appending the default domain name to a single-component lookup request (one that lacks a dot). By default, this option is enabled for **nslookup**. The default value for the domain name is the value given in **/etc/resolv.conf**, unless: there is an environmental value for **LOCALDOMAIN** when **nslookup** is run; a recent value has been specified through the **srchlist** command or the **set domain** command.

do[**main**]**=***string*

Change the default domain name to be appended to all lookup requests to *string.* For this option to have any effect, the **defname** option must also be enabled and the **search** option must be set in a compatible way.

The domain search list contains the parents of the default domain if it has at least two components in its name. For example, if the default domain is **CC.Berkeley.EDU**, the search list is **CC.Berkeley.EDU** and **Berkeley.EDU**. Use the **set srchlist** command to specify a different list. Use the **set all** command to display the list.

ignoretc
noignoretc

Ignore packet truncation errors. By default, this option is disabled.

srch[**list**]**=***name1/name2/. . .*

Change the default domain name to *name1* and the domain search list to *name1*, *name2*, etc. A maximum of 6 names can be specified, along with slash characters to separate them. For example,

```
example% set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU
```

sets the domain to **lcs.MIT.EDU** and the search list to all three names. This command overrides the default domain name and search list of the **set domain** command. Use the **set all** command to display the list.

search
nosearch

Enable or disable having the domain names in the domain search list appended to the request, generating a series of lookup queries if necessary until an answer is received. To take effect, the lookup request must contain at least one dot (period); yet it must not contain a trailing period. By default, this option is enabled.

po[**rt**]**=***value* Specify the default TCP / UDP name server port. By default, this value is **53**.

q[**uerytype**]=*value*
ty[**pe**]=*value*

Change the type of information returned from a query to one of:

A	The Internet address of the host
CNAME	The canonical name for an alias
HINFO	The host CPU and operating system type
MD	The mail destination
MX	The mail exchanger
MB	The mailbox domain name
MG	The mail group member
MINFO	The mailbox or mail list information
NS	The name server
PTR	The host name if the query is in the form of an Internet address; otherwise the pointer to other information
SOA	The domain's start-of-authority information
TXT	The text information
UINFO	The user information
WKS	The supported well-known services

(Other types specified in the *RFC 1035* document are valid, but they are not as useful.)

recurse
norecurse Enable or disable having to query other name servers before abandoning a search. By default, this feature is enabled.

ret[**ry**]=*count* Set the maximum number of times to retry a request before abandoning a search. When a reply to a request is not received within a certain amount of time (changed with **set timeout**), the timeout period is doubled and the request is resent. The retry value controls how many times a request is resent before the request is aborted. The default for *count* is **4**.

ro[**ot**]=*host* Change the name of the root server to *host*. This affects the **root** command. The default root server is **ns.internet.net**.

t[**timeout**]=*interval*
Change the amount of time to wait for a reply to *interval* seconds. Each retry doubles the timeout period. The default *interval* is **5** seconds.

vc
novc Enable or disable the use of a virtual circuit when sending requests to the server. By default, this feature is disabled.

root Change the default server to the server for the root of the domain name space. Currently, the host **ns.internic.net** is used; this command is a synonym for **server ns.internic.net**. The name of the root server can be changed with the **set root** command.

server *domain*
lserver *domain* Change the default server to *domain*. **lserver** uses the initial server to

look up information about *domain* while **server** uses the current default server. If an authoritative answer can not be found, the names of servers that might have the answer are returned.

view *filename* Sort the output of previous **ls** command(s) and display it one text screenful at a time, similar to **more**(1).

EXAMPLES

To effectively search the Internet domain namespace, it helps to know its structure. At present, the Internet domain name-space is tree-structured, with one top level domain for each country except the U.S.A. There are also some traditional top level domains, not explicitly tied to any particular country. These include:

COM	Commercial establishments
EDU	Educational institutions
ORG	Not-for-profit organizations
GOV	Government agencies
MIL	MILNET hosts

If you are looking for a specific host, you need to know something about the host's organization in order to determine the top-level domain that it belongs to. For instance, if you want to find the Internet address of a machine at UCLA , do the following:

- Connect with the root server using the **root** command. The root server of the name space has knowledge of the top-level domains.
- Since UCLA is a university, its domain name is **ucla.edu**. Connect with a server for the **ucla.edu** domain with the command **server ucla.edu**. The response produces the names of hosts that act as servers for that domain. Note: the root server does not have information about **ucla.edu**, but knows the names and addresses of hosts that do. Once located by the root server, all future queries will be sent to the UCLA name server.
- To request information about a particular host in the domain (for instance, **locus**), just type the host name. To request a listing of hosts in the UCLA domain, use the **ls** command. The **ls** command requires a domain name (in this case, **ucla.edu**) as an argument.

If you are connected with a name server that handles more than one domain, all lookups for host names must be fully specified with its domain. For instance, the domain **harvard.edu** is served by **seismo.css.gov**, which also services the **css.gov** and **cornell.edu** domains. A lookup request for the host **aiken** in the **harvard.edu** domain must be specified as **aiken.harvard.edu**. However, the

set domain=*name*

and

set defname

commands can be used to automatically append a domain name to each request.

After a successful lookup of a host, use the **finger**(1) command to see who is on the system, or to finger a specific person. (**finger** requires the type to be **A**.)

To get other information about the host, use the

set querytype=*value*

command to change the type of information desired and request another lookup.

ENVIRONMENT

HOSTALIASES	References the file containing host aliases
LOCALDOMAIN	Overrides default domain

EXIT STATUS

The process returns the following values:

0	On success.
1	On failure.

FILES

/etc/resolv.conf	initial domain name and name server addresses
$HOME/.nslookuprc	initial option commands
/usr/lib/nslookup.help	summary of commands

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

finger(1), **more**(1), **in.named**(1M), **nstest**(1M), **resolver**(3N), **resolv.conf**(4), **attributes**(5)

Mockapetris, Paul, *Domain Names - Concepts and Facilities*, RFC 1034, Network Information Center, SRI International, Menlo Park, Calif., November 1987.

Mockapetris, Paul, *Domain Names - Implementation and Specification*, RFC 1035, Network Information Center, SRI International, Menlo Park, Calif., November 1987.

DIAGNOSTICS

If the lookup request is successful, an error message is produced. Possible errors are:

Timed out The server did not respond to a request after a certain amount of time (changed with **set timeout**=*value*) and a certain number of retries (changed with **set retry=***value*).

No response from server
No name server is running on the server machine.

No records The server does not have resource records of the current query type for the host, although the host name is valid. The query type is specified with the **set querytype** command.

Non-existent domain
The host or domain name does not exist.

Connection refused
Network is unreachable
The connection to the name or finger server can not be made at the current time. This error commonly occurs with **ls** and **finger** requests.

Server failure The name server found an internal inconsistency in its database and could not return a valid answer.

Refused The name server refused to service the request.

Format error The name server found that the request packet was not in the proper format. This may indicate an error in *nslookup*.

NAME

nstest – DNS test shell

SYNOPSIS

nstest [**–d**] [**–i**] [**–r**] [**–v**] [**–p** *port*] [*inet_addr* [*logfile*]]

DESCRIPTION

nstest is an interactive DNS test program. Queries are formed and sent by user command; any reply received is printed on the standard output. *inet_addr* is the Internet address of the DNS resolver to which **nstest** should send its queries. If *inet_addr* is not included, **nstest** first tries to contact a DNS server on the local host; if that fails, it tries the servers listed in the **/etc/resolv.conf** file. If a *logfile* is supplied, **nstest** uses it to log the queries sent and replies received.

OPTIONS

–d Causes **nstest** to create a file named **ns_packet.dump** (if it does not exist) and write into it a raw (binary) copy of each packet sent. If **ns_packet.dump** does exist, **nstest** will truncate it.

–i Sets the **RES_IGNTC** flag on the queries it makes. See **resolver**(3N) for a description of the **RES_IGNTC** flag.

–r Turns off the **RES_RECURSE** flag on the queries it makes. See **resolver**(3N) for a description of the **RES_RECURSE** flag.

–v Turns on the **RES_USEVC** and **RES_STAYOPEN** flags on the **res_send ( )** calls made. See **resolver**(3N) for a description of the **RES_USEVC** and **RES_STAYOPEN** flags.

–p Causes **nstest** to use the supplied *port* instead of the default name server port.

USAGE

When **nstest** starts, it prints a prompt (">") and waits for user input. DNS queries are formed by typing a *key letter* followed by the appropriate *argument*. Each *key letter* results in a call to **res_mkquery ( )** with *op* set to either **IQUERY** or **QUERY** and *type* set to one of the type values (defined in **<arpa/nameser.h>**). (Any other **key letter** than those listed below causes **nstest** to print a summary of the following table.)

Key Letter & Argument	**Op**	**Type**
a*host*	**QUERY**	**T_A**
A*addr*	**IQUERY**	**T_A**
B*user*	**QUERY**	**T_MG**
b*user*	**QUERY**	**T_MB**
c*host*	**QUERY**	**T_CNAME**
f*host*	**QUERY**	**T_UINFO**
G*gid*	**IQUERY**	**T_GID**
g*host*	**QUERY**	**T_GID**
h*host*	**QUERY**	**T_HINFO**
i*host*	**QUERY**	**T_MINFO**
M*host*	**QUERY**	**T_MAILB**
m*host*	**QUERY**	**T_MX**

n*host*	**QUERY**	**T_NS**
p*host*	**QUERY**	**T_PTR**
r*host*	**QUERY**	**T_MR**
s*host*	**QUERY**	**T_SOA**
T*host*	**QUERY**	**T_TXT**
U*uid*	**IQUERY**	**T_UID**
u*host*	**QUERY**	**T_UID**
w*host*	**QUERY**	**T_WKS**
x*host*	**QUERY**	**T_AXFR**

After the query is successfully formed, **res_send ( )** is called to send it and wait for a reply. **nstest** then prints the following on the standard output:

- a summary of the request and reply packets, including the **HEADER** structure (defined in **<arpa/nameser.h>**) used in the request
- the question being asked of the name server
- an enumeration of the name server(s) being polled
- a summary of the **HEADER** structure received in the reply
- the question the name server answered
- the answer itself

EXAMPLES

To fetch the address of host **playground.sun.com** from the Sun name server, the user would enter:

```
$ nstest 192.9.5.1
> aplayground.sun.com
```

nstest would return the following:

```
res_mkquery(0, playground.sun.com, 1, 1)
res_send()
HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  rd
        qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
        playground.sun.com, type = A, class = IN

Querying server (# 1) address = 192.9.5.1
got answer:
HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  qr aa rd ra
        qdcount = 1, ancount = 1, nscount = 0, arcount = 0
```

```
QUESTIONS:
        playground.sun.com, type = A, class = IN

ANSWERS:
        playground.sun.com
        type = A, class = IN, ttl = 1 day, dlen = 4
        internet address = 192.9.5.5
```

To look up a PTR record, enter:

```
$ nstest 192.9.5.1
> p5.5.9.192.in-addr.arpa
```

nstest would return the following:

```
res_mkquery(0, 5.5.9.192.in-addr.arpa, 1, 12)
res_send()
HEADER:
        opcode = QUERY, id = 2, rcode = NOERROR
        header flags:  rd
        qdcount = 1, ancount = 0, nscount = 0, arcount = 0

QUESTIONS:
        5.5.9.192.in-addr.arpa, type = PTR, class = IN

Querying server (# 1) address = 192.9.5.1
got answer:
HEADER:
        opcode = QUERY, id = 2, rcode = NOERROR
        header flags:  qr aa rd ra
        qdcount = 1, ancount = 1, nscount = 0, arcount = 0

QUESTIONS:
        5.5.9.192.in-addr.arpa, type = PTR, class = IN

ANSWERS:
        5.5.9.192.in-addr.arpa
        type = PTR, class = IN, ttl = 7 hours 47 mins 2 secs, dlen = 23
        domain name = playground.sun.com
```

FILES

/usr/include/arpa/nameser.h	include file for implementation of DNS protocol
/usr/include/resolv.h	include file for the resolver daemon (in.named)

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	**ATTRIBUTE VALUE**
Availability	SUNWcsu

SEE ALSO

nslookup(1M), **resolver**(3N), **attributes**(5)

NAME

ntpdate – set the date and time via NTP

SYNOPSIS

/usr/sbin/ntpdate [–**bdosu**] [–**a** *key#*] [–**e** *authdelay*] [–**k** *keyfile*] [–**m**]
[–**o** *version*] [–**p** *samples*] [–**t** *timeout*] [–**w**] *server*. . .

DESCRIPTION

The **ntpdate** utility sets the local date and time. To determine the correct time, it polls the Network Time Protocol servers on the hosts given as arguments. This utility must be run as root on the local host. It obtains a number of samples from each of the servers and applies the standard NTP clock filter and selection algorithms to select the best of these. Typically, **ntpdate** can be inserted in the /etc/rc.local startup up script to set the time of day at boot time. It can also be run from time–to–time via **cron**(1M).

The reliability and precision of **ntpdate** improve dramatically with a greater number of servers. While a single server may be used, better performance and greater resistance to inaccuracy on the part of any one server can be obtained by providing at least three or four servers, if not more.

The **ntpdate** utility makes time adjustments in one of two ways. If it determines that your clock is off by more than 0.5 seconds it simply steps the time by calling **gettimeofday**(3C). If the error is less than 0.5 seconds, by default, it slews the clock's time with the offset, via a call to **adjtime**(2). The latter technique is less disruptive and more accurate when the offset is small; it works quite well when **ntpdate** is run by **cron** every hour or two. The adjustment made in the latter case is actually 50% larger than the measured offset. This adjustment tends to keep a badly drifting clock more accurate, at some expense to stability. This tradeoff is usually advantageous. At boot time, however, it is usually better to step the time. This can be forced in all cases by specifying the –**b** switch on the command line.

The **ntpdate** utility will decline to set the date if an NTP server daemon (for example, **xntpd**(1M)) is running on the same host. It can be run on a regular basis from **cron**(1M) as an alternative to running a daemon. Doing so once every hour or two will result in precise enough timekeeping to avoid stepping the clock.

OPTIONS

The following options are supported:

–**s** Log actions via the **syslog**(3) facility rather than to the standard output — a useful option when running the program from **cron**(1M).

–**d** Display what will be done without actually doing it. Information useful for general debugging is also printed.

–**a** *key#* Authenticate transactions, using the key number, *key#*.

–**e** *authdelay* Specify an authentication processing delay, *authdelay* in seconds (see **xntpd**(1M) for details). This number is usually small enough to be negligible for **ntpdate**'s purposes. However, specifying a value may improve timekeeping on very slow CPU's.

–**k** *keyfile* Read keys from the file *keyfile* instead of the default file, **/etc/ntp.keys**. *keyfile* should be in the format described in **xntpd**(1M).

−m Join multicast group specified in *server* and synchronize to multicast NTP packets. The standard NTP group is 224.0.1.1.

−o Force the program to poll as a version 1 or version 2 implementation. By default **ntpdate** claims to be an NTP version 3 implementation in its outgoing packets. However, some older software declines to respond to version 3 queries. This option can be used in these cases.

−p *samples* Set the number of samples *ntpdate* acquires from each server. *samples* can be between 1 and 8 inclusive. The default is 4.

−t *timeout* Set the time **ntpdate** spends, waiting for a response. *timeout* is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

−u Use an unprivileged port to send the packets from. This option is useful when you are behind a firewall that blocks incoming traffic to privileged ports, and you want to synchronize with hosts beyond the firewall. Note that the **−d** option always uses unprivileged ports.

−w When used together with -m, wait until able to join group and synchronize.

FILES

/etc/ntp.keys contains the encryption keys used by **ntpdate**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cron(1M), **xntpd**(1M), **adjtime**(2), **gettimeofday**(3C), **syslog**(3), **attributes**(5)

BUGS

The technique used for improving accuracy by compensating for clock oscillator errors is inadequate. However, doing better would require the program to save *state* from previous runs.

NAME

ntpq – standard Network Time Protocol query program

SYNOPSIS

/usr/sbin/ntpq [–**inp**] [–**c** *command*] [*host*] [ ]

DESCRIPTION

ntpq queries NTP servers which implement the recommended NTP mode 6 control message format, about current state. It can also request changes in that state. The program can be run in interactive mode; or it can be controlled using command line arguments. Requests to read and write arbitrary variables can be assembled, with raw and pretty–printed output options available. By sending multiple queries to the server, **ntpq** can also obtain and print a list of peers in a common format.

If one or more request options are included on the command line, **ntpq** sends each of the requests to NTP servers running on each of the hosts given as command line arguments. By default, **ntpq** sends its requests to **localhost**, if hosts are not included on the command line. If no request options are given, **ntpq** attempts to read commands from the standard input and execute them on the NTP server running on the first host given on the command line. Again, **ntqp** defaults to **localhost** if no other host is specified.

ntpq uses NTP mode 6 packets to communicate with an NTP server. Thus, it can be used to query any compatible server on the network that permits queries. Since NTP is a UDP protocol, this communication will be somewhat unreliable, especially over large distances. **ntpq** makes one attempt to retransmit requests; requests timeout if the remote host is not heard from within a suitable period.

OPTIONS

Command line options are described below. Specifying a command line option other than –**i** or –**n** causes the specified query (queries) to be sent, immediately to the indicated host(s). Otherwise, **ntpq** attempts to read interactive format commands from standard input.

–**c** Interpret the next argument as an interactive format command and add it to the list of commands to be executed on the specified host(s). Multiple –**c** options may be given.

–**i** Operate in interactive mode; write prompts to standard output and read commands from standard input.

–**n** Output all host addresses in dotted–quad numeric format rather than converting them to canonical host names.

–**p** Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the **peers** interactive command (see **USAGE** below).

USAGE

Interactive format commands consist of a keyword followed by up to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed. Normally, the output of a command is sent to standard output; but this output may be written to a file by appending a `>´, followed by a file name, to the command line.

Interactive Commands

A number of interactive format commands are executed entirely within the **ntpq** program itself. They do not result in NTP mode 6 requests being sent to a server. If no request options are included on the command line, and if the standard input is a terminal device, **ntpq** prompts for these commands. The interactive commands are described below:

? [*command_keyword*]
A \`?´ by itself prints a list of all the command keywords known to the current version of **ntpq**. A \`?´ followed by a command keyword prints function and usage information about the command.

timeout *milliseconds*
Specifies a time out period for responses to server queries. The default is about 5000 milliseconds. Since **ntpq** retries each query once after a time out, the total waiting time for a time out is twice the time out value that is set.

delay *milliseconds*
Specifies a time interval to be added to timestamps included in requests which require authentication. This command is used to enable (unreliable) server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. Currently, the server does not require time stamps in authenticated requests. Thus, this command may be obsolete.

host *hostname*
Set the name of the host to which future queries are to be sent. *Hostname* may be either a host name or a numeric address.

keyid *#*
Specify of a key number to be used to authenticate configuration requests. This number must correspond to a key number the server has been configured to use for this purpose.

passwd
Prompts the user to type in a password which will be used to authenticate configuration requests. If an authenticating key has been specified (see **keyid** above), this password must correspond to this key. **ntpq** does not echo the password as it is typed.

hostnames yes | no
If “yes” is specified, host names are printed in
information displays. If “no” is given, numeric addresses are printed instead. The default is “yes” unless modified using the command line **–n** switch.

raw Print all output from query commands exactly as it is received from the remote server. The only formatting / filtering done on the data is to transform nonascii data into printable form.

cooked Causes output from query commands to be “cooked”. The values of variables recognized by the server are reformatted, so that they can be more easily read. Variables which **ntpq** thinks should have a decode-able value, but do not, are marked with a trailing \`?´.

ntpversion [**1 | 2 | 3**]
Sets the NTP version number which **ntpq** claims in packets (defaults is 3). Note that mode 6 control messages (and modes, for that matter) did not exist in NTP version 1. There appear to be no servers left which demand version 1.

authenticate [**yes | no**]
The command **authenticate yes** instructs **ntpq** to send authentication with all requests it makes. Normally **ntpq** does not authenticate requests unless they are write requests. Authenticated requests cause some servers to handle requests slightly differently, and can occasionally cause a slowed response if you turn authentication on before doing a peer display.

addvars *variable_name*[=*value*] [,. . .] **rmvars** *variable_name* [,. . .] **clearvars**
The data carried by NTP mode 6 messages consists of a list of items of the form

variable_name=*value*

where the "=*value*" is ignored, and can be omitted, in requests to the server to read variables. **ntpq** maintains an internal list in which data to be included in control messages can be assembled, and sent. This is accomplished with the **readlist** and **writelist** commands described below. The **addvars** command allows variables and their optional values to be added to the list. If more than one variable is to be added, the list should be comma-separated, and it should not contain white space. The **rmvars** command can be used to remove individual variables from the list; the **clearlist** command removes all variables from the list.

debug [**more | less | off**]
Turns internal query program debugging on and off.

quit Exit **ntpq**.

Control Message Commands

Each peer known to an NTP server has a 16 bit integer *association identifier* assigned to it. NTP control messages which carry peer variables must identify the peer that the values correspond to, by including its association ID. An association ID of 0 is special. It indicates the variables are system variables, whose names are drawn from a separate name space.

Control message commands send one or more NTP mode 6 messages to the server, and cause the data returned to be printed in some format. Most commands currently implemented send a single message and expect a single response. The current exceptions are the **peers mreadlist** and **mreadvar** commands. The **peers** command sends a preprogrammed series of messages to obtain the data it needs. The **mreadlist** and **mreadvar** commands, iterate over a range of associations.

Control message commands are described below:

associations
Obtains and prints a list of association identifiers and peer statuses for in–spec peers of the server being queried. The list is printed in columns. The first of these is an index that numbers the associations from 1, for internal use. The second

column contains the actual association identifier returned by the server and the third the status word for the peer. This is followed by a number of columns containing data decoded from the status word. Note that the data returned by the **associations** command is cached internally in **ntpq**. The index is then of use when dealing with "dumb" servers which use association identifiers that are hard for humans to type. For any subsequent commands which require an association identifier as an argument, the identifier can be specified by using the form, *&index*. Here *index* is taken from the previous list.

lassocations

Obtains and prints a list of association identifiers and peer statuses for all associations for which the server is maintaining *state*. This command differs from the **associations** command only for servers which retain *state* for out–of–spec client associations. Such associations are normally omitted from the display when the **associations** command is used, but are included in the output of **lassociations**.

passociations

Prints association data concerning in–spec peers from the internally cached list of associations. This command performs identically to the **associations** command except that it displays the internally stored data rather than making a new query.

lpassociations

Print data for all associations, including out–of–spec client associations, from the internally cached list of associations. This command differs from **passociations** only when dealing with servers which retain *state* for out-of-spec client associations.

pstatus *assocID*

Sends a read status request to the server for the given association. The names and values of the peer variables returned will be printed. Note that the status word from the header is displayed preceding the variables, both in hexidecimal and in pigeon English.

readvar [*assoc*] [*variable_name*[=*value*] [,. . .]]

Requests that the values of the specified variables be returned by the server by sending a read variables request. If the association ID is omitted or is given as zero the variables are system variables, otherwise they are peer variables and the values returned will be those of the corresponding peer. Omitting the variable list will send a request with no data which should induce the server to return a default display.

rv [*assocID*] [*variable_name*[=*value*] [,. . .]]

An easy–to–type short form for the **readvar** command.

writevar *assocID variable_name*=*value* [,. . .]

Like the **readvar** request, except the specified variables are written instead of read.

readlist [*assocID*]

Requests that the values of the variables in the internal variable list be returned by the server. If the association ID is omitted or is 0 the variables are assumed to be system variables. Otherwise they are treated as peer variables. If the internal variable list is empty a request is sent without data, which should induce the remote server to return a default display.

rl [*assocID*]
An easy–to–type short form of the **readlist** command.

writelist [*assocID*]
Like the **readlist** request, except the internal list variables are written instead of read.

mreadvar *assocID assocID* [*variable_name*[=*value*] [,. . .]]
Like the **readvar** command except the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent **associations** command.

mrv *assocID assocID* [*variable_name*[=*value*] [,. . .]]
An easy–to–type short form of the **mreadvar** command.

mreadlist *assocID assocID*
Like the **readlist** command except the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent **associations** command.

mrl *assocID assocID*
An easy–to–type short form of the **mreadlist** command.

clockvar [*assocID*] [*variable_name*[=*value*] [,. . .]]
Requests that a list of the server's clock variables be sent. Servers which have a radio clock or other external synchronization respond positively to this. If the association identifier is omitted or zero the request is for the variables of the "system clock". This request generally gets a positive response from all servers with a clock. Some servers may treat clocks as pseudo–peers and, hence, can possibly have more than one clock connected at once. For these servers, referencing the appropriate peer association ID shows the variables of a particular clock. Omitting the variable list causes the server to return a default variable display.

cv [*assocID*] [*variable_name*[=*value*] [,. . .]]
An easy–to–type short form of the **clockvar** command.

peers Obtains a list of in–spec peers of the server, along with a summary of each peer's state. Summary information includes:

- The address of the remote peer
- The reference ID (0.0.0.0 if the refID is unknown)
- The stratum of the remote peer
- The type of the peer (local, unicast, multicast or broadcast) when the last packet was received
- The polling interval in seconds
- The reachability register, in octal
- The current estimated delay offset and dispersion of the peer, all in seconds.

The character in the left margin indicates the fate of this peer in the clock selection process. The codes mean:

SPACE	Discarded due to high stratum and/or failed sanity checks.
x	Designated falsticker by the intersection algorithm.
.	Culled from the end of the candidate list.
–	Discarded by the clustering algorithm.
+	Included in the final selection set.
#	Selected for synchronization; but distance exceeds maximum.
*	Selected for synchronization.
o	Selected for synchronization, pps signal in use.

Since the **peers** command depends on the ability to parse the values in the responses it gets, it may fail to work from time to time with servers which poorly control the data formats.

The contents of the host field may be given in one of four forms. It may be a host name, an IP address, a reference clock implementation name with its parameter or, **REFCLK**(*implementation number*,*parameter*). On “hostnames no” only IP–addresses will be displayed.

lpeers Like **peers**, except a summary of all associations for which the server is maintaining state is printed. This can produce a much longer list of peers from inadequate servers.

opeers An old form of the **peers** command with the reference ID replaced by the local interface address.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5)

BUGS

The **peers** command is non–atomic. It may occasionally result in spurious error messages about invalid associations occurring and terminating the command.

The timeout value is a fixed constant. As a result, it often waits a long time to timeout, since the fixed value assumes sort of a worst case. The program should improve the time out estimate as it sends queries to a particular host; but it does not.

NAME

ntptrace – trace a chain of NTP hosts back to their master time source

SYNOPSIS

/usr/sbin/ntptrace [**–vdn**] [**–r** *retries*] [**–t** *timeout*] [*server*]

DESCRIPTION

ntptrace determines where a given Network Time Protocol (NTP) server gets its time from, and follows the chain of NTP servers back to their master time source. If given no arguments, it starts with *localhost*.

Here is an example of the output from **ntptrace**:

```
% ntptrace
localhost: stratum 4, offset 0.0019529, synch distance 0.144135
server2.bozo.com: stratum 2, offset 0.0124263, synch distance 0.115784
usndh.edu: stratum 1, offset 0.0019298, synch distance 0.011993, refid ’WWVB’
```

On each line, the fields are (left to right):

- The server’s host name
- The server’s stratum
- The time offset between that server and the local host (as measured by **ntptrace**; this is why it is not always zero for *localhost*)
- The host’s synchronization distance
- The reference clock ID (only for stratum-1 servers)

All times are given in seconds. (Synchronization distance is a measure of the goodness of the clock’s time.)

OPTIONS

–d Turns on some debugging output.

–n Turns off the printing of host names; instead, host IP addresses are given. This may be necessary if a nameserver is down.

–r *retries* Sets the number of retransmission attempts for each host.

–t *timeout* Sets the retransmission timeout (in seconds); default = 2.

–v Prints verbose information about the NTP servers.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

xntpd(1M), **attributes**(5)

BUGS

This program makes no attempt to improve accuracy by doing multiple samples.

NAME

obpsym – Kernel Symbolic Debugging for OpenBoot Firmware

SYNOPSIS

modload –p misc/obpsym

DESCRIPTION

obpsym is a kernel module that installs OpenBoot callback handlers that provide kernel symbol information to OpenBoot. OpenBoot firmware user interface commands use the callbacks to convert numeric *addresses* to kernel symbol names for display purposes, and to convert kernel symbol names to numeric *literals* allowing symbolic names to be used as input arguments to user interface commands.

Once **obpsym** is installed, kernel symbolic names may be used anywhere at the OpenBoot firmware's user interface command prompt in place of a literal (numeric) string. For example, if **obpsym** is installed, the OpenBoot firmware commands **ctrace** and **dis** typically display symbolic names and offsets in the form *modname:symbolname* + *offset*. User interface Commands such as **dis** can be given a kernel symbolic name such as **ufs:ufs_mount** instead of a numeric address.

Placing the command

```
forceload: misc/obpsym
```

into the **system**(4) file forces the kernel module **misc/obpsym** to be loaded and activates the kernel callbacks during the kernel startup sequence.

obpsym may be useful as a kernel debugger in situations where other kernel debuggers are not useful. For example, on SPARC machines, if **obpsym** is loaded, you may be able to use the OpenBoot firmware's **ctrace** command to display symbolic names in the stack backtrace after a watchdog reset.

Kernel Symbolic Name Syntax

The syntax for a kernel symbolic name is:

[*module-name* **:**] *symbol-name*

Where *module-name* is the name of the kernel module that the symbol *symbol-name* appears in. A **NULL** module name is taken as "all modules, in no particular order" by **obpsym**. The module name **unix** is equivalent to a **NULL** module name, so that conflicts with words defined in the firmware's vocabulary can be avoided.

Typically, OpenBoot firmware reads a word from the input stream and looks the word up in its internal *vocabulary* before checking if the word is a *literal*. Thus, kernel symbols, such as **reset** may be given as **unix:reset** to avoid the unexpected side effect of the firmware finding and executing a matching word in its vocabulary.

FILES

/etc/system system configuration information file
/platform/*platform-name***/kernel/misc/obpsym**

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcar

SEE ALSO

kadb(1M), **kernel**(1M), **modload**(1M), **modunload**(1M), **uname**(1), **system**(4), **attributes**(5)

OpenBoot Command Reference, Version 2.x, 1994, SunSoft Press, ISBN Number 0-13-107194-7

WARNINGS

Some OpenBoot firmware user interface commands may use system resources incompatibly with the way they are used by the Unix kernel. These commands and the use of this feature as a kernel debugger may cause interactions that the Unix kernel is not prepared to deal with. If this occurs, the Unix kernel and / or the OpenBoot firmware user interface commands may react unpredictably and may panic the system, or may hang or may cause other unpredictable results. For these reasons, the use of this feature is only minimally supported and recommended to be used only as a kernel debugger of "last resort".

NOTES

platform-name can be found using the **–i** option of **uname**(1)

obpsym is supported only on architectures that support OpenBoot firmware.

On some systems, OpenBoot must be completely RAM resident so the **obpsym** symbol callback support can be added to the firmware, if the firmware doesn't include support for the symbol callbacks. On these systems, **obpsym** may complain that it requires that "you must use ramforth to use this module".

See the **OpenBoot Command Reference** manual for details on how to use the *ramforth* command, how to place the command into *nvramrc*, and how to set *use-nvramrc?* to *true*. On systems with version 1.x OpenBoot firmware, *nvramrc* doesn't exist, and the *ramforth* command must be typed manually after each reset, in order to use this module.

Once installed, the symbol table callbacks can be disabled by using the following Open-Boot firmware command:

```
0 0 set-symbol-lookup
```

NAME

parse_dynamic_clustertoc – parse clustertoc file based on dynamic entries

SYNOPSIS

<cdrom>/export/exec/sparc.Solaris_2.x/sbin/install.d/parse_dynamic_clustertoc
<cdrom>/export/exec/i386.Solaris_2.x/sbin/install.d/parse_dynamic_clustertoc

DESCRIPTION

This script parses the **clustertoc** file before the **suninstall**(1M) process is run. **parse_dynamic_clustertoc** is called by a modified sysconfig script on the install CD. When **parse_dynamic_clustertoc** runs, it reads the clustertoc and when it encounters **SUNW_CSRMBRIFF** lines, it either checks the platform using the script's builtin function, or calls an external script. The script exits with a 0 if the cluster entry is included, otherwise it will be ignored. If the cluster entry is to be included, the **SUNW_CSRMBRIFF =(<test> <test_arg>)<cluster>** line is converted to **SUNW_CSRMEMBER =<cluster>**.

EXAMPLES

The following is an example of a simple external test to check for a SX Framebuffer. The entry in the **clustertoc** file is shown and following that is the script that must be placed in the **install.d/dynamic_test** directory.

```
SUNW_CSRMBRIFF=(smcc.dctoc sx)SUNWCsx
#! /bin/sh
#
# Likewise, this file is expected to live under $(TESTDIR).
#
case "$1"
in
      sx)    prtconf -p  | grep ´SUNW,sx´ 1> /dev/null;;
esac
```

FILES

<cdrom>/Solaris_2.x/locale/C/.clustertoc.dynamic
dynamic version of the **clustertoc** file

<cdrom>/export/exec/sparc.Solaris_2.x/sbin/install.d/dynamic_test
directory that contains any additional tests

<cdrom>/export/exec/i386.Solaris_2.x/sbin/install.d/dynamic_test
directory that contains any additional tests

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SHWPcdrom (Solaris CD)

SEE ALSO

suninstall(1M), **clustertoc**(4), **attributes**(5)

NAME

passmgmt – password files management

SYNOPSIS

passmgmt –**a** *options name*
passmgmt –**m** *options name*
passmgmt –**d** *name*

DESCRIPTION

The **passmgmt** command updates information in the password files. This command works with both **/etc/passwd** and **/etc/shadow**.

passmgmt –**a** adds an entry for user *name* to the password files. This command does not create any directory for the new user and the new login remains locked (with the string ∗**LK**∗ in the password field) until the **passwd**(1) command is executed to set the password.

passmgmt –**m** modifies the entry for user *name* in the password files. The name field in the **/etc/shadow** entry and all the fields (except the password field) in the **/etc/passwd** entry can be modified by this command. Only fields entered on the command line will be modified.

passmgmt –**d** deletes the entry for user *name* from the password files. It will not remove any files that the user owns on the system; they must be removed manually.

passmgmt can be used only by the super-user.

OPTIONS

–**c** *comment* A short description of the login, enclosed in quotes. It is limited to a maximum of 128 characters and defaults to an empty field.

–**h** *homedir* Home directory of *name*. It is limited to a maximum of 256 characters and defaults to **/usr/***name*.

–**u** *uid* UID of the *name*. This number must range from 0 to the maximum non-negative value for the system. It defaults to the next available UID greater than 99. Without the –**o** option, it enforces the uniqueness of a UID.

–**o** This option allows a UID to be non-unique. It is used only with the –**u** option.

–**g** *gid* GID of *name*. This number must range from 0 to the maximum non-negative value for the system. The default is 1.

–**s** *shell* Login shell for *name*. It should be the full pathname of the program that will be executed when the user logs in. The maximum size of *shell* is 256 characters. The default is for this field to be empty and to be interpreted as **/usr/bin/sh**.

–**l** *logname* This option changes the *name* to *logname*. It is used only with the –**m** option.

The total size of each login entry is limited to a maximum of 511 bytes in each of the password files.

FILES

/etc/passwd
/etc/shadow
/etc/opasswd
/etc/oshadow

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

passwd(1), **useradd**(1M), **userdel**(1M), **usermod**(1M), **passwd**(4), **shadow**(4), **attributes**(5)

EXIT STATUS

The **passmgmt** command exits with one of the following values:

0 Success.

1 Permission denied.

2 Invalid command syntax. Usage message of the **passmgmt** command is displayed.

3 Invalid argument provided to option.

4 UID in use.

5 Inconsistent password files (for example, *name* is in the **/etc/passwd** file and not in the **/etc/shadow** file, or vice versa).

6 Unexpected failure. Password files unchanged.

7 Unexpected failure. Password file(s) missing.

8 Password file(s) busy. Try again later.

9 *name* does not exist (if −**m** or −**d** is specified), already exists (if −**a** is specified), or *logname* already exists (if −**m** −**l** is specified).

NOTES

Do not use a colon (**:**) or RETURN as part of an argument. It is interpreted as a field separator in the password file.

The **passmgmt** command will be removed in a future release. Its functionality has been replaced and enhanced by **useradd**, **userdel**, and **usermod**. These commands are currently available.

This command only modifies password definitions in the local **/etc/passwd** and **/etc/shadow** files. If a network nameservice such as NIS or NIS+ is being used to supplement the local files with additional entries, **passmgmt** cannot change information supplied by the network nameservice.

NAME

patchadd – apply a patch package to a Solaris 2.x system

SYNOPSIS

patchadd [**–d**] [**–u**] [**–B** *backout_dir*]
[**–C** *net_install_image* | **–R** *client_root_path* | **–S** *service*] *patch*

patchadd [**–d**] [**–u**] [**–B** *backout_dir*]
[**–C** *net_install_image* | **–R** *client_root_path* | **–S** *service*]
–M *patch_dir patch_id . . .* | *patch_dir patch_list*

patchadd [**–C** *net_install_image* | **–R** *client_root_path* | **–S** *service*] **–p**

DESCRIPTION

patchadd applies a patch package to a Solaris 2.*x* system. This patch installation utility can not be used to apply Solaris 1.*x* patches. **patchadd** must be run as root.

There are three forms of the **patchadd** command.

The first form of **patchadd** installs *one* patch to a system, client, service, or the mini root of a Net Install Image.

The second form of **patchadd** installs *more than one* patch to a system, client, service, or the mini root of a Net Install Image.

The third form of **patchadd** *displays* installed patches on the client, service, or the mini root of a Net Install Image.

OPTIONS

The following options are supported:

–d Does not back up the files to be patched. *The patch cannot be removed.*

–p Displays a list of the patches currently applied.

–u Installs unconditionally, turns off file validation. Applies the patch even if some of the files to be patched have been modified since their original installation.

–R *client_root_path*
Locates all patch files generated by **patchadd** under the directory *client_root_path*. *client_root_path* is the directory that contains the bootable root of a client from the server's perspective. Specify *client_root_path* as the absolute path name to the beginning of the directory tree under which all patch files generated by **patchadd** are to be located. **–R** cannot be specified with the **–S** option. See **NOTES**.

–B *backout_dir*
Saves backout data to a directory other than the package database. Specify *backout_dir* as an absolute path name.

–S *service* Specifies an alternate service (for example, Solaris_2.3). This service is part of the server and client model, and can only be used from the server's console. Servers can contain shared **/usr** file systems that are created by Host Manager. These service areas can then be made available to the clients they serve. **–S** cannot be specified with the **–R** option. See **NOTES**.

–M *patch_dir patch_id . . .* | *patch_dir patch_list*

Specifies the patches to be installed. Specify patches to the –**M** option in one of the following ways:

1. By directory location *and* patch number.

   To use the directory location and patch number, specify *patch_dir* as the absolute path name of the directory that contains spooled patches. Specify *patch_id* as the patch number of a given patch. Specifying multiple *patch_id*'s is recommended.

2. By directory location *and* the name of a file containing a patch list.

   To use the directory location and a file containing a patch list, specify *patch_dir* as the absolute path name of the directory containing the file with a list of patches to be installed. Specify *patch_list* as the name of the file containing the patches to be installed.

–**C** *net_install_image*

Patches the files located on the mini root on a Net Install Image created by **setup_install_server**. Specify *net_install_image* as the absolute path name to a Solaris 2.6 or later Net Install Image created by **setup_install_server**.

OPERANDS

The following operands are supported:

patch_id The patch number of a given patch. **104945-02** is an example of a *patch_id*.

patch_list The name of a file that contains a list of patches to install. *patch_list* files contain one *patch_id* on each line.

patch The absolute path name to *patch_id*. **/var/sadm/spool/patch/104945-02** is an example of a *patch*.

patch_dir The absolute path name to the directory that contains all the spooled patches. **/var/sadm/spool/patch** is an example of a **patch_dir**.

EXAMPLES

The examples in this section are all relative to the **/usr/sbin** directory.

The following example installs a patch to a standalone machine:

```
example# patchadd /var/spool/patch/104945-02
```

The following example installs a patch to a client from the server's console:

```
example# patchadd –R /export/root/client1 /var/spool/patch/104945-02
```

The following example installs a patch to a service from the server's console:

```
example# patchadd –S Solaris_2.3 /var/spool/patch/104945-02
```

The following example installs multiple patches in a single **patchadd** invocation:

```
example# patchadd –M /var/spool/patch 104945-02 104946-02 102345-02
```

The following example installs multiple patches specifying a file with the list of patches to install:

```
example# patchadd –M /var/spool/patch patchlist
```

The following example installs multiple patches to a client and saves the backout data to a directory other than the default:

```
example# patchadd –M /var/spool/patch –R /export/root/client1
        –B /export/backoutrepository 104945-02 104946-02 102345-02
```

The following example installs a patch to a Solaris 2.6 or later Net Install Image:

```
example# patchadd –C /export/Solaris_2.6/Tools/Boot
        /var/spool/patch/104945-02
```

The following example displays the patches installed on a client:

```
example# patchadd –R /export/root/client1 –p
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>**0** An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWswmt, SUNWcsu

SEE ALSO

cpio(1), **pkginfo**(1), **patchrm**(1M), **pkgadd**(1M), **pkgchk**(1M), **pkgrm**(1M), **showrev**(1M), **attributes**(5)

NOTES

To successfully install a patch to a client or server, **patchadd** must be issued twice, once with the –**R** option and once with the –**S** option. This guarantees that the patch is installed to both the **/usr** and **root** partitions. This is necessary if there are both **/usr** and **root** packages in the patch.

NAME

patchrm – remove a Solaris 2.x patch package and restore previously saved files

SYNOPSIS

patchrm [**–f**] [**–B** *backout_dir*]
[**–C** *net_install_image* | **–R** *client_root_path* | **–S** *service*] *patch_id*

DESCRIPTION

patchrm removes a patch package and restores previously saved files to a Solaris 2.*x* system. **patchrm** cannot be used with Solaris 1.*x* patches. **patchrm** must be run as root.

OPTIONS

The following options are supported:

–f Forces the patch removal regardless of whether the patch was superseded by another patch.

–B *backout_dir*
Removes a patch whose backout data has been saved to a directory other than the package database. This option is only needed if the original backout directory, supplied to the **patchrm** command at installation time, has been moved. Specify *backout_dir* as an absolute path name.

–R *client_root_path*
Locates all patch files generated by **patchrm** under the directory *client_root_path*. *client_root_path* is the directory that contains the bootable root of a client from the server's perspective. Specify *client_root_path* as the absolute path name to the beginning of the directory tree under which all patch files generated from **patchrm** will be located. **–R** cannot be specified with the **–S** option. See **NOTES**.

–S *service*
Specifies an alternate service (for example, Solaris_2.3). This service is part of the server and client model, and can only be used from the server's console. Servers can contain shared **/usr** file systems that are created by Host Manager. These service areas can then be made available to the clients they serve. **–S** cannot be specified with the **–R** option. See **NOTES**.

–C *net_install_image*
Removes the patched files located on the mini root on a Net Install Image created by **setup_install_server**. Specify *net_install_image* as the absolute path name to a Solaris 2.6 or later Net Install Image created by **setup_install_server**.

OPERANDS

The following operands are supported:

patch_id The patch number of a given patch. **104945-02** is an example of a *patch_id*.

EXAMPLES

The examples in this section assume that patch **104945-02** has been installed to the system prior to removal. All of the examples are relative to the **/usr/sbin** directory.

The following example removes a patch from a standalone system:

```
example# patchrm 104945-02
```

The following example removes a patch from a client's system from the server's console:

```
example# patchrm –R /export/root/client1 104945-02
```

The following example removes a patch from a server's service area:

```
example# patchrm –S Solaris_2.3 104945-02
```

The following example removes a patch from a Net Install Image:

```
example# patchrm –C /export/Solaris_2.6/Tools/Boot 104945-02
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>**0** An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWswmt, SUNWcsu

SEE ALSO

cpio(1), **pkginfo**(1), **patchadd**(1M), **pkgadd**(1M), **pkgchk**(1M), **pkgrm**(1M), **showrev**(1M), **attributes**(5)

NAME

pbind – control and query bindings of processes to processors

SYNOPSIS

pbind –b *processor_id pid ...*
pbind –u *pid ...*
pbind [**–q**] [*pid ...*]

DESCRIPTION

pbind controls and queries bindings of processes to processors. **pbind** binds all the LWPs (lightweight processes) of a process to a processor, or removes or displays the bindings.

When an LWP is bound to a processor, it will be executed only by that processor except when the LWP requires a resource that is provided only by another processor. The binding is not exclusive, that is, the processor is free execute other LWPs as well.

Bindings are inherited, so new LWPs and processes created by a bound LWP will have the same binding. Binding an interactive shell to a processor, for example, binds all commands executed by the shell.

Superusers may bind or unbind any process, and other users can use **pbind** to bind or unbind any process for which the user has permission to signal, that is, any process that has the same effective user ID as the user.

OPTIONS

The following options are supported:

–b *processor_id* Binds all the LWPs of the specified processes to the processor *processor_id*. Specify *processor_id* as the processor ID of the processor to be controlled or queried. *processor_id* must be present and on-line. Use the **psrinfo** command to determine whether or not *processor_id* is present and on-line. See **psrinfo**(1M).

–q Displays the bindings of the specified processes, or of all processes. If a process is composed of multiple LWPs, which have different bindings, the bindings of only one of the bound LWPs will be displayed.

–u Removes the bindings of all LWPs of the specified processes, allowing them to be executed on any on-line processor.

OPERANDS

The following operands are supported:

pid The process ID of the process to be controlled or queried.

EXAMPLES

Binding processes

The following example binds processes 204 and 223 to processor 2.

```
example% pbind –b 2 204 223
```

This command displays the following output:

```
process id 204: was 2, now 2
process id 223: was 3, now 2
```

Unbinding a process

The following example unbinds process 204.

```
example% pbind –u 204
```

Querying Bindings

The following example demonstrates that process 1 is bound to processor 0, process 149 has at least one LWP bound to CPU3, and process 101 has no bound LWPs.

```
example% pbind –q 1 149 101
```

This command displays the following output:

```
process id 1: 0
process id 149: 3
process id 101: not bound
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

SEE ALSO

psradm(1M), **psrinfo**(1M), **psrset**(1M), **processor_bind**(2), **processor_info**(2), **sysconf**(3C), **attributes**(5)

DIAGNOSTICS

pbind: cannot query pid 31: No such process

The process specified did not exist or has exited.

pbind: cannot bind pid 31: Not owner

The user does not have permission to bind the process.

pbind: cannot bind pid 31: Invalid argument

The specified processor is not on-line.

NAME

pcmciad – PCMCIA user daemon

SYNOPSIS

/usr/lib/pcmciad

DESCRIPTION

The PCMCIA user daemon provides user-level services for the PCMCIA nexus driver and PCMCIA card client drivers. There are no user-configurable options for this daemon.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcmcu

SEE ALSO

pcmcia(4), **attributes**(5)

DIAGNOSTICS

pcmciad: can't open /dev/pem: No such file or directory

The user daemon could not communicate with the PCMCIA event management driver.

NAME

pfinstall – tests installation profiles

SYNOPSIS

/usr/sbin/install.d/pfinstall −D | **−d** *disk_config* [**−c** *CDpath*] *profile*

DESCRIPTION

After you create a profile, you can use the **pfinstall** command to test the profile and see if it does what you want before using it to install or upgrade a system. **pfinstall** enables you to test a profile against:

- The system's disk configuration where **pfinstall** is being run.
- Other disks by using a disk configuration file that represents a structure of a disk. See **NOTES** on how to create a disk configuration file.

To successfully and accurately test a profile for a particular Solaris release, you must test a profile within the Solaris environment of the same release. For example, if you want to test a profile for Solaris 2.6, you have to run the **pfinstall** command on a system running Solaris 2.6.

So, on a system running Solaris 2.6, you can test Solaris 2.6 initial installation profiles. However, if you want to test a Solaris 2.6 upgrade profile on a system running a previous version of Solaris, or if you don't have a Solaris 2.6 system installed yet to test Solaris 2.6 initial installation profiles, you have to boot a system from a Solaris 2.6 CD image and temporarily create a Solaris 2.6 install environment. Then, you can run **pfinstall** in the Solaris 2.6 install environment to test your profiles.

To create a temporary Solaris 2.6 install environment, boot a system from a Solaris 2.6 CD image (just as you would to install), answer any system identification questions, choose the Solaris Interactive Installation program, and exit out of the first screen that is presented. Then, from the shell, you can execute the **pfinstall** command.

OPTIONS

The following options are supported:

−D **pfinstall** uses the system's disk configuration to test the profile. You must specify either this option or the **−d** option to test the profile (see **WARNINGS**).

−d *disk_config* **pfinstall** uses a disk configuration file, *disk_config*, to test the profile. See **NOTES** on how to create a disk configuration file. You must specify either this option or the **−D** option to test the profile (see **WARNINGS**).

This option cannot be used with an upgrade profile (**install_type upgrade**). You must always test an upgrade profile against a system's disk configuration (**−D** option).

−c *CDpath* The path to the Solaris 2.*x* installation image. This is required if the image is not mounted on **/cdrom**. (For example, use this option if you copied the installation image to disk or mounted the CD-ROM on a directory other than **/cdrom**.)

OPERANDS

The following operand is supported:

profile The file name of the profile to test. If *profile* is not in the directory where **pfinstall** is being run, you must specify the path.

EXAMPLES

The following example tests an upgrade profile, **upgrade.prof**, on a system with a previous version of the Solaris software installed.

1. Boot the system to be upgraded from a Solaris 2.6 image (just as you would to install). The image can be located in the system's local CD-ROM or on an install server.
2. Answer the system configuration questions, if prompted.
3. If you are presented with a choice of installation options, choose the Solaris Interactive Installation program.
4. Exit from the first screen of the Solaris Interactive Installation program.

   After the Solaris Interactive Installation program exits, a shell prompt is displayed.
5. Create a temporary mount point:

   ```
   example# mkdir /tmp/mnt
   ```
6. Mount the directory that contains the profile(s) you want to test.

If you want to:	Then enter:
Mount a remote NFS file system (for systems on the network)	**mount –F nfs** *server_name*:*path* **/tmp/mnt**
Mount a UFS-formatted diskette	**mount –F ufs /dev/diskette /tmp/mnt**
Mount a PCFS-formatted diskette	**mount –F pcfs /dev/diskette /tmp/mnt**

7. Change directory to **/tmp/mnt** where the profile resides:

   ```
   example# cd /tmp/mnt
   ```
8. Test the **upgrade.prof** profile:

   ```
   /usr/sbin/install.d/pfinstall –D upgrade.prof
   ```

The following example tests the **basic.prof** profile against the disk configuration on a Solaris 2.6 system where **pfinstall** is being run. The path to the Solaris CD image is specified because Volume Management is being used.

```
example# /usr/sbin/install.d/pfinstall –D –c /cdrom/cdrom0/s0 basic.prof
```

The following example tests the **basic.prof** profile against the **535_test** disk configuration file. This example uses a Solaris CD image located in the **/export/install** directory, and **pfinstall** is being run on a Solaris 2.6 system.

```
example# /usr/sbin/install.d/pfinstall –d 535_test –c /export/install basic.prof
```

EXIT STATUS

0 Successful (system rebooted).
1 Successful (system not rebooted).
2 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWinst

SEE ALSO

fdisk(1M), **prtvtoc**(1M), **attributes**(5)

Solaris Advanced Installation Guide

WARNINGS

If the **–d** or **–D** option is not specified, **pfinstall** may perform an actual installation on the system by using the specified profile, and the data on the system may be **overwritten**.

NOTES

You have to test a profile on a system with the same platform type for which the profile was created.

SPARC

To create a disk configuration file (**–d** option) for a SPARC based system:

1. Locate a SPARC based system with a disk that you want to test.
2. Create a disk configuration file by redirecting the output of the **prtvtoc**(1M) command to a file.

   ```
   example# prtvtoc /dev/rdsk/c0t3d0s2 > 535_disk
   ```
3. (Optional.) Concatenate disk configuration files into a single file to test a profile against multiple disks. The target numbers in the disk device names must be unique.

   ```
   example# cat 535_disk 1G_disk > mult_disks
   ```

x86

To create a disk configuration file (**–d** option) for an x86 based system:

1. Locate an x86 based system with a disk that you want to test.
2. Create part of the disk configuration file by saving the output of the **fdisk**(1M) command to a file:

   ```
   example# fdisk –R –W 535_disk –h /dev/rdsk/c0t3d0p0
   ```
3. Append the output of the **prtvtoc**(1M) command to the disk configuration file.

   ```
   example# prtvtoc /dev/rdsk/c0t3d0s2 >> 535_disk
   ```

4. (Optional.) Concatenate disk configuration files into a single file to test a profile against multiple disks. The target numbers in the disk device names must be unique.

   ```
   example# cat 535_disk 1G_disk > mult_disks
   ```

To test a profile with a specific system memory size, set **SYS_MEMSIZE** to the specific memory size (in Mbytes) before running **pfinstall**:

```
example# SYS_MEMSIZE=memory_size
example# export SYS_MEMSIZE
```

NAME

ping – send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

/usr/sbin/ping *host* [*timeout*]

/usr/sbin/ping [**–s**] [**–dlLnrRv**] [**–i** *interface*] [**–I** *interval*]
[**–t** *ttl*] *host* [*packetsize*] [*count*]

DESCRIPTION

The utility **ping** utilizes the ICMP protocol's **ECHO_REQUEST** datagram to elicit an ICMP **ECHO_RESPONSE** from the specified *host* or network gateway. If *host* responds, **ping** will print

host **is alive**

on the standard output and exit. Otherwise, after *timeout* seconds, it will write

no answer from *host*

The default value of *timeout* is **20** seconds.

When the **–s** flag is specified, **ping** sends one datagram per second (adjustable with **–I**) and prints one line of output for every **ECHO_RESPONSE** that it receives. No output is produced if there is no response. In this second form, **ping** computes round trip times and packet loss statistics; it displays a summary of this information upon termination or timeout. The default datagram packet size is **64** bytes, or you can specify a size with the *packetsize* command-line argument. If an optional *count* is given, **ping** sends only that number of requests.

When using **ping** for fault isolation, first **ping** the local host to verify that the local network interface is running.

OPTIONS

The following options are supported:

–d Set the **SO_DEBUG** socket option.

–l Loose source route. Use this option in the IP header to send the packet to the given host and back again. Usually specified with the **–R** option.

–L Turn off loopback of multicast packets. Normally, if there are members in the host group on the outgoing interface, a copy of the multicast packets will be delivered to the local machine.

–n Show network addresses as numbers. **ping** normally displays addresses as host names.

–r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to **ping** a local host through an interface that has been dropped by the router daemon (see **in.routed**(1M)).

–R Record route. Sets the IP record route option, which will store the route of the packet inside the IP header. The contents of the record route will only be printed if the **–v** option is given, and only be set on return packets if the target host preserves the record route option across echos, or the **–l** option is given.

–v Verbose output. List any ICMP packets, other than **ECHO_RESPONSE**, that are received.

–i *interface*
Specify the outgoing interface to use for multicast packets. The default interface for multicast packets is determined from the (unicast) routing tables.

–I *interval*
Specify the interval between successive transmissions. The default is one second.

–t *ttl*
Specify the IP time to live for unicast and multicast packets. The default time to live for unicast packets is set with **ndd**(1M) (using the **icmp_def_ttl** variable). The default time to live for multicast is one hop.

OPERANDS

host The network host

EXIT STATUS

The following exit values are returned:

0 Successful operation; the machine is alive.

non-zero An error has occurred; either a malformed argument has been specified, or the machine was not alive.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ifconfig(1M), **in.routed**(1M), **ndd**(1M), **netstat**(1M), **rpcinfo**(1M), **attributes**(5), **icmp**(7P)

NAME

pkgadd – transfer software packages to the system

SYNOPSIS

pkgadd [**–nv**] [**–a** *admin*] [**–d** *device*] [[**–M**] **–R** *root_path*] [**–r** *response*]
[**–V** *fs_file*] [*pkginst* ...]

pkgadd **–s** *spool* [**–d** *device*] [*pkginst* ...]

DESCRIPTION

pkgadd transfers the contents of a software package from the distribution medium or directory to install it onto the system. Used without the **–d** option, **pkgadd** looks in the default spool directory for the package (**/var/spool/pkg**). Used with the **–s** option, it writes the package to a spool directory instead of installing it.

Certain unbundled and third-party packages are no longer entirely compatible with the latest version of **pkgadd**. These packages require user interaction throughout the installation and not just at the very beginning.

To install these older packages (released prior to Solaris 2.4), set the following environment variable:

NONABI_SCRIPTS=TRUE

pkgadd will permit keyboard interaction throughout the installation as long as this environment variable is set.

OPTIONS

–a *admin* Define an installation administration file, *admin*, to be used in place of the default administration file. The token **none** overrides the use of any *admin* file, and thus forces interaction with the user. Unless a full path name is given, **pkgadd** first looks in the current working directory for the administration file. If the specified administration file is not in the current working directory, **pkgadd** looks in the **/var/sadm/install/admin** directory for the administration file.

–d *device* Install or copy a package from *device*. *device* can be a full path name to a directory or the identifiers for tape, floppy disk, or removable disk (for example, **/var/tmp** or **/floppy/***floppy_name*). It can also be a device alias (for example, **/floppy/floppy0**).

–M Instruct **pkgadd** not to use the **$***root_path***/etc/vfstab** file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

–n Installation occurs in non-interactive mode. The default mode is interactive.

–r *response* Identify a file or directory which contains output from a previous **pkgask**(1M) session. This file supplies the interaction responses that would be requested by the package in interactive mode. *response* must be a full pathname.

–R *root_path* Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, **/export/root/client1**).

–s *spool* Write the package into the directory *spool* instead of installing it.

–v Trace all of the scripts that get executed by **pkgadd**, located in the *pkginst*/**install** directory. This option is used for debugging the procedural and non-procedural scripts.

–V *fs_file* Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the **$***root_path***/etc/vfstab** file is non-existent or unreliable.

When executed without options or operands, **pkgadd** uses **/var/spool/pkg** (the default spool directory).

OPERANDS

pkginst The package instance or list of instances to be installed. The token **all** may be used to refer to all packages available on the source medium. The format *pkginst***.*** can be used to indicate all instances of a package.

The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, "*" must be surrounded by single quotes (**'**) or preceded by a backslash (\).

EXAMPLES

The following example installs a package from a Solaris CD-ROM. You are prompted for the name of the package you want to install.

```
example% pkgadd –d /cdrom/cdrom0/s0/Solaris_2.6
```

EXIT STATUS

0 Successful execution.

1 Fatal error.

2 Warning.

3 Interruption.

4 Administration.

10 Reboot after removal of all packages.

20 Reboot after removal of this package.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pkginfo(1), **pkgmk**(1), **pkgparam**(1), **pkgproto**(1), **pkgtrans**(1), **installf**(1M), **pkgask**(1M), **pkgrm**(1M), **removef**(1M), **admin**(4), **attributes**(5)

Application Packaging Developer's Guide

NOTES

When transferring a package to a spool directory, the −**r**, −**n**, and −**a** options cannot be used.

The −**r** option can be used to indicate a directory name as well as a filename. The directory can contain numerous response files, each sharing the name of the package with which it should be associated. This would be used, for example, when adding multiple interactive packages with one invocation of **pkgadd**.

Each package would need a response file. If you create response files with the same name as the package (for example, **pkinst1** and **pkinst2**), then name the directory in which these files reside after the −**r**.

The −**n** option causes the installation to halt if any interaction is needed to complete it.

NAME

pkgask – stores answers to a request script

SYNOPSIS

pkgask [**–d** *device*] [**–R** *root_path*] **–r** *response pkginst* . . .

DESCRIPTION

pkgask allows the administrator to store answers to an interactive package (one with a **request** script, that is, a user-created file that must be named **request**). Invoking this command generates a response file that is then used as input at installation time. The use of this response file prevents any interaction from occurring during installation since the file already contains all of the information the package needs.

OPTIONS

–d *device* Run the request script for a package on *device*. *device* can be a directory pathname or the identifiers for tape, floppy disk or removable disk (for example, **/var/tmp**, **/dev/diskette**, and **/dev/dsk/c1d0s0**). The default device is the installation spool directory.

–R *root_path* Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*.

–r *response* Identify a file or directory which should be created to contain the responses to interaction with the package. The name must be a full pathname. The file, or directory of files, can later be used as input to the **pkgadd**(1M) command.

OPERANDS

pkginst Specify the package instance, or list of instances for which **request** scripts will be created. The token **all** may be used to refer to all packages available on the source medium.

EXIT STATUS

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pkginfo(1), **pkgmk**(1), **pkgparam**(1), **pkgproto**(1), **pkgtrans**(1), **installf**(1M), **pkgadd**(1M), **pkgchk**(1M), **pkgrm**(1M), **removef**(1M), **attributes**(5)

Application Packaging Developer's Guide

NOTES

The **–r** option can be used to indicate a directory name as well as a filename. The directory name is used to create numerous response files, each sharing the name of the package with which it should be associated. This would be used, for example, when you will be adding multiple interactive packages with one invocation of **pkgadd**(1M). Each package would need a response file. To create multiple response files with the same name as the package instance, name the directory in which the files should be created and supply multiple instance names with the **pkgask** command. When installing the packages, you will be able to identify this directory to the **pkgadd**(1M) command.

NAME

pkgchk – check package installation accuracy

SYNOPSIS

pkgchk [**–l** | **–acfnqvx**] [**–i** *file*] [**–p** *path* ...] [**–R** *root_path*]
[[**–m pkgmap** [**–e** *envfile*]] | [*pkginst* ...]]

pkgchk –d *device* [**–l** | **–fv**] [**–i** *file*] [**–M**] [**–p** *path* ...] [**–V** *fs_file*] [*pkginst* ...]

DESCRIPTION

pkgchk checks the accuracy of installed files or, by using the **–l** option, displays information about package files. **pkgchk** checks the integrity of directory structures and files. Discrepancies are written to standard error along with a detailed explanation of the problem.

The first synopsis defined above is used to list or check the contents and/or attributes of objects that are currently installed on the system, or in the indicated **pkgmap**. Package names may be listed on the command line, or by default, the entire contents of a machine will be checked.

The second synopsis is used to list or check the contents of a package which has been spooled on the specified device, but not installed. Note that attributes cannot be checked for spooled packages.

OPTIONS

The following options are supported:

–a Audit the file attributes only and do not check file contents. Default is to check both.

–c Audit the file contents only and do not check file attributes. Default is to check both.

–d *device* Specify the device on which a spooled package resides. *device* can be a directory path name or the identifiers for tape, floppy disk, or removable disk (for example, **/var/tmp** or **/dev/diskette**).

–e *envfile* Request that the package information file named as *envfile* be used to resolve parameters noted in the specified **pkgmap** file.

–f Correct file attributes if possible. If used with the **–x** option, this option removes hidden files. When **pkgchk** is invoked with this option, it creates directories, named pipes, links, and special devices if they do not already exist. If the **–d** option calls out an uninstalled package, the **–f** option will only take effect if the package is in directory (not stream) format. All file attributes will be set to agree with the entries in the **pkgmap** file except that setuid, setgid, and sticky bits will not be set in the mode.

–i *file* Read a list of path names from *file* and compare this list against the installation software database or the indicated **pkgmap** file. Path names which are not contained in *file* are not checked.

−l List information on the selected files that make up a package. This option is not compatible with the **−a**, **−c**, **−f**, **−g**, and **−v** options.

−m pkgmap Check the package against the package map file, **pkgmap**.

−M Instruct **pkgchk** not to use the **$***root_path***/etc/vfstab** file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

−n Do not check volatile or editable files. This should be used for most post-installation checking.

−p *path* Only check the accuracy of the path name or path names listed. *path* can be one or more path names separated by commas (or by white space, if the list is quoted).

−q Quiet mode. Do not give messages about missing files.

−R *root_path* Define the full name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, **/export/root/client1**).

−v Verbose mode. Files are listed as processed.

−V *fs_file* Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the **$***root_path***/etc/vfstab** file is non-existent or unreliable.

−x Search exclusive directories, looking for files which exist that are not in the installation software database or the indicated **pkgmap** file.

OPERANDS

pkginst The package instance or instances to be checked. The format *pkginst***.**∗ can be used to check all instances of a package. The default is to display all information about all installed packages.

The asterisk character (∗) is a special character to some shells and may need to be escaped. In the C-Shell, "∗" must be surrounded by single quotes (**'**) or preceded by a backslash (\).

EXAMPLES

The following example displays information for **/usr/bin/ls**:

```
example% pkgchk −l −p /usr/bin/ls
```

EXIT STATUS

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pkginfo(1), **pkgtrans**(1), **pkgadd**(1M), **pkgask**(1M), **pkgrm**(1M), **attributes**(5)

Application Packaging Developer's Guide

NAME

pkgrm – remove a package from the system

SYNOPSIS

pkgrm [**–nv**] [**–a** *admin*] [[**–A** | **–M**] **–R** *root_path*] [**–V** *fs_file*]

pkgrm **–s** *spool* [*pkginst* ...]

DESCRIPTION

pkgrm will remove a previously installed or partially installed package from the system. A check is made to determine if any other packages depend on the one being removed. If a dependency exists, the action taken is defined in the *admin* file.

The default state for the command is in interactive mode, meaning that prompt messages are given during processing to allow the administrator to confirm the actions being taken. Non-interactive mode can be requested with the **–n** option.

The **–s** option can be used to specify the directory from which spooled packages should be removed.

Certain unbundled and third-party packages are no longer entirely compatible with the latest version of **pkgrm**. These packages require user interaction throughout the removal and not just at the very beginning.

To remove these older packages (released prior to Solaris 2.4), set the following environment variable:

```
NONABI_SCRIPTS=TRUE
```

pkgrm will permit keyboard interaction throughout the removal as long as this environment variable is set.

OPTIONS

–a *admin* Use the installation administration file, *admin*, in place of the default *admin* file. **pkgrm** first looks in the current working directory for the administration file. If the specified administration file is not in the current working directory, **pkgrm** looks in the **/var/sadm/install/admin** directory for the administration file.

–A Remove the package files from the client's file system, absolutely. If a file is shared with other packages, the default behavior is to not remove the file from the client's file system.

–M Instruct **pkgrm** not to use the **$***root_path***/etc/vfstab** file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

–n Non-interactive mode. If there is a need for interaction, the command will exit. Use of this option requires that at least one package instance be named upon invocation of the command.

–R *root_path* Defines the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*.

−s *spool* Remove the specified package(s) from the directory *spool*. The default directory for spooled packages is **/var/sadm/spool/pkg**.

−v Trace all of the scripts that get executed by **pkgrm**, located in the *pkginst*/**install** directory. This option is used for debugging the procedural and non-procedural scripts.

−V *fs_file* Specify an alternative *fs_file* to map the client's file systems. Used in situations where the **$***root_path***/etc/vfstab** file is non-existent or unreliable.

OPERANDS

pkginst Specifies the package to be removed. The format *pkginst***.*** can be used to remove all instances of a package.

The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, "*" must be surrounded by single quotes (') or preceded by a backslash (\).

EXAMPLES

The following example removes all instances of **SUNWjunk** from **client1**:

```
example% pkgrm −R /export/root/client1 SUNWjunk*
```

EXIT STATUS

0 Successful execution.

1 Fatal error.

2 Warning.

3 Interruption.

4 Administration.

10 Reboot after removal of all packages.

20 Reboot after removal of this package.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pkginfo(1), **pkgmk**(1), **pkgparam**(1), **pkgproto**(1), **pkgtrans**(1), **installf**(1M), **pkgadd**(1M), **pkgask**(1M), **pkgchk**(1M), **removef**(1M), **attributes**(5)

Application Packaging Developer's Guide

NAME

pmadm – port monitor administration

SYNOPSIS

pmadm –**a** [–**p** *pmtag* | –**t** *type*] –**s** *svctag* –**i** *id* –**m** *pmspecific* –**v** *ver* [–**f xu**]
    [–**y** *comment*] [–**z** *script*]
pmadm –**r** –**p** *pmtag* –**s** *svctag*
pmadm –**e** –**p** *pmtag* –**s** *svctag*
pmadm –**d** –**p** *pmtag* –**s** *svctag*
pmadm –**l** [–**t** *type* | –**p** *pmtag*] [–**s** *svctag*]
pmadm –L [–**t** *type* | –**p** *pmtag*] [–**s** *svctag*]
pmadm –**g** –**p** *pmtag* –**s** *svctag* [–**z** *script*]
pmadm –**g** –**s** *svctag* –**t** *type* –**z** *script*

DESCRIPTION

pmadm is the administrative command for the lower level of the Service Access Facility hierarchy, that is, for service administration. A port may have only one service associated with it although the same service may be available through more than one port. In order to uniquely identify an instance of a service, the **pmadm** command must identify both the port monitor or port monitors through which the service is available (–**p** or –**t**) and the service (–**s**). See the option descriptions below.

pmadm performs the following functions:

- add or remove a service
- enable or disable a service
- install or replace a per-service configuration script
- print requested service information

Any user on the system may invoke **pmadm** to request service status (–**l** or –**L**) or to print per-service configuration scripts (–**g** without the –**z** option). **pmadm** with other options may be executed only by a privileged user.

OPTIONS

–**a** Add a service. **pmadm** adds an entry for the new service to the port monitor's administrative file. Because of the complexity of the options and arguments that follow the –**a** option, it may be convenient to use a command script or the menu system to add services.

–**d** Disable a service. Add **x** to the flag field in the entry for the service *svctag* in the port monitor's administrative file. This is the entry used by port monitor *pmtag*. See the –**f** option, below, for a description of the flags available.

–**e** Enable a service. Remove **x** from the flag field in the entry for the service *svctag* in the port monitor administrative file. This is the entry used by port monitor *pmtag*. See the –**f** option, below, for a description of the flags available.

–f xu The **–f** option specifies one or both of the following two flags which are then included in the flag field of the entry for the new service in the port monitor's administrative file. If the **–f** option is not included, no flags are set and the default conditions prevail. By default, a new service is enabled and no **utmp** entry is created for it. A **–f** option without a following argument is illegal.

x Do not enable the service *svctag* available through port monitor *pmtag*.

u Create a **utmp** entry for service *svctag* available through port monitor *pmtag*.

–g Print, install, or replace a per-service configuration script. The **–g** option with a **–p** option and a **–s** option prints the per-service configuration script for service *svctag* available through port monitor *pmtag*. The **–g** option with a **–p** option, a **–s** option, and a **–z** option installs the per-service configuration script contained in the file *script* as the per-service configuration script for service *svctag* available through port monitor *pmtag*. The **–g** option with a **–s** option, a **–t** option, and a **–z** option installs the file *script* as the per-service configuration script for service *svctag* available through any port monitor of type *type*. Other combinations of options with **–g** are invalid.

–i *id* *id* is the identity that is to be assigned to service *svctag* when it is started. *id* must be an entry in **/etc/passwd**.

–l The **–l** option requests service information. Used by itself and with the options described below it provides a filter for extracting information in several different groupings.

–l By itself, the **–l** option lists all services on the system.

–l –p *pmtag* Lists all services available through port monitor *pmtag*.

–l –s *svctag* Lists all services with tag *svctag*.

–l –p *pmtag*–**s***svctag*
Lists service *svctag*.

–l –t *type* Lists all services available through port monitors of type *type*.

–l –t *type*–**s***svctag*
Lists all services with tag *svctag* available through a port monitor of type *type*.

Other combinations of options with **–l** are invalid.

–L The **–L** option is identical to the **–l** option except that output is printed in a condensed format.

–m *pmspecific*
pmspecific is the port monitor-specific portion of the port monitor administrative file entry for the service.

–p *pmtag* Specifies the tag associated with the port monitor through which a service (specified as **–s** *svctag*) is available.

–r Remove a service. When **pmadm** removes a service, the entry for the service is removed from the port monitor's administrative file.

–s *svctag* Specifies the service tag associated with a given service. The service tag is assigned by the system administrator and is part of the entry for the service in the port monitor's administrative file.

–t *type* Specifies the the port monitor type.

–v *ver* Specifies the version number of the port monitor administrative file. The version number may be given as

```
–v ´pmspec –V`
```

where *pmspec* is the special administrative command for port monitor *pmtag*. This special command is **ttyadm** for **ttymon** and **nlsadmin** for **listen**. The version stamp of the port monitor is known by the command and is returned when *pmspec* is invoked with a **–V** option.

–y *comment*
Associate *comment* with the service entry in the port monitor administrative file.

–z *script* Used with the **–g** option to specify the name of the file that contains the per-service configuration script. Modifying a configuration script is a three-step procedure. First a copy of the existing script is made (**–g** alone). Then the copy is edited. Finally, the copy is put in place over the existing script (**–g** with **–z**).

Options that request information write the requested information to the standard output. A request for information using the **–l** option prints column headers and aligns the information under the appropriate headings. In this format, a missing field is indicated by a hyphen. A request for information in the condensed format using the **–L** option prints the information in colon-separated fields; missing fields are indicated by two successive colons. **#** is the comment character.

EXAMPLES

Add a service to a port monitor with tag **pmtag**. Give the service the tag **svctag**. Port monitor-specific information is generated by **specpm**. The service defined by **svctag** will be invoked with identity **root**.

```
pmadm –a –p pmtag –s svctag –i root –m `specpm –a arg1 –b arg2`
    –v `specpm –V`
```

Add a service with service tag **svctag**, identity **guest**, and port monitor-specific information generated by **specpm** to all port monitors of type **type**:

```
pmadm –a –s svctag –i guest –t type –m `specpm –a arg1 –b arg2`
    –v `specpm –V`
```

Remove the service **svctag** from port monitor **pmtag**:

```
pmadm –r –p pmtag –s svctag
```

Enable the service **svctag** available through port monitor **pmtag**:

```
pmadm –e –p pmtag –s svctag
```

Disable the service **svctag** available through port monitor **pmtag**:

```
pmadm –d –p pmtag –s svctag
```

List status information for all services:

```
pmadm –l
```

List status information for all services available through the port monitor with tag **ports**:

```
pmadm –l –p ports
```

List the same information in condensed format:

```
pmadm –L –p ports
```

List status information for all services available through port monitors of type **listen**:

```
pmadm –l –t listen
```

Print the per-service configuration script associated with the service **svctag** available through port monitor **pmtag**:

```
pmadm –g –p pmtag –s svctag
```

EXIT CODES

If successful, **pmadm** will exit with a status of **0**. If it fails for any reason, it will exit with a non-zero status.

FILES

/etc/saf/*pmtag***/_config**
/etc/saf/*pmtag/svctag*
/var/saf/*pmtag***/***

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

sac(1M), **sacadm**(1M), **doconfig**(3N), **attributes**(5)

NAME

pmconfig – configure the power management system

SYNOPSIS

/usr/sbin/pmconfig [**–r**]

DESCRIPTION

The **pmconfig** utility reads the configuration file **power.conf**(4) and issues the commands that activate the power management configuration. The **pmconfig** utility is run at system boot. This utility can also be run from the command line after manual changes have been made to the **power.conf**(4) file. Users must run **pmconfig** after editing **power.conf** to have (editing) changes to the **power.conf**(4) file take effect.

OPTIONS

The following options are supported:

–r Reset all power managed devices to unconfigured.

FILES

/etc/power.conf system power management configuration file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpmu

SEE ALSO

powerd(1M), **power.conf**(4), **attributes**(5), **pm**(7D)

DIAGNOSTICS

If the program cannot open either the pseudo driver or the configuration file, it prints an error message to standard error. If the program encounters a syntax error in the configuration file, it prints an error message and the line number of the error in the configuration file. It then skips the rest of the information on that line and processes the next line. Any configuration information already processed on the line containing the error is used.

The **pmconfig** utility generates the following error messages, all of which begin with **pmconfig (line** *n***):**

Can't find device name:
The first field is not a device name.

Can't find threshold value:
The field following the device name is not an integer.

Too many threshold values:
More idle times than the device supports were given.

Unrecognizable dependent name:
The dependent field is not a device name.

a standard error message
Returned from the **pm** driver.

NAME

pntadm – DHCP network table management utility

SYNOPSIS

pntadm –**C** [–**r** *resource*] [–**p** *path*] *network*

pntadm –**A** *name_IP_address* [–**c** *comment*] [–**e** *mm/dd/yyyy*]
[–**f** *num* | *keywords*] [–**h** *host name*] [–**i** [–**a**] *client ID*]
[–**m** [–**y**] *dhcptab_macro*] [–**s** *server*] [–**r** *resource*] [–**p** *path*] *network*

pntadm –**M** *name_IP_address* [–**c** *comment*] [–**e** *mm/dd/yyyy*] [–**f** *num* | *keywords*]
[–**h** *host name*] [–**i** [–**a**] *client ID*] [–**m** [–**y**] *dhcptab_macro*] [–**s** *server*]
[–**r** *resource*] [–**p** *path*] *network*

pntadm –**D** *name_IP_address* [–**y**] [–**r** *resource*] [–**p** *path*] *network*

pntadm –**P** [–**v**] [–**r** *resource*] [–**p** *path*] *network*

pntadm –**R** [–**r** *resource*] [–**p** *path*] *network*

DESCRIPTION

The **pntadm** command manages the **dhcp network** DHCP client tables. One of the following option flags must be specified: –**C**, –**A**, –**M**, –**D**, or –**R**.

Note also that if the networks you wish to add are subnetted, you will need to update the **netmasks**(4) table. Depending on the resource type (–**r** option), you must have the proper file permissions or NIS+ credentials.

For a description of the format of **dhcp network** tables, see **dhcp_network**(4).

OPTIONS

–**A** *name_IP_address*

Add a client entry with hostname or client IP address, *name_IP_address*, to the named **dhcp network** table. Optional sub-options, with defaults, are:

Option(s)	Argument	Description	Default
–**c**	*comment*	Comment text	**NULL**
–**e**	*mm/dd/yyyy*	Absolute lease	**0**
–**f**	*num* | *keywords*	Flag value	**00**
–**h**	*host name*	Client hostname	**NULL**
–**i**	*client ID*	Client identifier[–**a**]	**00**
–**m**	**dhcptab** *macro*[–**y**]	Macro name	**UNKNOWN**
–**s**	*server*	Server IP or name	nodename

The flag (–**f**) option can be specified either as a single number denoting the intended flag value, or a series of the following keywords, combined using the plus (**+**) symbol:

Keyword	Numeric	Description
DYNAMIC	**00**	Server manages assignment
PERMANENT	**01**	Lease on entry is permanent
MANUAL	**02**	Administrator managed assignment
UNUSABLE	**04**	Entry is not valid
BOOTP	**08**	Entry reserved for BOOTP clients

For a more detailed description of the flag values, see **dhcp_network**(4).

The –**i** option modified with –**a** specifies that the client identifier is in ASCII format, and thus needs to be converted to hexadecimal format before insertion into the table.

The –**m** option modified with –**y** verifies the existence of the named macro in the **dhcptab** table before adding the entry.

–**C** Create the DHCP network table for the network specified by *network* (see **OPERANDS**). For details, see **dhcp_network**(4) and **networks**(4).

–**D** *name_IP_address*

Delete the specified client entry with hostname or client IP address, *name_IP_address*, in the named **dhcp network** table. (See **dhcp_network**(4).) Optional sub-options are:

Option	Description
–**y**	Remove associated host table entry

–**M** *name_IP_address*

Modify the specified client entry with hostname or client IP address, *name_IP_address*, in the named **dhcp network** table. (See **dhcp_network**(4).) Optional sub-options are:

Option(s)	Argument	Description	Default
–**c**	*comment*	Comment text	**NULL**
–**e**	*mm/dd/yyyy*	Absolute lease	**0**
–**f**	*num* | *keywords*	Flag value	**00**
–**h**	*host name*	Client hostname	**NULL**
–**i**	*client ID*	Client identifier[–**a**]	**00**
–**m**	**dhcptab** *macro*[–**y**]	Macro name	**UNKNOWN**
–**n**	*client IP*	New IP address	**NULL**
–**s**	*server*	Server IP or name	nodename

For more detailed description of the sub-options and flag values, see the information given under –**A** option above and **dhcp_network**(4).

–P Display the named **dhcp network** table. See **dhcp_network**(4). Optional sub-options are:

Options	Description
–v	Display lease time in verbose format

–p *path* Override the **/etc/default/dhcp** configuration value for resource path, *path*. The resource path for the **files** resource is an absolute UNIX pathname and a fully specified **nisplus** directory (including the tailing period) for the NIS+ resource. See **dhcp**(4) for more details.

–R Remove the named **dhcp network** table. See **dhcp_network**(4).

–r *resource* Override the **/etc/default/dhcp** configuration value for resource type, *resource*. Currently supported resource types are **files** or **nisplus**. See **dhcp**(4) for more details.

OPERANDS

network The network address or network name which corresponds to the **dhcp network** table. See **dhcp_network**(4).

EXAMPLES

1) The following command creates a table for the **10.0.0.0** (subnetted to class C) **dhcp network** table. Note that if you have an alias for this network in your **networks**(4) table, you can use that value rather than the dotted Internet Address notation.

```
# pntadm –C 10.0.0.0
```

2) The following command adds an entry to the **10.0.0.0** table in the **files** resource in the **/var/mydhcp** directory:

```
# pntadm –r files –p /var/mydhcp –A 10.0.0.1 10.0.0.0
```

3) The following command modifies the **10.0.0.1** entry of the **10.0.0.0** table, changing the macro name to **Green**, setting the flags field to **MANUAL** and **PERMANENT**:

```
# pntadm –M 10.0.0.1 –m Green –f ’PERMANENT + MANUAL’ 10.0.0.0
```

4) The following command changes the **10.0.0.1** entry to **10.0.0.2**, making an entry in the **hosts**(4) table called **myclient**:

```
# pntadm –M 10.0.0.1 –n 10.0.0.2 –h myclient 10.0.0.0
```

5) The following command sets the client ID as ASCII **aruba.foo.com** for the **myclient** entry:

```
# pntadm –M myclient –i ’aruba.foo.com’ –a 10.0.0.0
```

6) The following command deletes the **myclient** (**10.0.0.2**) entry from the **10.0.0.0** table:

```
# pntadm –D myclient 10.0.0.0
```

7) The following command removes the named **dhcp network** table in the **nisplus** directory specified:

```
# pntadm –r nisplus –p Test.Nis.Plus. –R 10.0.0.0
```

EXIT STATUS

0	Successful completion.
1	Object already exists.
2	Object does not exist.
3	Non-critical error.
4	Critical error.

FILES

/var/dhcp/XXX_XXX_XXX_XXX	files or NIS+ tables where **XXX** represents octets of the dotted IP address
/etc/default/dhcp	DHCP service configuration file
/etc/inet/hosts	file or NIS+ table

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu

SEE ALSO

dhcpconfig(1M), **dhcp**(4), **dhcp_network**(4), **dhcptab**(4), **hosts**(4), **netmasks**(4), **networks**(4), **attributes**(5)

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 1533, Lachman Technology, Inc., Bucknell University, October 1993.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 1541, Bucknell University, October 1993.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

NAME

ports – creates /dev entries and inittab entries for serial lines

SYNOPSIS

/usr/sbin/ports [**–r** *rootdir*]

DESCRIPTION

The **ports** command creates symbolic links in the **/dev/term** and **/dev/cua** directories to the serial-port character device files in **/devices** and adds new entries in **/etc/inittab** for non-system ports found. System-board ports are given single lower-case letters for names (such as **a** and **b**) while other ports are named numerically.

ports searches the kernel device tree to find the serial devices attached to the system. It also checks **/dev/term** and **/dev/cua** to see what symbolic links to serial devices already exist. **ports** then performs the following:

1. Assigns new numbers (or letters for system-board ports) to ports that are attached to the system but do not have **/dev/term** and **/dev/cua** entries. The numbers or letters assigned are the lowest-unused numbers or letters.
2. Removes dangling links: links from **/dev/term** and **/dev/cua** pointing to no-longer-existing ports.
3. Creates new **/dev/term** and **/dev/cua** links for new serial devices.
4. Invokes **sacadm**(1M) to make new port monitor entries for the new devices. This is not done automatically for on-board ports; on workstations these ports are often not used for dial-in sessions, so a port-monitor for one of these ports must be created explicitly.

If the configuration has not changed, **ports** exits without doing anything.

ports is run each time a reconfiguration-boot is performed, or when **add_drv**(1M) is executed. When invoking **ports** manually, first run **drvconfig**(1M) to ensure **/devices** is consistent with the current device configuration.

Notice to Driver Writers

ports considers devices with a node type of **DDI_NT_SERIAL**, **DDI_NT_SERIAL_MB**, **DDI_NT_SERIAL_DO**, or **DDI_NT_SERIAL_MB_DO** to be serial port devices. Devices with one of these node types must create minor device names that obey the following conventions when calling **ddi_create_minor_node**(9F).

- The minor name for non-system port devices (**DDI_NT_SERIAL**) consists of an ASCII numeric string, where the first port on the device is named **0**, the second named **1**, the third named **2**, up to the number of ports provided by the device.
- The minor name for non-system dialout devices (**DDI_NT_SERIAL_DO**) is the ASCII numeric port name, concatenated with **,cu**. For example, the minor name for the first dialout port on the serial board is **0,cu**.
- The minor name for system-board port devices (**DDI_NT_SERIAL_MB**) consists of a string containing a single ASCII lowercase character, where the first port on the device is named **a**, the second is named **b**, the third is named **c**, for all ports on the device (or up through port **z**).
- The minor name for system-board dialout devices (**DDI_NT_SERIAL_MB_DO**) consists

of the lowercase character port name, concatenated with **,cu**. For example, the minor name for the first dialout port on the on-board serial device is **a,cu**.

To prevent **disks** from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using a node type: **DDI_NT_BLOCK**, **DDI_NT_BLOCK_CHAN, DDI_NT_CD**, or **DDI_NT_CD_CHAN** when calling **ddi_create_minor_node**(9F).

OPTIONS

–r *rootdir* Causes **ports** to presume that the **/dev/term**, **/dev/cua**, and **/devices** directories are found under *rootdir*, not directly under **/**. If this argument is specified, **sacadm**(1M) is not invoked, since it would update terminal administration files under **/etc** without regard to the *rootdir*.

EXAMPLES

The following demonstrates creating the serial and dialout minor device nodes from the **xkserial** driver's **attach**(9E) function.

```
/*
 * Create the minor number by combining the instance number
 * with the port number.
 */
#define XKNUMPORTS          8
#define XKMINORNUM(i, p)    ((i) << 4 | (p))
#define XKMINORNUM_DO(i, p)  ((i) << 4 | (p) | 0x80)

int
xkserialattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
   int instance, portnum;
   char name[8];

          /* other stuff in attach... */

   instance = ddi_get_instance(dip);
   for (portnum = 0; portnum < XKNUMPORTS; portnum++) {
      /*
       * create the serial port device
       */
       sprintf(name, "%d", portnum);
       ddi_create_minor_node(dip, name, S_IFCHR,
          XKMINORNUM(instance, portnum), DDI_NT_SERIAL, 0);

      /*
       * create the dialout device
       */
       sprintf(name,"%d,cu", portnum);
       ddi_create_minor_node(dip, name, S_IFCHR,
            XKMINORNUM_DO(instance, portnum), DDI_NT_SERIAL_DO, 0);
```

```
        }
    }
```

Installing the **xkserial** port driver on a SPARCstation 20 (with the driver controlling the fictional XKSerial 8 port serial board) and performing a reconfiguration-boot would create the following special files in **/devices**.

```
# ls -l /devices/iommu@f,e0000000/sbus@f,e0001000/xkserial@f,800000/

crw-r-----   1 root sys   32,  16 Aug 29 00:02 xkserial@2000:0
crw-r-----   1 root sys   32, 144 Aug 29 00:02 xkserial@2000:0,cu
crw-r-----   1 root sys   32,  17 Aug 29 00:02 xkserial@2000:1
crw-r-----   1 root sys   32, 145 Aug 29 00:02 xkserial@2000:1,cu
crw-r-----   1 root sys   32,  18 Aug 29 00:02 xkserial@2000:2
crw-r-----   1 root sys   32, 146 Aug 29 00:02 xkserial@2000:2,cu
crw-r-----   1 root sys   32,  19 Aug 29 00:02 xkserial@2000:3
crw-r-----   1 root sys   32, 147 Aug 29 00:02 xkserial@2000:3,cu
crw-r-----   1 root sys   32,  20 Aug 29 00:02 xkserial@2000:4
crw-r-----   1 root sys   32, 148 Aug 29 00:02 xkserial@2000:4,cu
crw-r-----   1 root sys   32,  21 Aug 29 00:02 xkserial@2000:5
crw-r-----   1 root sys   32, 149 Aug 29 00:02 xkserial@2000:5,cu
crw-r-----   1 root sys   32,  22 Aug 29 00:02 xkserial@2000:6
crw-r-----   1 root sys   32, 150 Aug 29 00:02 xkserial@2000:6,cu
crw-r-----   1 root sys   32,  23 Aug 29 00:02 xkserial@2000:7
crw-r-----   1 root sys   32, 151 Aug 29 00:02 xkserial@2000:7,cu
```

/dev/term will contain symbolic links to the serial port device nodes in **/devices**

```
# ls -l /dev/term
/dev/term/0 -> ../../devices/[....]/xkserial@2000:0
/dev/term/1 -> ../../devices/[....]/xkserial@2000:1
/dev/term/2 -> ../../devices/[....]/xkserial@2000:2
/dev/term/3 -> ../../devices/[....]/xkserial@2000:3
/dev/term/4 -> ../../devices/[....]/xkserial@2000:4
/dev/term/5 -> ../../devices/[....]/xkserial@2000:5
/dev/term/6 -> ../../devices/[....]/xkserial@2000:6
/dev/term/7 -> ../../devices/[....]/xkserial@2000:7
```

and **/dev/cua** will contain symbolic links to the dialout port device nodes in **/devices**

```
# ls -l /dev/cua

/dev/cua/0 -> ../../devices/[....]/xkserial@2000:0,cu
/dev/cua/1 -> ../../devices/[....]/xkserial@2000:1,cu
/dev/cua/2 -> ../../devices/[....]/xkserial@2000:2,cu
/dev/cua/3 -> ../../devices/[....]/xkserial@2000:3,cu
/dev/cua/4 -> ../../devices/[....]/xkserial@2000:4,cu
/dev/cua/5 -> ../../devices/[....]/xkserial@2000:5,cu
```

/dev/cua/6 -> ../../devices/[....]/xkserial@2000:6,cu
/dev/cua/7 -> ../../devices/[....]/xkserial@2000:7,cu

FILES

/dev/term/*n*	logical serial port devices
/dev/cua/*n*	logical dialout port devices
/etc/inittab	
/etc/saf/*∗*	

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

add_drv(1M), **devlinks**(1M), **disks**(1M), **drvconfig**(1M), **pmadm**(1M), **sacadm**(1M), **tapes**(1M), **attributes**(5), **attach**(9E), **ddi_create_minor_node**(9F)

Writing Device Drivers

NAME

powerd – power manager daemon

SYNOPSIS

/usr/lib/power/powerd [**–n**]

DESCRIPTION

The **powerd** daemon manages two types of system shutdown: automatic shutdown and low power shutdown. Low power shutdown is found on systems which support battery operation.

The daemon reads the automatic shutdown information from the file **/etc/power.conf**. The daemon will reread the automatic shutdown information whenever it receives a hang-up signal, **SIGHUP**.

Automatic shutdown can occur only if the following two conditions are met:

- The current time is between the start and finish times.
- The system has been idle for at least the set time period. System idleness is determined by the inactivity of the system. See **power.conf**(4).

The start and finish times are specified in **power.conf**(4) and measured from the start of the day (12:00 a.m.). If the finish time is less than or equal to the start time, the active period of the daemon will span from midnight to the finish time and from the start time to the following midnight. To specify continuous operation, the finish time may be set equal to the start time. To disable automatic shutdown, specify **noshutdown** for the *behavior* field.

Low power shutdown will occur if the system is running from battery, and the daemon monitors that the charge in the battery is too low to reliably continue operation.

Immediately prior to system shutdown, the daemon notifies **syslogd**(1M) of the shutdown, which broadcasts a notification.

OPTIONS

The following options are supported:

–n No broadcast mode. The daemon will silently shut down the system without notifying **syslogd**(1M).

FILES

/etc/power.conf power management configuration information file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpmu

SEE ALSO

pmconfig(1M), **poweroff**(1M), **syslogd**(1M), **power.conf**(4), **attributes**(5), **cpr**(7), **pm**(7D)

NOTES

The daemon uses shared memory IPC, which may increase the system image size if the shared memory module has not already been loaded.

The daemon ensures that only one daemon is running. If another daemon is running, then the new daemon will exit with an error. If the daemon dies unexpectedly (non-maskable signal), residual shared memory state will remain. Starting a new daemon will remove this residual state.

NAME

praudit – print contents of an audit trail file

SYNOPSIS

praudit [–**lrs**] [–**d***del*] [*filename* . . .]

DESCRIPTION

praudit reads the listed *filename*s (or standard input, if no *filename* is specified) and interprets the data as audit trail records as defined in **audit.log**(4). By default, times, user and group IDs (UIDs and GIDs, respectively) are converted to their ASCII representation. Record type and event fields are converted to their ASCII representation. A maximum of 100 audit files can be specified on the command line.

OPTIONS

–**l** Prints one line per record. The record type and event fields are always converted to their short ASCII representation as is done for the **-s** option.

–**r** Print records in their raw form. Times, UIDs, GIDs, record types, and events are displayed as integers. This option and the –**s** option are exclusive. If both are used, a format usage error message is output.

–**s** Print records in their short form. All numeric fields are converted to ASCII and displayed. The short ASCII representations for the record type and event fields are used. This option and the –**r** option are exclusive. If both are used, a format usage error message is output.

–**d***del* Use *del* as the field delimiter instead of the default delimiter, which is the comma. If *del* has special meaning for the shell, it must be quoted. The maximum size of a delimiter is four characters.

FILES

/etc/security/audit_event
/etc/security/audit_class

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

bsmconv(1M), **audit**(2), **getauditflags**(3), **audit.log**(4), **audit_class**(4), **audit_event**(4), **group**(4), **passwd**(4), **attributes**(5)

NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See **bsmconv**(1M) for more information.

NAME

prtconf – print system configuration

SPARC SYNOPSIS

/usr/sbin/prtconf [**–F**] [**–p**] [**–P**] [**–v**] [**–V**] [**–D**]

x86 SYNOPSIS

/usr/sbin/prtconf [**–p**] [**–P**] [**–v**] [**–V**] [**–D**]

DESCRIPTION

The **prtconf** command prints the system configuration information. The output includes the total amount of memory, and the configuration of system peripherals formatted as a device tree.

OPTIONS

–P Include information about pseudo devices. By default, information regarding pseudo devices is omitted.

–v Specifies verbose mode.

–F (SPARC only). Return the device pathname of the console frame buffer, if one exists. If there is no frame buffer, **prtconf** returns a non-zero exit code. This flag overrides all others, and returns only the name of the console, frame buffer device or a non-zero exit code. For example, if the console frame buffer on a SPARCstation 1 is cgthree in SBus slot #3, the command returns: **/sbus@1,f80000000/cgthree@3,0**. This option could be used to create a symlink for **/dev/fb** to the actual console device.

–p Display information derived from the device tree provided by the firmware (PROM) on SPARC platforms or the booting system on x86 platforms.

–V Display platform-dependent PROM (on SPARC platforms) or booting system (on x86 platforms) version information. This flag must be used by itself, because it overrides all others. The output is a string. The format of the string is arbitrary and platform-dependent.

–D For each system peripheral in the device tree, display the name of the device driver used to manage the peripheral.

EXIT STATUS

If successful, **prtconf** returns **0**. If an error occurs, **prtconf** prints an error message and returns **1**. For example, when an illegal option is specified, **prtconf** returns **1**. On a SPARC system, when the **–F** option is specified and the console output device is not a framebuffer, **prtconf** returns **1**.

SPARC EXAMPLES

Running **prtconf** on a Sun4/65 series machine produces the following sample output:

```
example% prtconf
System Configuration:  Sun Microsystems  sun4c
Memory size: 16 Megabytes
System Peripherals (Software Nodes):

Sun 4_65
   options, instance #0
   zs, instance #0
```

```
        zs, instance #1
        fd (driver not attached)
        audio (driver not attached)
        sbus, instance #0
            dma, instance #0
            esp, instance #0
                sd (driver not attached)
                st (driver not attached)
                sd, instance #0
                sd, instance #1 (driver not attached)
                sd, instance #2 (driver not attached)
                sd, instance #3
                sd, instance #4 (driver not attached)
                sd, instance #5 (driver not attached)
                sd, instance #6 (driver not attached)
            le, instance #0
            cgsix (driver not attached)
        auxiliary-io (driver not attached)
        interrupt-enable (driver not attached)
        memory-error (driver not attached)
        counter-timer (driver not attached)
        eeprom (driver not attached)
        pseudo, instance #0
```

x86 EXAMPLES

Running **prtconf** on an x86 machine produces the following sample output:

```
example% prtconf
System Configuration:  Sun Microsystems  i86pc
Memory size: 32 Megabytes
System Peripherals (Software Nodes):

i86pc
    eisa, instance #0
        kd, instance #0
        ata, instance #0
            cmdk, instance #0
        aha, instance #0
            cmdk, instance #1 (driver not attached)
            cmdk, instance #2 (driver not attached)
            cmdk, instance #3 (driver not attached)
            cmdk, instance #4 (driver not attached)
            cmdk, instance #5 (driver not attached)
            cmdk, instance #6 (driver not attached)
            cmdk, instance #7
        chanmux, instance #0
        asy, instance #0
        asy, instance #1
        elx, instance #0
        elx, instance #1 (driver not attached)
        elx, instance #2 (driver not attached)
        elx, instance #3 (driver not attached)
        fdc, instance #0
            fd, instance #0
            fd, instance #1
    options, instance #0
    objmgr, instance #0
    pseudo, instance #0
example%
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

modinfo(1M), **sysdef**(1M), **attributes**(5)

SPARC Only

openprom(7D)

NOTES

The output of the **prtconf** command is highly dependent on the version of the PROM installed in the system. The output will be affected in potentially all circumstances.

The **driver not attached** message means that no driver is currently attached to that instance of the device. In general, drivers are loaded and installed (and attached to hardware instances) on demand, and when needed, and may be uninstalled and unloaded when the device is not in use.

NAME

prtdiag – display system diagnostic information

SYNOPSIS

/usr/platform/*platform-name***/sbin/prtdiag** [**–v**] [**–l**]

DESCRIPTION

prtdiag displays system configuration and diagnostic information.

The diagnostic information lists any failed Field Replaceable Units (FRUs) in the system.

The interface, output, and location in the directory hierarchy for **prtdiag** are uncommitted and subject to change in future releases.

platform-name is the name of the platform implementation and can be found using the **–i** option of **uname**(1).

OPTIONS

The following options are supported:

–v Verbose mode. Displays the time of the most recent AC Power failure, and the most recent hardware fatal error information, and (if applicable) environmental status.

The hardware fatal error information is useful to repair and manufacturing for detailed diagnostics of FRUs.

–l Log output. If failures or errors exist in the system, output this information to **syslogd**(1M) only.

EXIT STATUS

The following exit values are returned:

0 No failures or errors are detected in the system.

1 Failures or errors are detected in the system.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm

SEE ALSO

uname(1), **modinfo**(1M), **prtconf**(1M), **psrinfo**(1M), **sysdef**(1M), **syslogd**(1M), **attributes**(5), **openprom**(7D)

NAME

prtvtoc – report information about a disk geometry and partitioning

SYNOPSIS

prtvtoc [**–fhs**] [**–t** *vfstab*] [**–m** *mnttab*] *device*

DESCRIPTION

The **prtvtoc** command allows the contents of the VTOC (volume table of contents) to be viewed. The command can be used only by the super-user.

The *device* name can be the file name of a raw device in the form of **/dev/rdsk/c?t?d?s2** or can be the file name of a block device in the form of **/dev/dsk/c?t?d?s2**.

OPTIONS

The following options are supported:

–f Report on the disk free space, including the starting block address of the free space, number of blocks, and unused partitions.

–h Omit the headers from the normal output.

–s Omit all headers but the column header from the normal output.

–t *vfstab* Use *vfstab* as the list of filesystem defaults, in place of **/etc/vfstab**.

–m *mnttab* Use *mnttab* as the list of mounted filesystems, in place of **/etc/mnttab**.

EXAMPLES

The command line entry and system response shown below are for a 424-megabyte hard disk:

```
example# prtvtoc /dev/rdsk/c0t3d0s2
* /dev/rdsk/c0t3d0s2  partition map
*
* Dimension:
*     512 bytes/sector
*      80 sectors/track
*       9 tracks/cylinder
*     720 sectors/cylinder
*    2500 cylinders
*    1151 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                                 First     Sector    Last
* Partition    Tag    Flags     Sector     Count     Sector    Mount Directory
       0        2      00            0     76320      76319    /
       1        3      01        76320    132480     208799
       2        5      00            0    828720     828719
       5        6      00       208800    131760     340559    /opt
       6        4      00       340560    447120     787679    /usr
       7        8      00       787680     41040     828719    /export/home
example#
```

Codes for TAG are:

NAME	**NUMBER**
UNASSIGNED	**0**
BOOT	**1**
ROOT	**2**
SWAP	**3**
USR	**4**
BACKUP	**5**
VAR	**7**
HOME	**8**

FLAG indicates how the partition is to be mounted.

NAME	**NUMBER**
MOUNTABLE, READ AND WRITE	**00**
NOT MOUNTABLE	**01**
MOUNTABLE, READ ONLY	**10**

The following example shows output for the –**f** option for the same disk as above.

```
example# prtvtoc –f /dev/rdsk/c0t3d0s0
FREE_START=0 FREE_SIZE=0 FREE_COUNT=0 FREE_PART=4
example#
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

devinfo(1M), **fmthard**(1M), **format**(1M), **attributes**(5)

WARNINGS

The **mount** command does not check the "not mountable" bit.

NAME

psradm – set processors on-line or off-line

SYNOPSIS

psradm –f | **–n** [**–v**] *processor_id* ...

psradm –a –f | **–n** [**–v**]

DESCRIPTION

psradm sets processors on-line or off-line. An off-line processor does little or no work. The actual effect of being off-line may vary from machine to machine.

A processor may not be taken off-line if there are LWPs (lightweight processes) that are bound to the processor. On some architectures, it might not be possible to take certain processors off-line if, for example, the system depends on some resource provided by the processor.

At least one processor must remain on-line. If any of the specified processors are powered off, **psradm** may power on one or more processors.

Only superusers can use the **psradm** command.

OPTIONS

The following options are supported:

–a Perform the action on all processors, or as many as possible.

–f Take the specified processors off-line.

–n Bring the specified processors on-line.

–v Output a message giving the results of each attempted operation.

OPERANDS

The following operands are supported:

processor_id The processor ID of the processor to be set on-line or off-line.

EXAMPLES

The following example sets processors 2 and 3 off-line.

```
psradm –f 2 3
```

The following example sets all processors on-line.

```
psradm –a –n
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

FILES

/etc/wtmp for records logging processor status changes

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

psrinfo(1M), **psrset**(1M), **p_online**(2), **attributes**(5)

DIAGNOSTICS

psradm: processor 4: Invalid argument
The specified processor does not exist in the configuration.

psradm: processor 3: Device busy
The specified processor could not be taken off-line because it either has LWPs bound to it, is the last on-line processor in the system, or is needed by the system because it provides some essential service.

psradm: processor 3: Device busy
The specified processor is powered off, and it cannot be powered on because some platform-specific resource is unavailable.

psradm: processor 0: Not owner
The user does not have permission to change processor status.

psradm: processor 2: Operation not supported
The specified processor is powered off, and the platform does not support power on of individual processors.

NAME

psrinfo – displays information about processors

SYNOPSIS

psrinfo [**–v**] [*processor_id* . . .]

psrinfo –s *processor_id*

DESCRIPTION

psrinfo displays information about processors.

Without the *processor_id* operand, **psrinfo** displays one line for each configured processor, displaying whether it is on-line, off-line, or powered off, and when that status last changed. Use the *processor_id* operand to display information about a specific processor. See **OPERANDS**.

OPTIONS

The following options are supported:

–s *processor_id* Silent mode. Displays **1** if the specified processor is on-line, and **0** if the specified processor is off-line or powered off.

Use silent mode when using **psrinfo** in shell scripts.

–v Verbose mode. Displays additional information about the specified processors, including: processor type, floating point unit type and clock speed. If any of this information cannot be determined, **psrinfo** displays **unknown**.

OPERANDS

The following operands are supported:

processor_id The processor ID of the processor about which information is to be displayed.

EXAMPLES

The following example displays information about all configured processors in verbose mode.

```
psrinfo –v
```

The following example uses **psrinfo** in a shell script to determine if a processor is on-line.

```
if [ "‘psrinfo –s 3 2> /dev/null‘" –eq 1 ]
then
        echo "processor 3 is up"
fi
```

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

psradm(1M), **p_online**(2), **processor_info**(2), **attributes**(5)

DIAGNOSTICS

psrinfo: processor 9: Invalid argument
The specified processor does not exist.

NAME

psrset – creation and management of processor sets

SYNOPSIS

psrset –**c** [*processor_id ...*]
psrset –**d** *processor_set_id*
psrset –**a** *processor_set_id processor_id ...*
psrset –**r** *processor_id ...*
psrset –**p** [*processor_id ...*]
psrset –**b** *processor_set_id pid ...*
psrset –**u** *pid ...*
psrset –**q** [*pid ...*]
psrset [–**i**] [*processor_set_id ...*]

DESCRIPTION

psrset controls the management of processor sets. Processor sets allow the binding of processes to groups of processors, rather than just a single processor. There are two types of processor sets, those created by the user using the **psrset** command or the **pset_create**(2) system call, and those automatically created by the system. Processors assigned to user-created processor sets will run only LWPs that have been bound to that processor set, but system processor sets may run other LWPs as well.

System-created processor sets will not always exist on a given machine. When they exist, they will generally represent particular characteristics of the underlying machine, such as groups of processors that can communicate more quickly with each other than with other processors in the system. These processor sets cannot be modified or removed, but processes may be bound to them.

OPTIONS

–**c** Creates a new processor set.

–**d** Removes the specified processor set, releasing all processors and processes associated with it.

–**a** Assigns the specified processors to the specified processor set.

–**r** Removes the specified processors from the processor sets to which they are assigned.

–**p** Displays the processor set assignments of the specified processors, or of all processors.

–**b** Binds all the LWPs of the specified processes to the specified processor set.

–**u** Removes the processor set bindings of all LWPs of the specified processes.

–**q** Displays the processor set bindings of the specified processes, or of all processes.

–**i** Displays the type and processor assignments of the specified processor sets, or of all processor sets.

USAGE

The –**c** option creates a processor set and displays the new processor set ID. If a list of processors is given, it also attempts to assign those processors to the processor set. If this succeeds, the processors will be idle until LWPs are bound to the processor set. This option is restricted to use by the super-user.

The –**d** option removes a previously created processor set. Processor sets automatically created by the system cannot be removed. This option is restricted to use by the super-user.

The –**a** option assigns a list of processors to a processor set. Processor sets automatically created by the system cannot have processors assigned to them. However, processors belonging to system processor sets may be assigned to user-created processor sets. This option is restricted to use by the super-user.

The –**r** option removes a list of processors from their current processor sets. Processors that are removed will return to either the system processor set to which they previously belonged, or to the general pool of processors if they did not belong to a system processor set. This option is restricted to use by the super-user.

Processors with LWPs bound to them using **pbind**(1M) cannot be assigned to or removed from processor sets.

The –**p** option displays the processor set assignments for the specified list of processors. If no argument is given, the processor set assignments for all processors in the system is given.

The –**b** option binds all of the LWPs of the specified processes to the specified processor set. LWPs bound to a processor set will be restricted to run only on the processors in that set unless they require resources available only on another processor. Processes may only be bound to non-empty processor sets, that is, processor sets that have had processors assigned to them.

Bindings are inherited, so new LWPs and processes created by a bound LWP will have the same binding. Binding an interactive shell to a processor, for example, binds all commands executed by the shell.

The –**u** option removes the processor set bindings from all the LWPs of the specified processes, allowing them to be executed on any on-line processor if they are not bound to individual processors through **pbind**.

The super-user may bind or unbind any process to any active processor set. Other users may only bind or unbind processes to system processor sets. Furthermore, they may only bind or unbind processes for which they have permission to signal, that is, any process that has the same effective user ID as the user.

The –**q** option displays the processor set bindings of the specified processes. If a process is composed of multiple LWPs, which have different bindings, the bindings of only one of the bound LWPs will be shown. If no argument is given, the processor set bindings of all processes in the system is displayed.

The –**i** option displays a list of processors assigned to each named processor set. If no argument is given, a list of all processor sets and the processors assigned to them is displayed. This is also the default operation if the **psrset** command is not given an option.

Formats

The following format will be used for the first line of output of the **−c** option when the **LC_MESSAGES** locale category specifies the "C" locale. In other locales, the strings **created**, **processor**, and **set** may be replaced with more appropriate strings corresponding to the locale.

"**created processor set %d\n**" <*processor set ID*>

EXIT STATUS

The following exit values are returned:

0 Successful completion.

non-0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pbind(1M), **psradm**(1M), **psrinfo**(1M), **processor_bind**(2), **processor_info**(2), **pset_bind**(2), **pset_create**(2), **pset_info**(2), **sysconf**(3C), **attributes**(5)

DIAGNOSTICS

psrset: cannot query pid 31: No such process
The process specified did not exist or has exited.

psrset: cannot bind pid 31: Not owner
The user does not have permission to bind the process.

psrset: cannot assign processor 4: Not owner
The user does not have permission to assign the processor.

psrset: cannot assign processor 8: Invalid argument
The specified processor is not on-line, or the specified processor does not exist.

psrset: cannot bind pid 67: Device busy
An LWP in the specified process is bound to a processor and cannot be bound to a processor set that does not include that processor.

psrset: cannot assign processor 7: Device busy
The specified processor could not be added to the processor set. This may be due to bound LWPs on that processor, or because that processor cannot be combined in the same processor set with other processors in that set, or because the processor is the last one in its current processor set.

NAME

putdev – edits device table

SYNOPSIS

putdev –a *alias* [*attribute*=*value* [**...**]]
putdev –m *device attribute*=*value* [*attribute*=*value* [**...**]]
putdev –d *device* [*attribute* [**...**]]

DESCRIPTION

putdev can add a new device to the device table, modify an existing device description or remove a device entry from the table. The first synopsis is used to add a device. The second synopsis is used to modify existing entries by adding or changing attributes. If a specified attribute is not defined, this option adds that attribute to the device definition. If a specified attribute is already defined, it modifies the attribute definition. The third synopsis is used to delete either an entire device entry or, if the attribute argument is used, to delete an attribute assignment for a device.

OPTIONS

–a Add a device to the device table using the specified attributes. The device must be referenced by its *alias*.

–m Modify a device entry in the device table. If an entry already exists, it adds any specified attributes that are not defined. It also modifies any attributes which already have a value with the value specified by this command.

–d Remove a device from the device table, when executed without the *attributes* argument. Used with the *attribute* argument, it deletes the given attribute specification for *device* from the table.

alias Designate the alias of the device to be added.

device Designate the pathname or alias of the device whose attribute is to be added, modified, or removed.

attribute Designate a device attribute to be added, modified, or deleted. Can be any of the device attributes described under **DEVICE ATTRIBUTES** except **alias**. This prevents an accidental modification or deletion of a device’s alias from the table.

value Designate the value to be assigned to a device’s attribute.

DEVICE ATTRIBUTES

The following list shows the standard device attributes, used by applications such as **ufsdump**(1M) and **ufsrestore**(1M), which can be defined for a device. You are not limited to this list, you can define any attribute you like.

alias The unique name by which a device is known. No two devices in the database may share the same alias name. The name is limited in length to 14 characters and should contain only alphanumeric characters and the following special characters if they are escaped with a backslash: underscore (_), dollar sign ($), hyphen (–), and period (.).

bdevice The pathname to the block special device node associated with the device, if any. The associated major / minor combination should be unique within the database and should match that associated with the **cdevice** field, if any. (It is the administrator's responsibility to ensure that these **major/minor** numbers are unique in the database.)

capacity The capacity of the device or of the typical volume, if removable.

cdevice The pathname to the character special device node associated with the device, if any. The associated major / minor combination should be unique within the database and should match that associated with the **bdevice** field, if any. (It is the administrator's responsibility to ensure that these **major/minor** numbers are unique in the database.)

cyl Used by the command specified in the **mkfscmd** attribute.

desc A description of any instance of a volume associated with this device (such as floppy diskette).

dpartlist The list of disk partitions associated with this device. Used only if **type=disk**. The list should contain device aliases, each of which must have **type=dpart**.

dparttype The type of disk partition represented by this device. Used only if **type=dpart**. It should be either **fs** (for file system) or **dp** (for data partition).

erasecmd The command string that, when executed, erases the device.

fmtcmd The command string that, when executed, formats the device.

fsname The file system name on the file system administered on this partition, as supplied to the **/usr/sbin/labelit** command. This attribute is specified only if **type=dpart** and **dparttype=fs**.

gap Used by the command specified in the **mkfscmd** attribute.

mkfscmd The command string that, when executed, places a file system on a previously formatted device.

mountpt The default mount point to use for the device. Used only if the device is mountable. For disk partitions where **type=dpart** and **dparttype=fs**, this attribute should specify the location where the partition is normally mounted.

nblocks The number of blocks in the file system administered on this partition. Used only if **type=dpart** and **dparttype=fs**.

ninodes The number of inodes in the file system administered on this partition. Used only if **type=dpart** and **dparttype=fs**.

norewind The name of the character special device node that allows access to the serial device without rewinding when the device is closed.

pathname Defines the pathname to an i-node describing the device (used for non-block or character device pathnames, such as directories).

type A token that represents inherent qualities of the device. Standard types include: 9-track, ctape, disk, directory, diskette, dpart, and qtape.

volname The volume name on the file system administered on this partition, as supplied to the **/usr/sbin/labelit** command. Used only if **type=dpart** and **dparttype=fs**.

volume A text string used to describe any instance of a volume associated with this device. This attribute should not be defined for devices which are not removable.

ERRORS

putdev will exit with one of the following values:

0 Successful completion of the task.

1 Command syntax incorrect, invalid option used, or internal error occurred.

2 Device table could not be opened for reading, or new device table could not be created.

3 If executed with the **−a** option, indicates that an entry in the device table with the alias *alias* already exits. If executed with the **−m** or **−d** options, indicates that no entry exists for device *device.*

4 Indicates that **−d** was requested and one or more of the specified attributes were not defined for the device.

FILES

/etc/device.tab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

devattr(1M), **putdgrp**(1M), **ufsdump**(1M), **ufsrestore**(1M), **attributes**(5)

System Administration Guide

NAME

putdgrp – edits device group table

SYNOPSIS

putdgrp [–**d**] *dgroup* [*device* **...**]

DESCRIPTION

putdgrp modifies the device group table. It performs two kinds of modification. It can modify the table by creating a new device group or removing a device group. It can also change group definitions by adding or removing a device from the group definition.

When the command is invoked with only a *dgroup* specification, the command adds the specified group name to the device group table if it does not already exist. If the –**d** option is also used with only the *dgroup* specification, the command deletes the group from the table.

When the command is invoked with both a *dgroup* and a *device* specification, it adds the given device name(s) to the group definition. When invoked with both arguments and the **-d** option, the command deletes the device name(s) from the group definition.

When the command is invoked with both a *dgroup* and a *device* specification and the device group does not exist, it creates the group and adds the specified devices to that new group.

OPTIONS

–**d** Delete the group or, if used with *device*, delete the device from a group definition.

dgroup Specify a device group name.

device Specify the pathname or alias of the device that is to be added to, or deleted from, the device group.

ERRORS

The command will exit with one of the following values:

0 = successful completion of the task.

1 = command syntax incorrect, invalid option used, or internal error occurred.

2 = device group table could not be opened for reading or a new device group table could not be created.

3 = if executed with the –**d** option, indicates that an entry in the device group table for the device group *dgroup* does not exist and so cannot be deleted. Otherwise, indicates that the device group *dgroup* already exists and cannot be added.

4 = if executed with the –**d** option, indicates that the device group *dgroup* does not have as members one or more of the specified devices. Otherwise, indicates that the device group *dgroup* already has one or more of the specified devices as members.

EXAMPLES

To add a new device group:

```
putdgrp floppies
```

To add a device to a device group:

```
putdgrp floppies diskette2
```

To delete a device group:

```
putdgrp –d floppies
```

To delete a device from a device group:

```
putdgrp –d floppies diskette2
```

FILES

/etc/dgroup.tab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

listdgrp(1M), **putdev**(1M), **attributes**(5)

NAME

pwck, grpck – password / group file checkers

SYNOPSIS

/usr/sbin/pwck [*filename*]
/usr/sbin/grpck [*filename*]

DESCRIPTION

pwck scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and the program-to-use-as-shell exist. The default password file is **/etc/passwd**.

grpck verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, whether any login names belong to more than NGROUPS_MAX groups and that all login names appear in the password file. The default group file is **/etc/group**.

FILES

/etc/group
/etc/passwd

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

getpwent(3C), **group**(4), **passwd**(4), **attributes**(5)

DIAGNOSTICS

Group entries in **/etc/group** with no login names are flagged.

Group file '*filename***' is empty**
The **/etc/passwd** or **/etc/group** file is an empty file.

cannot open file *filename***: No such file or directory**
The **/etc/passwd** or **/etc/group** file does not exist.

NOTES

If no filename argument is given, **grpck** checks the local group file, **/etc/group**, and also makes sure that all login names encountered in the checked group file are known to the system **getpwent**(3C) routine. This means that the login names may be supplied by a network name service.

NAME

pwconv – installs and updates /etc/shadow with information from /etc/passwd

SYNOPSIS

pwconv

DESCRIPTION

The **pwconv** command creates and updates **/etc/shadow** with information from **/etc/passwd**.

pwconv relies on a special value of 'x' in the password field of **/etc/passwd.** This value of 'x' indicates that the password for the user is already in **/etc/shadow** and should not be modified.

If the **/etc/shadow** file does not exist, this command will create **/etc/shadow** with information from **/etc/passwd**. The command populates **/etc/shadow** with the user's login name, password, and password aging information. If password aging information does not exist in **/etc/passwd** for a given user, none will be added to **/etc/shadow**. However, the last changed information will always be updated.

If the **/etc/shadow** file does exist, the following tasks will be performed:

> Entries that are in the **/etc/passwd** file and not in the **/etc/shadow** file will be added to the **/etc/shadow** file.
>
> Entries that are in the **/etc/shadow** file and not in the **/etc/passwd** file will be removed from **/etc/shadow**.
>
> Password attributes (for example, password and aging information) that exist in an **/etc/passwd** entry will be moved to the corresponding entry in **/etc/shadow**.

The **pwconv** command can only be used by the super-user.

FILES

/etc/opasswd
/etc/oshadow
/etc/passwd
/etc/shadow

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

passwd(1), **passmgmt**(1M), **usermod**(1M), **passwd**(4), **attributes**(5)

DIAGNOSTICS

pwconv exits with one of the following values:

0	SUCCESS.
1	Permission denied.
2	Invalid command syntax.
3	Unexpected failure. Conversion not done.
4	Unexpected failure. Password file(s) missing.

5 Password file(s) busy. Try again later.

6 Bad entry in **/etc/shadow** file.

NAME

quot – summarize file system ownership

SYNOPSIS

quot [**–acfhnv**] *filesystem*
quot –a [**–cfhnv**]

DESCRIPTION

quot displays the number of blocks (1024 bytes) in the named *filesystem* currently owned by each user. There is a limit of 2048 blocks. Files larger than this will be counted as a 2048 block file, but the total block count will be correct.

OPTIONS

The following options are supported:

–a Generate a report for all mounted file systems.

–c Display three columns giving a file size in blocks, the number of files of that size, and a cumulative total of blocks containing files of that size or a smaller size.

–f Display count of number of files as well as space owned by each user. This option is incompatible with the **–c** and **–v** options.

–h Estimate the number of blocks in the file. This does not account for files with holes in them.

–n Attach names to the list of files read from standard input. **quot –n** cannot be used alone, because it expects data from standard input. For example, the pipeline

```
ncheck myfilesystem | sort +0n | quot –n myfilesystem
```

will produce a list of all files and their owners. This option is incompatible with all other options.

–v In addition to the default output, display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

OPERANDS

filesystem mount-point of the filesystem being checked

USAGE

See **largefile**(5) for the description of the behavior of **quot** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

0	Successful operation.
32	Error condition (bad or missing argument, bad path, or other error).

FILES

/etc/mnttab	mounted file systems
/etc/passwd	to get user names

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

du(1M), **mnttab**(4), **passwd**(4), **attributes**(5), **largefile**(5)

NOTES

This command may only be used by the super-user.

NAME

quota – display a user's ufs file system disk quota and usage

SYNOPSIS

quota [**–v**] [*username*]

DESCRIPTION

quota displays users' ufs disk usage and limits. Only the super-user may use the optional *username* argument to view the limits of other users.

quota without options only display warnings about mounted file systems where usage is over quota. Remotely mounted file systems which do not have quotas turned on are ignored.

username can be the numeric UID of a user.

OPTIONS

–v Display user's quota on all mounted file systems where quotas exist.

USAGE

See **largefile**(5) for the description of the behavior of **quota** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/mnttab list of currently mounted filesystems

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

edquota(1M), **quotaon**(1M), **quotacheck**(1M), **repquota**(1M), **rquotad**(1M), **attributes**(5), **largefile**(5)

NOTES

quota will also display quotas for NFS mounted ufs-based file systems if the **rquotad** daemon is running. See **rquotad**(1M).

NAME

quotacheck – ufs file system quota consistency checker

SYNOPSIS

quotacheck [**–p**] [**–v**] *filesystem* . . .
quotacheck **–a** [**–pv**]

DESCRIPTION

quotacheck examines each mounted ufs file system, builds a table of current disk usage, and compares this table against the information stored in the file system's disk quota file. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated.

filesystem is either a file system mount point or the block device on which the file system resides.

quotacheck expects each file system to be checked to have a quota file named **quotas** in the root directory. If none is present, **quotacheck** will not check the file system.

quotacheck accesses the character special device in calculating the actual disk usage for each user. Thus, the file systems that are checked should be quiescent while **quotacheck** is running.

OPTIONS

–p Check quotas of file systems in parallel.

–v Indicate the calculated disk quotas for each user on a particular file system. **quotacheck** normally reports only those quotas modified.

–a Check the file systems which **/etc/mnttab** indicates are ufs file systems. These file systems must be read-write mounted with disk quotas enabled, and have an **rq** entry in the **mntopts** field in **/etc/vfstab**.

USAGE

See **largefile**(5) for the description of the behavior of **quotacheck** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/mnttab	mounted file systems
/etc/vfstab	list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

edquota(1M), **quota**(1M), **quotaon**(1M), **repquota**(1M), **attributes**(5), **largefile**(5), **quotactl**(7I)

NAME

quotaon, quotaoff – turn ufs file system quotas on and off

SYNOPSIS

quotaon [**–v**] *filesystem* . . .
quotaon **–a** [**–v**]

quotaoff [**–v**] *filesystem* . . .
quotaoff **–a** [**–v**]

DESCRIPTION

quotaon turns on disk quotas for one or more ufs file systems.

Before a file system may have quotas enabled, a file named **quotas**, owned by root, must exist in the root directory of the file system. See **edquota**(1M) for details on how to modify the contents of this file.

quotaoff turns off disk quotas for one or more ufs file systems.

The file systems specified must already be mounted.

These commands update the **mntopts** field of the appropriate entries in **/etc/mnttab** to indicate when quotas are on or off for each file system. If quotas are on, ‘‘quota’’ will be added to **mntopts**; if quotas are off, **mntopts** will be marked ‘‘noquota’’.

filesystem must be either the mount point of a file system, or the block device on which the file system resides.

OPTIONS

quotaon

–a This option is normally used at boot time to enable quotas. It applies only to those file systems in **/etc/vfstab** which have ‘‘rq’’ in the **mntopts** field, are currently mounted ‘‘rw’’, and have a **quotas** file in the root directory.

–v Display a message for each file system after quotas are turned on.

quotaoff

–a Force all file systems in **/etc/mnttab** to have their quotas disabled.

–v Display a message for each file system affected.

USAGE

See **largefile**(5) for the description of the behavior of **quotaon** and **quotaoff** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES

/etc/mnttab	mounted file systems
/etc/vfstab	list of default parameters for each file system

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

edquota(1M), **quota**(1M), **quotacheck**(1M), **repquota**(1M), **mnttab**(4), **vfstab**(4), **attributes**(5), **largefile**(5), **quotactl**(7I)

NAME

rdate – set system date from a remote host

SYNOPSIS

rdate *hostname*

DESCRIPTION

rdate sets the local date and time from the *hostname* given as an argument. You must be super-user on the local system. Typically **rdate** can be inserted as part of a startup script.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5)

NAME

reboot – restart the operating system

SYNOPSIS

/usr/sbin/reboot [–**dlnq**] [*boot arguments*]

DESCRIPTION

reboot restarts the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to the loaded kernel.

Although **reboot** can be run by the super-user at any time, **shutdown**(1M) is normally used first to warn all users logged in of the impending loss of service. See **shutdown**(1M) for details.

reboot performs a **sync**(1M) operation on the disks, and then a multi-user reboot is initiated. See **init**(1M) for details.

reboot normally logs the reboot to the system log daemon, **syslogd**(1M), and places a shutdown record in the login accounting file **/var/adm/wtmp**. These actions are inhibited if the –**n** or –**q** options are present.

Normally, the system will reboot itself at power-up or after crashes.

OPTIONS

–**d** Dump system core before rebooting. This option is provided for compatibility, but is not supported by the underlying **reboot**(3C) call.

–**l** Suppress sending a message to the system log daemon, **syslogd**(1M) about who executed **reboot**.

–**n** Avoid the **sync**(1M) operation. Use of this option can cause file system damage.

–**q** Quick. Reboot quickly and ungracefully, without shutting down running processes first.

boot arguments
These arguments are accepted for compatibility, and are passed unchanged to the **uadmin**(2) system call.

EXAMPLES

In the example below, the delimiter ‘—’ (two hyphens) must be used to separate the options of **reboot** from the arguments of **boot**(1M).

```
example# reboot –dl — –rv
```

FILES

/var/adm/wtmp login accounting file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

boot(1M), **crash**(1M), **fsck**(1M), **halt**(1M), **init**(1M), **shutdown**(1M), **sync**(1M), **syslogd**(1M), **uadmin**(2), **reboot**(3C), **attributes**(5)

NAME

rem_drv – remove a device driver from the system

SYNOPSIS

rem_drv [–**b** *basedir*] *device_driver*

DESCRIPTION

The **rem_drv** command informs the system that the device driver *device_driver* is no longer valid. If possible, **rem_drv** unloads *device_driver* from memory. Entries for the device in the **/devices** namespace are removed. **rem_drv** also updates the system driver configuration files.

If **rem_drv** has been executed, the next time the system is rebooted it will automatically perform a reconfiguration boot (see **kernel**(1M)).

OPTIONS

–**b** *basedir* Sets the path to the root directory of the diskless client. Used on the server to execute **rem_drv** for a client. The client machine must be rebooted to unload the driver.

EXAMPLES

The following example removes the **sd** driver from use:

```
example% rem_drv sd
```

The next example removes the driver from the **sun1** diskless client. The driver will not be uninstalled nor unloaded until the client machine is rebooted.

```
example% rem_drv -b /export/root/sun1 sd
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

add_drv(1M), **drvconfig**(1M), **kernel**(1M), **attributes**(5)

NAME

removef – remove a file from software database

SYNOPSIS

removef [[–**M**] –**R** *root_path*] [–**V** *fs_file*] *pkginst path* . . .

removef [[–**M**] –**R** *root_path*] [–**V** *fs_file*] –**f** *pkginst*

DESCRIPTION

removef informs the system that the user, or software, intends to remove a pathname. Output from **removef** is the list of input pathnames that may be safely removed (no other packages have a dependency on them).

OPTIONS

–**f** After all files have been processed, **removef** should be invoked with the –**f** option to indicate that the removal phase is complete.

–**M** Instruct **removef** not to use the **$***root_path***/etc/vfstab** file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

–**R** *root_path* Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, **/export/root/client1**).

–**V** *fs_file* Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the **$***root_path***/etc/vfstab** file is nonexistent or unreliable.

OPERANDS

pkginst The package instance from which the pathname is being removed.

path The pathname to be removed.

EXAMPLES

The following shows the use of **removef** in an optional pre-install script:

```
echo   "The following files are no longer part of this package
            and are being removed."
removef $PKGINST /dev/xt[0-9][0-9][0-9] |
while read pathname
do
            echo "$pathname"
            rm –f $pathname
done
removef –f $PKGINST || exit 2
```

EXIT STATUS

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pkginfo(1), **pkgmk**(1), **pkgparam**(1), **pkgproto**(1), **pkgtrans**(1), **installf**(1M), **pkgadd**(1M), **pkgask**(1M), **pkgchk**(1M), **attributes**(5)

Application Packaging Developer's Guide

NAME

repquota – summarize quotas for a ufs file system

SYNOPSIS

repquota [–**v**] *filesystem* . . .
repquota –**a** [–**v**]

DESCRIPTION

repquota prints a summary of the disk usage and quotas for the specified ufs file systems. The current number of files and amount of space (in kilobytes) is printed for each user along with any quotas created with **edquota**(1M).

The *filesystem* must have the file **quotas** in its root directory.

Only the super-user may view quotas which are not their own.

OPTIONS

–**a** Report on all mounted ufs file systems that have **rq** in the **mntopts** field of the **/etc/vfstab** file.

–**v** Report quotas for all users, even those who do not consume resources.

USAGE

See **largefile**(5) for the description of the behavior of **repquota** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

edquota(1M), **quota**(1M), **quotacheck**(1M), **quotaon**(1M), **attributes**(5), **largefile**(5), **quotactl**(7I)

NAME

re-preinstall – installs the JumpStart software on a system

SYNOPSIS

cdrom-mnt-pt/**Solaris_2.6/Tools/Boot/usr/sbin/install.d/re-preinstall**
[–**m** *Solaris_boot_dir*] [–**k** *platform_name*] *target-slice*

DESCRIPTION

re-preinstall installs the JumpStart software (Preinstall Boot Image) on a system, so you can power-on the system and have it automatically install the Solaris software (perform a JumpStart installation on the system). When you turn on a re-preinstalled system, the system looks for the JumpStart software on the system's default boot disk. *All* new SPARC systems have the JumpStart software already preinstalled.

There are two ways to use the **re-preinstall** command. The most common way is to run **re-preinstall** on a system to install the JumpStart software on its own default boot disk. This is useful if you want to restore a system to its original factory conditions. (See the first procedure described in **EXAMPLES**.)

You can also run **re-preinstall** on a system to install JumpStart software on any attached disk (non-boot disk). Once you install the JumpStart software on a disk, you can move the disk to a different system and perform a JumpStart installation on the different system. (See the second procedure described in **EXAMPLES**.)

re-preinstall creates a standard file system on the specified *target-slice* (usually slice 0), and **re-preinstall** makes sure there is enough space on the *target-slice* for the JumpStart software. If sufficient space is not available, **re-preinstall** fails with the following message:

re-preinstall: *target-slice* **too small** *xx* **Megabytes required**

You can use the **format**(1M) command to create sufficient space on the *target-slice* for the JumpStart software.

OPTIONS

The following options are supported:

–**k** *platform_name* Platform name of the system that will use the disk with the JumpStart software. The default is the platform name of the system running **re-preinstall**. (Use the **uname**(1) command (–**i** option) to determine a system's platform name.)

–**m** *Solaris_boot_dir* Absolute path to the **Solaris_2.6/Tools/Boot** subdirectory of a mounted Solaris CD or a Solaris CD copied to disk that **re-preinstall** uses to install the JumpStart software. The default is **/cdrom/Solaris_2.6/Tools/Boot**, which is where the Solaris CD is mounted in single-user mode.

OPERANDS

The following operands are supported:

target-slice Device name of the disk slice where the JumpStart software will be installed (usually slice 0). For example, **c0t3d0s0**.

EXAMPLES

The following procedure installs the Jumpstart software on a system's own default boot disk:

1. From the "ok" prompt, boot the system from the Solaris CD (local or remote) in single-user mode:

   ok **boot cdrom –s**

2. With the **re-preinstall** command, install the JumpStart software on the system's default boot disk, which is a slice on the disk (usually slice 0) where the system automatically boots from. (The system's default boot disk is probably where the current root (**/**) file system is located, which can be determined with the **format**(1M) command.)

   For example, the following command installs the JumpStart software on the system's default boot disk, **c0t3d0s0**:

```
example# /cdrom/Solaris_2.6/Tools/Boot/usr/sbin/install.d
         /re-preinstall c0t3d0s0
```

The following procedure installs the JumpStart software on a system's attached disk (non-boot disk):

1. Mount the Solaris CD if **vold**(1M) is not running or CD is not mounted.
2. Use the **format**(1M) command to determine the target-slice where JumpStart will be installed.
3. Use the **uname**(1) command (**–i** option) to determine the platform name of the system that will use the re-preinstalled disk
4. Run **re-preinstall** with the **–m** *Solaris_boot_dir* option if the Solaris CD is not mounted on **/cdrom**.

   For example, the following command installs the JumpStart software on the system's attached disk for a system with a Sun4c kernel architecture, and it uses the Solaris CD mounted with **vold**(1M):

```
example# /cdrom/cdrom0/s0/Solaris_2.6/Tools/Boot/usr/bin
         /install.d/re-preinstall
         –m /cdrom/cdrom0/s0/Solaris_2.6/Tools/Boot
         –k sun4c c0t2d0s0
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcdrom (Solaris CD, SPARC Platform Edition)

SEE ALSO

uname(1), **eeprom**(1M), **format**(1M), **mount**(1M), **vold**(1M), **attributes**(5)

Solaris Advanced Installation Guide

NAME

rmmount – removable media mounter that automatically mounts a file system on a CD-ROM and floppy

SYNOPSIS

/usr/sbin/rmmount [–**D**]

DESCRIPTION

rmmount is a removable media mounter that is executed by Volume Management whenever a CD-ROM or floppy is inserted. The Volume Management daemon, **vold**(1M), manages CD-ROM and floppy devices.

Upon insertion, **rmmount** determines what type of file system (if any) is on the media. If a file system is present, **rmmount** mounts the file system in one of the following locations.

Mount Location	**State of Media**
/floppy/floppy0	symbolic link to mounted floppy in local floppy drive
/floppy/floppy_name	mounted named floppy
/floppy/unnamed_floppy	mounted unnamed floppy
/cdrom/cdrom0	symbolic link to mounted CD-ROM in local CD-ROM drive
/cdrom/CD-ROM_name	mounted named CD-ROM
/cdrom/CD-ROM_name/partition	mounted named CD-ROM with partitioned file system
/cdrom/unnamed_cdrom	mounted unnamed CD-ROM

If the media is read-only (either CD-ROM or floppy with write-protect tab set), the file system is mounted read-only.

If a file system is not identified, **rmmount** does not mount a file system. See the *System Administration Guide* for more information on the location of CD-ROM and floppy media without file sytems. Also see **volfs**(7FS).

If a file system type has been determined, it is then checked to see that it is “clean.” If the file system is “dirty,” **fsck -p** (see **fsck**(1M)) is run in an attempt to clean it. If **fsck** fails, the file system is mounted read-only.

After the mount is complete, “actions” associated with the media type are executed. These actions allow for the notification to other programs that new media are available. These actions are shared objects and are described in the configuration file, **/etc/rmmount.conf**.

Actions are executed in the order in which they appear in the configuration file. The action function can return either **1** or **0**. If it returns **0**, no further actions will be executed. This allows the function to control which applications are executed.

In order to execute an action, **rmmount** performs a **dlopen**(3X) on the shared object and calls the action function defined within it. The definition of the interface to actions can be found in **/usr/include/rmmount.h**.

File systems mounted by **rmmount** are always mounted with the **nosuid** flag set, thereby disabling set-uid programs and access to block or character devices in that file system. Upon ejection, **rmmount** unmounts mounted file systems and executes actions associated with the media type. If a file system is “busy” (i.e., it contains the current working directory of a live process), the ejection will fail.

OPTIONS

−D Turn on the debugging output from the **rmmount** dprintf calls.

FILES

/etc/rmmount.conf	removable media mounter configuration file.
/usr/lib/rmmount/*.so.1	shared objects used by **rmmount**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWvolu

SEE ALSO

volcancel(1), **volcheck**(1), **volmissing**(1) **fsck**(1M), **vold**(1M), **dlopen**(3X), **rmmount.conf**(4), **vold.conf**(4), **attributes**(5), **volfs**(7FS)

System Administration Guide

NAME

rmt – remote magtape protocol module

SYNOPSIS

/usr/sbin/rmt

DESCRIPTION

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. **rmt** is normally started up with an **rexec**(3N) or **rcmd**(3N) call.

The **rmt** program accepts requests that are specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of

A*number* \ *n*

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

E*error-number* **\ n***error-message* **\ n**

where *error-number* is one of the possible error numbers described in **intro**(2), and *error-message* is the corresponding error string as printed from a call to **perror**(3C). The protocol consists of the following commands:

S \ n Return the status of the open device, as obtained with a **MTIOCGET ioctl** call. If the operation was successful, an “ack” is sent with the size of the status buffer, then the status buffer is sent (in binary).

C*device* **\ n** Close the currently open device. The *device* specified is ignored.

I*operation* **\ n***count* **\ n**
Perform a **MTIOCOP ioctl**(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the **ioctl** call. When the operation is successful the return value is the *count* parameter.

L*offset* **\ n***whence* **\ n**
Perform an **lseek**(2) operation using the specified parameters. The response value is returned from the **lseek** call.

O*device* **\ n***mode* **\ n**
Open the specified *device* using the indicated *mode*. *device* is a full pathname, and *mode* is an ASCII representation of a decimal number suitable for passing to **open**(9E). If a device is already open, it is closed before a new open is performed.

R*count***\n** Read *count* bytes of data from the open device. **rmt** performs the requested **read**(9E) and responds with **A***count-read\n* if the read was successful; otherwise an error in standard format is returned. If the read was successful, the data read is sent.

W*count***\n** Write data onto the open device. **rmt** reads *count* bytes from the connection, aborting if a premature EOF is encountered. The response value is returned from the **write**(9E) call.

Any other command causes **rmt** to exit.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ufsdump(1M), **ufsrestore**(1M), **intro**(2), **ioctl**(2), **lseek**(2), **perror**(3C), **rcmd**(3N), **rexec**(3N), **attributes**(5), **mtio**(7I), **open**(9E), **read**(9E), **write**(9E)

DIAGNOSTICS

All responses are of the form described above.

BUGS

Do not use this for a remote file access protocol.

NAME

route – manually manipulate the routing tables

SYNOPSIS

route [**–fnvq**] *command* [[*modifiers*] *args*]
route [**–fnvq**] **add** | **change** | **delete** | **get** [**host** | **net**] *destination* [*gateway* [*args*]]
route [**–n**] **monitor**
route [**–n**] **flush**

DESCRIPTION

route manually manipulates the network routing tables. These tables are normally maintained by the system routing daemon, by **routed**(1M), or through default routes and redirect messages from routers.

This utility supports a limited number of general options, but a rich command language. It enables the user to specify any arbitrary request that could be delivered via the programmatic interface discussed in **route**(4).

route uses a routing socket and the new message types RTM_ADD, RTM_DELETE, RTM_GET, and RTM_CHANGE. As such, only the super-user may modify the routing tables.

OPTIONS

–f Flush the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, **route** flushes the gateways before performing the command.

–n Prevent attempts to print host and network names symbolically when reporting actions. This is useful, for example, when all name servers are down on your local net, and you need a route before you can contact the name server.

–v (Verbose) Print additional details.

–q Suppress all output.

Commands

route executes one of four *commands* on a route to a *destination*. Two additional *commands* operate globally on all routing information. The (six) commands are:

add Add a route.

change Change aspects of a route (such as its gateway).

delete Delete a specific route.

flush Remove all gateway entries from the routing table.

get Lookup and display the route for a destination.

monitor Continuously report any changes to the routing information base, routing lookup misses, or suspected network partitionings.

The add, delete, and change commands have the following syntax:

route [**–fnvq**] **command** [**–net** | **–host**] *destination gateway* where *destination* is the destination host or network, and *gateway* is the next-hop intermediary via where packets should be routed.

OPERANDS

route executes its commands on routes to destinations.

Destinations

All symbolic names specified for a *destination* or *gateway* are looked up first as a host name, using **gethostbyname**(3N)**.** If this lookup fails, **getnetbyname**(3N) is used to interpret the name as that of a network.

An optional modifier may be included on the command line before a *destination*, to force how **route** interprets a destination:

–**host** Forces the destination to be interpreted as a host.
–**net** Forces the destination to be interpreted as a network.

Routes to a particular host may be distinguished from those to a network by interpreting the Internet address specified as the *destination*. If the *destination* has a "local address part" of INADDR_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

For example, the route:

	128.32	is interpreted as	–host	128.0.0.32
	128.32.130	is interpreted as	–host	128.32.0.130
–net	128.32	is interpreted as		128.32.0.0
–net	128.32.130	is interpreted as		128.32.130.0

If the destination is directly reachable via an interface requiring no intermediary system to act as a gateway, this can be indicated by including one of two optional modifiers after the destination: The –**interface** modifier can be included or a *metric* of **0** can be specified. These modifiers are illustrated in the following alternative examples:

```
route add default hostname -interface
route add default hostname 0
```

hostname is the name or IP address associated with the network interface all packets should be sent over. On a host with a single network interface, *hostname* is normally the same as the *nodename* returned by **uname –n** (see **uname**(1)).

In the above examples, the route does not refer to a gateway, but rather to one of the machine's interfaces. Destinations matching such a route are sent out on the interface identified by the *gateway* address. For interfaces using the **ARP** protocol, this type of route is used to specify *all destinations are local.* That is, a host should **ARP** for all addresses by adding a default route using one of the two commands listed above.

The optional –**netmask** qualifier is intended to manually add subnet routes with netmasks different from that of the implied network interface. The implicit network mask generated in the AF_INET case can be overridden by making sure this option, and an ensuing address parameter (to be interpreted as a network mask), follows the destination parameter.

Routing Flags

Routes have associated flags which influence operation of the protocols when sending to destinations matched by the routes. These flags may be set (or sometimes cleared) by including the following corresponding modifiers on the command line:

Modifier	Flag	Description
–cloning	**RTF_CLONING**	generates a new route on use
–xresolve	**RTF_XRESOLVE**	emit mesg on use (for external lookup)
–iface	**˜RTF_GATEWAY**	destination is directly reachable
–static	**RTF_STATIC**	manually added route
–nostatic	**˜RTF_STATIC**	pretend route added by kernel or daemon
–reject	**RTF_REJECT**	emit an ICMP unreachable when matched
–blackhole	**RTF_BLACKHOLE**	silently discard pkts (during updates)
–proto1	**RTF_PROTO1**	set protocol specific routing flag #1
–proto2	**RTF_PROTO2**	set protocol specific routing flag #2
–llinfo	**RTF_LLINFO**	validly translates proto addr to link addr

The optional modifiers:

–rtt,
–rttvar,
–sendpipe,
–recvpipe,
–mtu,
–hopcount,
–expire,
–ssthresh

provide initial values to quantities maintained in the routing entry by transport level protocols, such as TCP. These may be individually locked by preceding each such modifier to be locked by the –**lock** meta-modifier, or one can specify that all ensuing metrics may be locked by the –**lockrest** meta-modifier.

In a **change** or **add** command where the destination and gateway are not sufficient to specify the route (e.g., when several interfaces have the same address), the –**ifp** or –**ifa** modifiers may be used to determine the interface or interface address.

FILES

/etc/hosts	list of host names and net addresses
/etc/networks	list of network names and addresses

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

get(1), **uname**(1), **in.rdisc**(1M), **netstat**(1M), **routed**(1M), **ioctl**(2), **gethostbyname**(3N), **getnetbyname**(3N), **hosts**(4), **networks**(4), **route**(4), **routing**(4), **attributes**(5), **ARP**(7P)

DIAGNOSTICS

add [**host** | **network]** *destination*:*gateway flags*

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the **ioctl**(2) call. If the gateway address used was not the primary address of the gateway (the first one returned by **gethostbyname**(3N)**),** the gateway address is printed numerically as well as symbolically.

delete [**host** | **network**] *destination*:*gateway flags*

As above, but when deleting an entry.

destination **done**

When the –**f** flag is specified, or in the **flush** command, each routing table entry deleted is indicated with a message of this form.

Network is unreachable

An attempt to add a route failed because the gateway listed was not on a directly-connected network. Give the next-hop gateway instead.

not in table

A delete operation was attempted for an entry that is not in the table.

routing table overflow

An add operation was attempted, but the system was unable to allocate memory to create the new entry.

NOTES

All destinations are local assumes that the routers implement the protocol, **proxy arp**. Normally, using **router discovery** (see **in.rdisc**(1M)) is more reliable than using **proxy arp**.

Combining the *all destinations are local* route with subnet or network routes can lead to unpredictable results: the search order as it relates to the *all destinations are local* route are undefined and may vary from release to release.

NAME

rpcbind – universal addresses to RPC program number mapper

SYNOPSIS

rpcbind [–**d**] [–**w**]

DESCRIPTION

rpcbind is a server that converts RPC program numbers into universal addresses. It must be running on the host to be able to make RPC calls on a server on that machine.

When an RPC service is started, it tells **rpcbind** the address at which it is listening, and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts **rpcbind** on the server machine to determine the address where RPC requests should be sent.

rpcbind should be started before any other RPC service. Normally, standard RPC servers are started by port monitors, so **rpcbind** must be started before port monitors are invoked.

When **rpcbind** is started, it checks that certain name-to-address translation-calls function correctly. If they fail, the network configuration databases may be corrupt. Since RPC services cannot function correctly in this situation, **rpcbind** reports the condition and terminates.

rpcbind can only be started by the super-user.

OPTIONS

–**d** Run in debug mode. In this mode, **rpcbind** will not fork when it starts, will print additional information during operation, and will abort on certain errors. With this option, the name-to-address translation consistency checks are shown in detail.

–**w** Do a warm start. If **rpcbind** aborts or terminates on SIGINT or SIGTERM, it will write the current list of registered services to **/tmp/portmap.file** and **/tmp/rpcbind.file**. Starting **rpcbind** with the –**w** option instructs it to look for these files and start operation with the registrations found in them. This allows **rpcbind** to resume operation without requiring all RPC services to be restarted.

NOTES

Terminating **rpcbind** with SIGKILL will prevent the warm-start files from being written.

All RPC servers must be restarted if the following occurs: **rpcbind** crashes (or is killed with SIGKILL) and is unable to to write the warm-start files; **rpcbind** is started without the –**w** option after a graceful termination; or, the warm-start files are not found by **rpcbind.**

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

rpcinfo(1M), **rpcbind**(3N), **attributes**(5)

FILES

/tmp/portmap.file
/tmp/rpcbind.file

NAME

rpc.bootparamd, bootparamd – boot parameter server

SYNOPSIS

/usr/sbin/rpc.bootparamd [**−d**]

DESCRIPTION

rpc.bootparamd is a server process that provides information from a **bootparams** database to diskless clients at boot time. See **bootparams**(4)

The source for the **bootparams** database is determined by the **nsswitch.conf**(4) file (on the machine running the **rpc.bootparamd** process).

The **rpc.bootparamd** program can be invoked either by **inetd**(1M) or directly from the command line.

OPTIONS

−d Display debugging information.

FILES

/etc/bootparams	boot parameter data base
/etc/nsswitch.conf	configuration file for the name-service switch

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M), **bootparams**(4), **nsswitch.conf**(4), **attributes**(5)

NOTES

A diskless client requires service from at least one **rpc.bootparamd** process running on a server that is on the same IP subnetwork as the diskless client.

Some routines that compare hostnames use case-sensitive string comparisons; some do not. If an incoming request fails, verify that the case of the hostname in the file to be parsed matches the case of the hostname called for, and attempt the request again.

NAME

rpcinfo – report RPC information

SYNOPSIS

rpcinfo [**–m** | **–s**] [*host*]
rpcinfo **–p** [*host*]
rpcinfo **–T** *transport host prognum* [*versnum*]
rpcinfo **–l** [**–T** *transport*] *host prognum versnum*
rpcinfo [**–n** *portnum*] **–u** *host prognum* [*versnum*]
rpcinfo [**–n** *portnum*] **–t** *host prognum* [*versnum*]
rpcinfo **–a** *serv_address* **–T** *transport prognum* [*versnum*]
rpcinfo **–b** [**–T** *transport*] *prognum versnum*
rpcinfo **–d** [**–T** *transport*] *prognum versnum*

DESCRIPTION

rpcinfo makes an RPC call to an RPC server and reports what it finds.

In the first synopsis, **rpcinfo** lists all the registered RPC services with **rpcbind** on *host*. If *host* is not specified, the local host is the default. If **–s** is used, the information is displayed in a concise format.

In the second synopsis, **rpcinfo** lists all the RPC services registered with **rpcbind**, version 2. Note that the format of the information is different in the first and the second synopsis. This is because the second synopsis is an older protocol used to collect the information displayed (version 2 of the **rpcbind** protocol).

The third synopsis makes an RPC call to procedure 0 of *prognum* and *versnum* on the specified *host* and reports whether a response was received. *transport* is the transport which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote **rpcbind**.

The *prognum* argument is a number that represents an RPC program number (see **rpc**(4)).

If a *versnum* is specified, **rpcinfo** attempts to call that version of the specified *prognum*. Otherwise, **rpcinfo** attempts to find all the registered version numbers for the specified *prognum* by calling version 0, which is presumed not to exist; if it does exist, **rpcinfo** attempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version. Note that the version number is required for **–b** and **–d** options.

The **EXAMPLES** section describe other ways of using **rpcinfo**.

OPTIONS

–T *transport* Specify the transport on which the service is required. If this option is not specified, **rpcinfo** uses the transport specified in the **NETPATH** environment variable, or if that is unset or **NULL**, the transport in the **netconfig**(4) database is used. This is a generic option, and can be used in conjunction with other options as shown in the **SYNOPSIS**.

–a *serv_address* Use *serv_address* as the (universal) address for the service on *transport* to ping procedure 0 of the specified *prognum* and report whether a response was received. The **–T** option is required with the **–a** option.

If *versnum* is not specified, **rpcinfo** tries to ping all available version numbers for that program number. This option avoids calls to remote **rpcbind** to find the address of the service. The *serv_address* is specified in universal address format of the given transport.

–b Make an RPC broadcast to procedure 0 of the specified *prognum* and *versnum* and report all hosts that respond. If *transport* is specified, it broadcasts its request only on the specified transport. If broadcasting is not supported by any transport, an error message is printed. Use of broadcasting should be limited because of the potential for adverse effect on other systems.

–d Delete registration for the RPC service of the specified *prognum* and *versnum*. If *transport* is specified, unregister the service on only that transport, otherwise unregister the service on all the transports on which it was registered. Only the owner of a service can delete a registration, except the superuser who can delete any service.

–l Display a list of entries with a given *prognum* and *versnum* on the specified *host*. Entries are returned for all transports in the same protocol family as that used to contact the remote **rpcbind**.

–m Display a table of statistics of **rpcbind** operations on the given *host*. The table shows statistics for each version of **rpcbind** (versions 2, 3 and 4), giving the number of times each procedure was requested and successfully serviced, the number and type of remote call requests that were made, and information about RPC address lookups that were handled. This is useful for monitoring RPC activities on *host*.

–n *portnum* Use *portnum* as the port number for the **–t** and **–u** options instead of the port number given by **rpcbind**. Use of this option avoids a call to the remote **rpcbind** to find out the address of the service. This option is made obsolete by the **–a** option.

–p Probe **rpcbind** on *host* using version 2 of the **rpcbind** protocol, and display a list of all registered RPC programs. If *host* is not specified, it defaults to the local host. Note that version 2 of the **rpcbind** protocol was previously known as the portmapper protocol.

–s Display a concise list of all registered RPC programs on *host*. If *host* is not specified, it defaults to the local host.

–t Make an RPC call to procedure 0 of *prognum* on the specified *host* using TCP, and report whether a response was received. This option is made obsolete by the **–T** option as shown in the third synopsis.

–u Make an RPC call to procedure 0 of *prognum* on the specified *host* using UDP, and report whether a response was received. This option is made obsolete by the **–T** option as shown in the third synopsis.

EXAMPLES

To show all of the RPC services registered on the local machine use:

```
example% rpcinfo
```

To show all of the RPC services registered with **rpcbind** on the machine named **klaxon** use:

```
example% rpcinfo klaxon
```

The information displayed by the above commands can be quite lengthy. Use the **–s** option to display a more concise list:

```
        example% rpcinfo –s klaxon
 program   version(s)   netid(s)                          service     owner
   100000  2,3,4        tcp,udp,ticlts,ticots,ticotsord   rpcbind     superuser
   100008  1            ticotsord,ticots,ticlts,udp,tcp   walld       superuser
   100002  2,1          ticotsord,ticots,ticlts,udp,tcp   rusersd     superuser
   100001  2,3,4        ticotsord,ticots,tcp,ticlts,udp   rstatd      superuser
   100012  1            ticotsord,ticots,ticlts,udp,tcp   sprayd      superuser
   100007  3            ticotsord,ticots,ticlts,udp,tcp   ypbind      superuser
   100029  1            ticotsord,ticots,ticlts           keyserv     superuser
   100078  4            ticotsord,ticots,ticlts           kerbd       superuser
   100024  1            ticotsord,ticots,ticlts,udp,tcp   status      superuser
   100021  2,1          ticotsord,ticots,ticlts,udp,tcp   nlockmgr    superuser
   100020  1            ticotsord,ticots,ticlts,udp,tcp   llockmgr    superuser
```

To show whether the RPC service with program number *prognum* and version *versnum* is registered on the machine named **klaxon** for the transport TCP use:

```
example% rpcinfo –T tcp klaxon prognum versnum
```

To show all RPC services registered with version 2 of the **rpcbind** protocol on the local machine use:

```
example% rpcinfo –p
```

To delete the registration for version 1 of the **walld** (program number **100008**) service for all transports use:

```
example# rpcinfo –d 100008 1
```

or

```
example# rpcinfo –d walld 1
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

rpcbind(1M), **rpc**(3N), **netconfig**(4), **rpc**(4), **attributes**(5)

NAME

rpc.nisd, nisd – NIS+ service daemon

SYNOPSIS

/usr/sbin/rpc.nisd [**–ACDFhlv**] [**–Y** [**–B** [**–t** *netid*]]] [**–d** *dictionary*] [**–L** *load*] [**–S** *level*]

DESCRIPTION

The **rpc.nisd** daemon is an RPC service that implements the NIS+ service. This daemon must be running on all machines which serve a portion of the NIS+ namespace.

rpc.nisd is usually started from a system startup script.

The **–B** option causes **rpc.nisd** to start an auxiliary process, **rpc.nisd_resolv**, which provides ypserv compatible DNS forwarding for NIS host requests. **rpc.nisd_resolv** can also be started independently. See **rpc.nisd_resolv**(1M) for more information on using **rpc.nisd_resolv** independently.

OPTIONS

–A Authentication verbose mode. The daemon logs all the authentication related activities to **syslogd**(1M) with **LOG_INFO** priority.

–C Open diagnostic channel on **/dev/console**.

–D Debug mode (don't fork).

–F Force the server to do a checkpoint of the database when it starts up. Forced checkpoints may be required when the server is low on disk space. This option removes updates from the transaction log that have propagated to all of the replicas.

–h Print list of options.

–v Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon (see **syslogd**(1M)) at **LOG_INFO** priority. This option is most useful for debugging problems with the service (see also **–A** option).

–Y Put the server into NIS (YP) compatibility mode. When operating in this mode, the NIS+ server will respond to NIS Version 2 requests using the version 2 protocol. Because the YP protocol is not authenticated, only those items that have read access to nobody (the unauthenticated request) will be visible through the V2 protocol. It supports only the standard Version 2 maps in this mode (see **–B** option and **NOTES** in **ypfiles**(4)).

–B Provide ypserv compatible DNS forwarding for NIS host requests. The DNS resolving process, **rpc.nisd_resolv**, is started and controlled by **rpc.nisd**. This option requires that the **/etc/resolv.conf** file be setup for communication with a DNS nameserver. The **nslookup** utility can be used to verify communication with a DNS nameserver. See **resolv.conf**(4) and **nslookup**(1M).

–t *netid* Use *netid* as the transport for communication between **rpc.nisd** and **rpc.nisd_resolv**. The default transport is **ticots**(7D) (**tcp** on SunOS 4.x systems).

–d *dictionary*
Specify an alternate dictionary for the NIS+ database. The primary use of this option is for testing. Note that the string is not interpreted, rather it is simply passed to the **db_initialize** function. See **nis_db**(3N).

–L *number* Specify the ''load'' the NIS+ service is allowed to place on the server. The load is specified in terms of the *number* of child processes that the server may spawn. This *number must* be at least 1 for the callback functions to work correctly. The default is 128.

–S *level* Set the authorization security level of the service. The argument is a number between 0 and 2. By default, the daemon runs at security level 2.

0 Security level 0 is designed to be used for testing and initial setup of the NIS+ namespace. When running at level 0, the daemon does not enforce any access controls. Any client is allowed to perform any operation, including updates and deletions.

1 At security level 1, the daemon accepts both **AUTH_SYS** and **AUTH_DES** credentials for authenticating clients and authorizing them to perform NIS+ operations. This is not a secure mode of operation since **AUTH_SYS** credentials are easily forged. It should not be used on networks in which any untrusted users may potentially have access.

2 At security level 2, the daemon accepts only **AUTH_DES** credentials for authentication and authorization. This is the highest level of security currently provided by the **NIS+** service. This is the default security level if the **–S** option is not used.

EXAMPLES

The following example sets up the NIS+ service.

```
example% rpc.nisd
```

The following example sets up the NIS+ service, emulating YP with DNS forwarding.

```
example% rpc.nisd –YB
```

ENVIRONMENT

NETPATH The transports that the NIS+ service will use can be limited by setting this environment variable (see **netconfig**(4)).

FILES

/var/nis/parent.object This file contains an XDR encoded NIS+ object that describes the namespace above a root server. This parent namespace may be another NIS+ namespace or a foreign namespace such as one served by the Domain Name Service. It is only present on servers that are serving the root of the namespace.

/var/nis/root.object This file contains an XDR encoded NIS+ object that describes the root of the namespace. It is only present on servers that are serving the root of the namespace.

/etc/init.d/rpc initialization script for NIS+

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nis_cachemgr(1M), **nisinit**(1M), **nissetup**(1M), **nslookup**(1M), **syslogd**(1M), **rpc.nisd_resolv**(1M), **rpc.nispasswdd**(1M), **nis_db**(3N), **netconfig**(4), **nisfiles**(4), **resolv.conf**(4), **ypfiles**(4), **attributes**(5), **ticots**(7D)

NAME

rpc.nisd_resolv, nisd_resolv – NIS+ service daemon

SYNOPSIS

rpc.nisd_resolv [**–v** | **–V**] [**–F** [**–C** *fd*]] [**–t** *xx*] [**–p** *yy*]

DESCRIPTION

rpc.nisd_resolv is an auxiliary process to **rpc.nisd** used to provide ypserv compatible DNS forwarding for NIS host requests. It is generally started by invoking **rpc.nisd** with the **–B** option. **rpc.nisd_resolv** can also be started independently with the following options.

OPTIONS

–F Run in foreground.

–C *fd* Use *fd* for service xprt (from **nisd**).

–v Verbose. Send output to the syslog daemon.

–V Verbose. Send output to stdout.

–t *xx* Use transport *xx*.

–p *yy* Use transient program# *yy*.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nslookup(1M), **rpc.nisd**(1M), **resolv.conf**(4), **attributes**(5)

NOTES

This command requires that the **/etc/resolv.conf** file be setup for communication with a DNS nameserver. The **nslookup** utility can be used to verify communication with a DNS nameserver. See **resolv.conf**(4) and **nslookup**(1M).

NAME

rpc.nispasswdd, nispasswdd – NIS+ password update daemon

SYNOPSIS

/usr/sbin/rpc.nispasswdd [**–a** *attempts*] [**–c** *minutes*] [**–D**] [**–g**] [**–v**]

DESCRIPTION

rpc.nispasswdd daemon is an ONC+ RPC service that services password update requests from **nispasswd**(1) and **yppasswd**(1). It updates password entries in the NIS+ **passwd** table.

rpc.nispasswdd is normally started from a system startup script after the NIS+ server (**rpc.nisd**(1M)) has been started. **rpc.nispasswdd** will determine whether it is running on a machine that is a master server for one or more NIS+ directories. If it discovers that the host is not a master server, then it will promptly exit. It will also determine if **rpc.nisd**(1M) is running in NIS(YP) compatibility mode (the **–Y** option) and will register as **yppasswdd** for NIS(YP) clients as well.

rpc.nispasswdd will syslog all failed password update attempts, which will allow an administrator to determine whether someone was trying to "crack" the passwords.

rpc.nispasswdd has to be run by a superuser.

OPTIONS

–a *attempts* Set the maximum number of attempts allowed to authenticate the caller within a password update request session. Failed attempts are **syslogd**(1M) and the request is cached by the daemon. After the maximum number of allowed attempts the daemon severs the connection to the client. The default value is set to **3**.

–c *minutes* Set the number of minutes a failed password update request should be cached by the daemon. This is the time during which if the daemon receives further password update requests for the same user and authentication of the caller fails, then the daemon will simply not respond. The default value is set to **30** minutes.

–D Debug. Run in debugging mode.

–g Generate DES credential. By default the DES credential is not generated for the user if they do not have one. By specifying this option, if the user does not have a credential, then one will be generated for them and stored in the NIS+ cred table.

–v Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon. This option is useful for debugging problems.

EXIT STATUS

0 success

1 an error has occurred.

FILES

/etc/init.d/rpc initialization script for NIS+

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

nispasswd(1), **passwd**(1), **yppasswd**(1), **rpc.nisd**(1M), **syslogd**(1M), **nsswitch.conf**(4), **attributes**(5)

NAME

rpc.rexd, rexd – RPC-based remote execution server

SYNOPSIS

/usr/sbin/rpc.rexd [**–s**]

DESCRIPTION

rpc.rexd is the Sun RPC server for remote program execution. This daemon is started by **inetd**(1M) whenever a remote execution request is made.

For non-interactive programs, the standard file descriptors are connected directly to TCP connections. Interactive programs involve pseudo-terminals, in a fashion that is similar to the login sessions provided by **rlogin**(1). This daemon may use NFS to mount file systems specified in the remote execution request.

SECURITY

rpc.rexd uses **pam**(3) for account and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **rpc.rexd**. Here is a partial **pam.conf** file with **rpc.rexd** entries for account and session management using the UNIX module.

```
rpc.rexd    account    required    /usr/lib/security/pam_unix.so.1
rpc.rexd    session    required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **rpc.rexd** service, then the entries for the "other" service will be used. **rpc.rexd** uses the **getpwuid( )** call to determine whether the given user is a legal user.

OPTIONS

–s Secure. When specified, requests must have valid DES credentials. If the request does not have a DES credential it is rejected. The default publickey credential is rejected. Only newer **on**(1) commands send DES credentials.

If access is denied with an authentication error, you may have to set your publickey with the **chkey**(1) command.

Specifying the **–s** option without presenting secure credentials will result in an error message: **Unix too weak auth (DesONly)!**

FILES

/dev/pts*n*	pseudo-terminals used for interactive mode
/etc/passwd	authorized users
/tmp_rex/rexd??????	temporary mount points for remote file systems.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

chkey(1), **on**(1), **rlogin**(1), **inetd**(1M), **pam**(3), **exports**(4), **inetd.conf**(4), **pam.conf**(4), **publickey**(4), **attributes**(5), **pam_unix**(5)

DIAGNOSTICS

Diagnostic messages are normally printed on the console, and returned to the requestor.

NOTES

Root cannot execute commands using **rexd** client programs such as **on**(1).

NAME

rpc.rstatd, rstatd – kernel statistics server

SYNOPSIS

/usr/lib/netsvc/rstat/rpc.rstatd

DESCRIPTION

rpc.rstatd is a server which returns performance statistics obtained from the kernel. **rup**(1) uses **rpc.rstatd** to collect the *uptime* information that it displays.

rpc.rstatd is an RPC service.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

rup(1), **inetd**(1M), **services**(4), **attributes**(5)

NAME

rpc.rusersd, rusersd – network username server

SYNOPSIS

/usr/lib/netsvc/rusers/rpc.rusersd

DESCRIPTION

rpc.rusersd is a server that returns a list of users on the host. The **rpc.rusersd** daemon may be started by **inetd**(1M) or **listen**(1M).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M), **listen**(1M), **pmadm**(1M), **sacadm**(1M), **attributes**(5)

NAME

rpc.rwalld, rwalld – network rwall server

SYNOPSIS

/usr/lib/netsvc/rwall/rpc.rwalld

DESCRIPTION

rpc.rwalld is a server that handles **rwall**(1M) requests. It is implemented by calling **wall**(1M) on all the appropriate network machines. The **rpc.rwalld** daemon may be started by **inetd**(1M) or **listen**(1M).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M), **listen**(1M), **rwall**(1M), **wall**(1M), **attributes**(5)

NAME

rpc.sprayd, sprayd – spray server

SYNOPSIS

/usr/lib/netsvc/spray/rpc.sprayd

DESCRIPTION

rpc.sprayd is a server that records the packets sent by **spray**(1M). The **rpc.sprayd** daemon may be started by **inetd**(1M) or **listen**(1M).

The service provided by **rpc.sprayd** is not useful as a networking benchmark as it uses unreliable connectionless transports, (udp for example). It can report a large number of packets dropped when the drops were caused by the program sending packets faster than they can be buffered locally (before the packets get to the network medium).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M) **listen**(1M), **pmadm**(1M), **sacadm**(1M), **spray**(1M), **attributes**(5)

NAME

rpc.yppasswdd, yppasswdd – server for modifying NIS password file

SYNOPSIS

/usr/lib/netsvc/yp/rpc.yppasswdd [–**D** *directory*] [–**nogecos**] [–**noshell**] [–**nopw**]
[–**m** *argument1 argument2 . . .*]

/usr/lib/netsvc/yp/rpc.yppasswdd [*passwordfile* [*adjunctfile*]] [–**nogecos**] [–**noshell**]
[–**nopw**] [–**m** *argument1 argument2 . . .*]

DESCRIPTION

rpc.yppasswdd is a server that handles password change requests from **yppasswd**(1). It changes a password entry in the **passwd**, **shadow**, and **security/passwd.adjunct** files. The **passwd** and **shadow** files provide the basis for the **passwd.byname** and **passwd.byuid** maps. The **passwd.adjunct** file provides the basis for the **passwd.adjunct.byname** and **passwd.adjunct.byuid** maps. Entries in the **passwd**, **shadow** or **passwd.adjunct** files are only changed if the password presented by **yppasswd**(1) matches the encrypted password of the entry. All password files are located in the **PWDIR** directory.

If the –**D** option is given, the **passwd**, **shadow,** or **passwd.adjunct** files are located under the directory path specified with –**D**.

If the –**noshell**, –**nogecos** or –**nopw** options are given, these fields may not be changed remotely using **chfn**, **chsh**, or **passwd**(1).

If the –**m** option is given, a **make**(1) is performed in **/var/yp** after any of the **passwd**, **shadow**, or **passwd.adjunct** files are modified. Any arguments following the flag are passed to **make**.

The second of the listed syntaxes is provided only for backward compatibility. If the second syntax is used the **passwordfile** is the full pathname of the password file and **adjunctfile** is the full pathname of the optional **passwd.adjunct** file. If a shadow file is found in the same directory as **passwordfile** the **shadowfile** is used as described above. Use of this syntax and the discovery of a **shadowfile** file generates diagnostic output. The daemon, however, starts normally.

The first and second syntaxes are mutually exclusive. You cannot specify the full pathname of the **passwd**, **passwd.adjunct** files and use the –**D** option at the same time.

The daemon is started automatically on the master server of the **passwd** map by the **/etc/init.d/rpc** script (see **makedbm**(1M))**.**

The server does not insist on the presence of a **shadow** file unless there is no –**D** option present or the directory named with the –**D** option is **/etc**. In addition, a **passwd.adjunct** file is not necessary. If the –**D** option is given, the server attempts to find a **passwd.adjunct** file in the **security** subdirectory of the named directory. For example, in the presence of ‘‘–**D /var/yp**’’ the server checks for a ‘‘**/var/yp/security/passwd.adjunct**’’ file.

If there is only a **passwd** file, then the encrypted password is expected in the second field. If there is a **passwd** and a **passwd.adjunct** file, the encrypted password is expected in the second field of the adjunct file with *##username* in the second field of the **passwd** file. If all three files are in use, the encrypted password is expected in the adjunct file, the

shadow file contains the *##username* and the **passwd** file contains the usual ‘‘x’’ in the second field. Any deviation causes a password update to fail.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

SEE ALSO

make(1), **passwd**(1), **yppasswd**(1), **inetd**(1M), **ypmake**(1M), **passwd**(4), **shadow**(4), **ypfiles**(4), **attributes**(5)

NOTES

If **make** has not been installed and the **−m** option is given, the daemon outputs a warning and proceeds, effectively ignoring the **−m** flag.

When using the **−D** option, you should make sure that the **PWDIR** of the **/var/yp/Makefile** is set accordingly.

The second listed syntax is supplied only for backwards compatibility and may be removed in a future release of this daemon.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

rpc.ypupdated, ypupdated – server for changing NIS information

SYNOPSIS

/usr/lib/netsvc/yp/rpc.ypupdated [**–is**]

DESCRIPTION

ypupdated is a daemon that updates information in the Network Information Service (NIS). **ypupdated** consults the **updaters**(4) file in the **/var/yp** directory to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (insecure).

OPTIONS

–i Accept RPC calls with the insecure AUTH_UNIX credentials. This allows programmatic updating of the NIS maps in all networks.

–s Accept only calls authenticated using the secure RPC mechanism (AUTH_DES authentication). This disables programmatic updating of the NIS maps unless the network supports these calls.

FILES

/var/yp/updaters Configuration file for **rpc.updated** command.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	.B SUNWypu

SEE ALSO

keyserv(1M), **updaters**(4), **attributes**(5)

System Administration Guide
Network Interfaces Programmer's Guide

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

rpld – x86 Network Booting RPL (Remote Program Load) Server

SYNOPSIS

/usr/sbin/rpld [**–fdDMblgz**] *interface*
/usr/sbin/rpld –a [**–fdDMblgz**]

DESCRIPTION

The RPL server provides network booting functionality to x86 clients by listening to boot requests from them according to the RPL protocol specifications. Boot requests can be generated by clients using the boot floppy supplied in the x86 distribution. Once the request has been received, the server validates the client and adds it to its internal service list. Subsequent requests from the client to download bootfiles will result in the sending of data frames from the server to the client specifying where to load the boot program in memory. When all the bootfiles have been downloaded, the server specifies where to start execution to initiate the boot process.

In the first synopsis, the interface parameter names the network interface upon which **rpld** is to listen for requests. For example:

```
/usr/sbin/rpld /dev/le0
/usr/sbin/rpld /dev/smc0
```

In the second synopsis, **rpld** locates all of the network interfaces present on the system and starts a daemon process for each one.

The server starts by reading the default configuration file, or an alternate configuration file if one is specified. If no configuration file can be found, internal default values will be used. Alternatively, command line options are available to override any of the values in the configuration file. After the configuration options are set, it then opens the network interface as specified in the command line and starts listening to RPL boot requests.

Network boot x86 clients have to have information pre-configured on a server for the RPL server to validate and serve them. This involves putting configuration information in both the **ethers**(4) and the **bootparams**(4) databases. The **ethers** database contains a translation from the physical node address to the IP address of the clients and is normally used by the RARP server. The **bootparams** database stores all other information needed for booting off this client, such as the number of bootfiles and the file names of the various boot components. Both databases can be looked up by the RPL server through NIS. See the sub-section **Client Configuration** for information on how to set up these databases.

To assist in the administration and maintenance of the network boot activity, there are two run-time signals that the server will accept to change some run-time parameters and print out useful status information. See the sub-section **Signals** for details.

The RPL server is not limited to the ability to boot only x86 clients. If properly configured, the server should be able to download any bootfiles to the clients.

Client Configuration

The following configuration information is specific to booting x86 clients.

In order to allow clients to boot x86 from across the network, the client's information has to be pre-configured in two databases: **ethers**(4) and **bootparams**(4). Both databases can be accessed through NIS. Refer to *Solaris Advanced Installation Guide* for information on how to configure a diskless x86 client. The discussion contained in the rest of this section is provided for your information only and should not be performed manually.

The **ethers** database contains a translation table to convert the physical node address to the IP address of the client. Therefore, an IP address must be assigned to the client (if this has not been done already), the node address of the client must be obtained, and then this information needs to be entered in the **ethers** database.

The bulk of the configuration is done in the **bootparams** database. This is a free-format database that essentially contains a number of keyword-value string pairs. A number of keywords have been defined for specific purposes, like the **bootparams** RPC in **bootparamd**(1M). Three more keywords have been defined for the RPL server. They are **numbootfiles**, **bootfile**, and **bootaddr**. All three keywords must be in lowercase letters with no spaces before or after the equals symbol following the keyword.

numbootfiles Specifies the number of files to be downloaded to the network boot client. The format of this option is:

numbootfiles=n

Always use **numbootfiles=3** to boot x86 across the network.

bootfile Specifies the path name of the bootfile to be downloaded and where in memory to start loading the bootfile. A complete path name should be used. For example, assuming the client's IP address is 129.181.32.15:

```
bootfile=/rplboot/129.181.32.15.hw.com:45000
bootfile=/rplboot/129.181.32.15.glue.com:35000
bootfile=/rplboot/129.181.32.15.inetboot=8000
```

The path name following the equals symbol specifies the bootfile to be downloaded, and the hex address following the colon (:) is the absolute address of the memory location to start loading that bootfile. These addresses should be in the range of 7c00 to a0000 (i.e., the base 640K range excluding the interrupt vector and BIOS data areas). Address 45000 for this **hw.com** bootfile is also a suggested value and if possible should not be changed. The address of 35000 for **glue.com** is a suggested value that, if possible, should not be changed. The address of 8000 for **inetboot** is an absolute requirement and should never be changed.

These files, when created following the procedures in the *Solaris Advanced Installation Guide*, are actually symbolic links to to the real file to be downloaded to the client. **hw.com** is linked to a special driver that corresponds to the network interface card of the client. **glue.com** and **inetboot** are generic to all network boot clients.

The order of these bootfile lines is not significant, but because problems have been found with certain boot PROMs, it is highly recommended that the bootfile lines be ordered in descending order of the load addresses.

bootaddr

The absolute address in memory to start executing after all the bootfiles have been downloaded. This address should always correspond to the address where **glue.com** is being loaded. If possible, always use:

bootaddr=35000

OPTIONS

–**f** config filename.

Use this to specify a configuration file name other than the system default **/etc/rpld.conf** file.

–**d** debug level.

Specify a level of **0** if you do not want any error or warning messages to be generated, or a level from **1** to **9** for increasing amounts of messages. This option corresponds to the *DebugLevel* setting in the configuration file. The default value is **0**. Note that it is best to limit the level to **8** or below; use of level **9** may generate so many debug messages that the performance of the RPL server may be impacted.

–**D** debug destination.

Specify **0** to send error or warning messages to standard output, **1** to syslogd, and **2** to the log file. This option corresponds to the *DebugDest* setting in the configuration file. The default value is **2**.

–**M** maximum clients.

Specify the maximum number of simultaneous network boot clients to be served. This option corresponds to the *MaxClients* setting in the configuration file. A value of –**1** means unlimited, and the actual number will depend on available system resources. The default value is –**1**.

–**b** background mode.

Specify **1** to run the server in the background and relinquish the controlling terminal, or **0** to run in the foreground without relinquishing the controlling terminal. This option corresponds to the *BackGround* setting in the configuration file. If you have specified that the error or warning messages be sent to standard output in the configuration file or by using the –**D** option above, the server cannot be run in background mode. Doing so will cause the server to exit after announcing the error.

–**l** log filename.

Specify an alternate log file name to hold the error or warning messages in connection with the –**D 2** option or the configuration file *DebugDest = 2* setting. This option corresponds to the *LogFile* setting in the configuration file. The default is **/var/spool/rpld.log**.

–s start delay count.

This option corresponds to the *StartDelay* setting in the configuration file. Specify the number of delay units between outgoing data frames sent to clients to avoid retransmission requests from them. Using the LLC type **1** protocol, data transfer is a one-way, best-effort delivery mechanism. The server, without any type of delay mechanism, can overrun the client by sending data frames too quickly. Therefore, a variable delay is built into the server to limit the speed of sending data to the clients, thus avoiding the clients sending back retransmission requests. This value should be machine environment specific. If you have a fast server machine but slow client machines, you may want to set a large start delay count. If you have comparable server and client machines, the delay count may be set to **1**. The delay is only approximate and should not be taken as an accurate measure of time. There is no specific correlation between the delay unit and the actual time of delay. The default value is **20**.

–g delay granularity.

This corresponds to the *DelayGran* setting in the configuration file. If retransmission requests from clients do occur, the delay granularity factor will be used to adjust the delay count for this client upwards or downwards. If the retransmission request is caused by data overrun, the delay count will be incremented by delay granularity units to increase the delay between data frames. If the retransmission request is caused by sending data too slowly, this will be used to adjust the delay count downwards to shorten the delay. Eventually the server will settle at the delay count value that works best with the speed of the client and no retransmission request will be needed. The default value is **2**.

–z frame size.

This option corresponds to the *FrameSize* setting in the configuration file. This specifies the size of the data frames used to send data to the clients. This is limited by the underlying physical medium. For **ethernet/802.3**, the maximum physical frame size is 1500 octets. The default value is **1500**. Note that the protocol overhead of LLC1 and RPL is 32 octets, resulting in a maximum data length of 1468 octets.

Signals

The RPL server accepts two signals to change run-time parameters and display status information, respectively:

HANGUP

This will cause the RPL server to reread the default configuration file **/etc/rpld.conf** or an alternate configuration file if one is specified when the server is started. New values of certain parameters can be used immediately, such as *DebugLevel*, *DebugDest*, *LogFile*, *DelayGran*, and *FrameSize*. For *MaxClients*, if the server is already serving more than the new value, the server will not accept additional boot requests until the number has fallen below the *MaxClients* parameter. For *StartDelay*, this will only affect new boot requests. All the existing delay counts for the various clients in service will not be affected. Finally, the *BackGround* parameter will have no effect once the server has been running. You cannot change the mode of service without first killing the server and then restarting it.

USR1 This signal will cause the server to dump all the parameter values and the status of each individual boot client to the destination specified by *DebugDest*.

FILES

/usr/sbin/rpld
/etc/rpld.conf
/var/spool/rpld.log
/etc/ethers
/etc/bootparams
/rplboot

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWcsu

SEE ALSO

bootparamd(1M), **in.rarpd**(1M), **bootparams**(4), **ethers**(4), **nsswitch.conf**(4), **rpld.conf**(4), **attributes**(5)

Solaris Advanced Installation Guide

NAME

rquotad – remote quota server

SYNOPSIS

/usr/lib/nfs/rquotad

DESCRIPTION

rquotad is an **rpc**(4) server which returns quotas for a user of a local file system which is mounted by a remote machine over the NFS. The results are used by **quota**(1M) to display user quotas for remote file systems. The **rquotad** daemon is normally invoked by **inetd**(1M).

FILES

quotas	quota file at the file system root

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M), **quota**(1M), **rpc**(4), **services**(4), **attributes**(5)

Solaris Advanced Installation Guide

NAME

rsh, restricted_shell – restricted shell command interpreter

SYNOPSIS

/usr/lib/rsh [**–acefhiknprstuvx**] [*argument*. . .]

DESCRIPTION

rsh is a limiting version of the standard command interpreter **sh**, used to restrict logins to execution environments whose capabilities are more controlled than those of **sh** (see **sh**(1) for complete description and usage).

When the shell is invoked, it scans the environment for the value of the environmental variable, **SHELL.** If it is found and **rsh** is the file name part of its value, the shell becomes a restricted shell.

The actions of **rsh** are identical to those of **sh**, except that the following are disallowed:

changing directory (see **cd**(1)),
setting the value of **$PATH,**
specifying path or command names containing **/**,
redirecting output (**>** and **>>**).

The restrictions above are enforced after *.profile* is interpreted.

A restricted shell can be invoked in one of the following ways:

(1) **rsh** is the file name part of the last entry in the **/etc/passwd** file (see **passwd**(4));

(2) the environment variable **SHELL** exists and **rsh** is the file name part of its value; the environment variable **SHELL** needs to be set in the **.login** file;

(3) the shell is invoked and **rsh** is the file name part of argument 0;

(4) the shell is invoke with the **–r** option.

When a command to be executed is found to be a shell procedure, **rsh** invokes **sh** to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the *.profile* (see **profile**(4)) has complete control over user actions by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands (that is, **/usr/rbin**) that can be safely invoked by a restricted shell. Some systems also provide a restricted editor, **red**.

EXIT CODES

Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cd(1), **intro**(1), **login**(1), **sh**(1), **exec**(2), **passwd**(4), **profile**(4), **attributes**(5)

NOTES

The restricted shell, **/usr/lib/rsh**, should not be confused with the remote shell, **/usr/bin/rsh**, which is documented in **rsh**(1).

NAME

rtc – provide all real-time clock and GMT-lag management

SYNOPSIS

rtc [**–c**] [**–z** *zone-name*]

DESCRIPTION

The **/usr/sbin/rtc** command is used to reconcile the difference in the way time is established between UNIX systems, which utilize Greenwich Mean Time (GMT), and MS-DOS systems, which utilize local time. When used with no arguments, **rtc** displays the currently configured time zone string which was recorded when **rtc –z** *zone-name* ws last run.

The **rtc** command is not normally run from a shell prompt; it is generally invoked by the system. Commands such as **date**(1) and **rdate**(1M), which are used to set the time on a system, invoke **/usr/sbin/rtc –c** to ensure that daylight savings time (DST) is corrected for properly.

OPTIONS

–c This option checks for DST and makes corrections if necessary. It is normally run once a day by a *cron* job.

If there is no RTC time zone or **/etc/rtc_config** file, this option will do nothing.

–z *zone-name* This option, which is normally run by the system at software installation time, is used to specify the time zone in which the RTC is to be maintained. It updates the configuration file **/etc/rtc_config** with the name of the specified zone and the current GMT lag for that zone. If there is an existing **rtc_config** file, this command will update it. If not, this command will create it.

FILES

/usr/sbin/rtc The command used to provide all real-time clock and GMT lag management.

/etc/rtc_config The data file used to record the time zone and GMT lag. This file is completely managed by **/usr/sbin/rtc**, and it is read by the kernel.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWcsu

SEE ALSO

date(1), **rdate**(1M), **attributes**(5)

NAME

runacct – run daily accounting

SYNOPSIS

/usr/lib/acct/runacct [*mmdd* [*state*]]

DESCRIPTION

runacct is the main daily accounting shell procedure. It is normally initiated using **cron**. **runacct** processes connect, fee, disk, and process accounting files. It also prepares summary files for **prdaily** or billing purposes. **runacct** is distributed only to source code licensees.

runacct takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into **active**. When an error is detected, a message is written to **/dev/console**, mail (see **mail**(1)) is sent to **root** and **adm**, and **runacct** terminates. **runacct** uses a series of lock files to protect against re-invocation. The files **lock** and **lock1** are used to prevent simultaneous invocation, and **lastdate** is used to prevent more than one invocation per day.

runacct breaks its processing into separate, restartable *states* using **statefile** to remember the last *state* completed. It accomplishes this by writing the *state* name into **statefile**. **runacct** then looks in **statefile** to see what it has done and to determine what to process next. *states* are executed in the following order:

SETUP Move active accounting files into working files.

WTMPFIX Verify integrity of **wtmp** file, correcting date changes if necessary.

CONNECT Produce connect session records in **tacct.h** format.

PROCESS Convert process accounting records into **tacct.h** format.

MERGE Merge the connect and process accounting records.

FEES Convert output of **chargefee** into **tacct.h** format, merge with connect, and process accounting records.

DISK Merge disk accounting records with connect, process, and fee accounting records.

MERGETACCT
Merge the daily total accounting records in **daytacct** with the summary total accounting records in **/var/adm/acct/sum/tacct**.

CMS Produce command summaries.

USEREXIT Any installation dependent accounting programs can be included here.

CLEANUP Clean up temporary files and exit.

To restart **runacct** after a failure, first check the **active** file for diagnostics, then fix any corrupted data files, such as **pacct** or **wtmp**. The **lock**, **lock1**, and **lastdate** files must be removed before **runacct** can be restarted. The argument *mmdd* is necessary if **runacct** is being restarted. *mmdd* specifies the month and day for which **runacct** will rerun the accounting. The entry point for processing is based on the contents of **statefile**; to override this, include the desired *state* on the command line to designate where processing should begin.

EXAMPLES

To start **runacct**:

```
example% nohup runacct 2> /var/adm/acct/nite/fd2log &
```

To restart **runacct**:

```
example% nohup runacct 0601 2>> /var/adm/acct/nite/fd2log &
```

To restart **runacct** at a specific *state*:

```
example% nohup runacct 0601 MERGE 2>> /var/adm/acct/nite/fd2log &
```

FILES

/var/adm/wtmp
/var/adm/pacct*incr*
/var/adm/acct/nite/active
/var/adm/acct/nite/daytacct
/var/adm/acct/nite/lock
/var/adm/acct/nite/lock1
/var/adm/acct/nite/lastdate
/var/adm/acct/nite/statefile

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

acctcom(1), **mail**(1), **acct**(1M), **acctcms**(1M), **acctcon**(1M), **acctmerg**(1M), **acctprc**(1M), **acctsh**(1M), **cron**(1M), **fwtmp**(1M), **acct**(2), **acct**(4), **utmp**(4), **attributes**(5)

NOTES

It is not recommended to restart **runacct** in the **SETUP** *state*. Run **SETUP** manually and restart using:

runacct *mmdd* **WTMPFIX**

If **runacct** failed in the **PROCESS** *state*, remove the last **ptacct** file because it will not be complete.

The **runacct** command can process a maximum of

6000 distinct sessions
1000 distinct terminal lines
2000 distinct login names

during a single invocation of the command. If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

NAME

rwall – write to all users over a network

SYNOPSIS

/usr/sbin/rwall *hostname* . . .
/usr/sbin/rwall –**n** *netgroup* . . .
/usr/sbin/rwall –**h** *hostname* –**n** *netgroup*

DESCRIPTION

rwall reads a message from standard input until EOF. It then sends this message, preceded by the line:

Broadcast Message . . .

to all users logged in on the specified host machines. With the –**n** option, it sends to the specified network groups.

OPTIONS

–**n** *netgroup* Send the broadcast message to the specified network groups.

–**h** *hostname* Specify the *hostname*, the name of the host machine.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

inetd(1M), **listen**(1M), **pmadm**(1M), **sacadm**(1M), **wall**(1M), **attributes**(5)

NOTES

The timeout is fairly short to allow transmission to a large group of machines (some of which may be down) in a reasonable amount of time. Thus the message may not get through to a heavily loaded machine.

NAME

sac – service access controller

SYNOPSIS

sac –**t** *sanity_interval*

/usr/lib/saf/sac

DESCRIPTION

The Service Access Controller (SAC) is the overseer of the server machine. It is started when the server machine enters multiuser mode. The SAC performs several important functions as explained below.

Customizing the SAC *environment*. When **sac** is invoked, it first looks for the per-system configuration script **/etc/saf/_sysconfig**. **sac** interprets **_sysconfig** to customize its own environment. The modifications made to the SAC environment by **_sysconfig** are inherited by all the children of the SAC. This inherited environment may be modified by the children.

Starting port monitors. After it has interpreted the **_sysconfig** file, the **sac** reads its administrative file **/etc/saf/_sactab**. **_sactab** specifies which port monitors are to be started. For each port monitor to be started, **sac** forks a child (see **fork**(2)) and creates a **utmp** entry with the *type* field set to **LOGIN_PROCESS**. Each child then interprets its per-port monitor configuration script **/etc/saf/***pmtag***/_config**, if the file exists. These modifications to the environment affect the port monitor and will be inherited by all its children. Finally, the child process **exec**s the port monitor, using the command found in the **_sactab** entry. (See **sacadm**; this is the command given with the –**c** option when the port monitor is added to the system.)

Polling port monitors to detect failure. The –**t** option sets the frequency with which **sac** polls the port monitors on the system. This time may also be thought of as half of the maximum latency required to detect that a port monitor has failed and that recovery action is necessary.

Administrative functions. The Service Access Controller represents the administrative point of control for port monitors. Its administrative tasks are explained below.

When queried (**sacadm** with either –**l** or –**L**), the Service Access Controller returns the status of the port monitors specified, which **sacadm** prints on the standard output. A port monitor may be in one of six states:

ENABLED The port monitor is currently running and is accepting connections. See **sacadm**(1M) with the –**e** option.

DISABLED The port monitor is currently running and is not accepting connections. See **sacadm** with the –**d** option, and see **NOTRUNNING**, below.

STARTING The port monitor is in the process of starting up. **STARTING** is an intermediate state on the way to **ENABLED** or **DISABLED**.

FAILED The port monitor was unable to start and remain running.

STOPPING The port monitor has been manually terminated but has not completed its shutdown procedure. **STOPPING** is an intermediate state on the way to **NOTRUNNING**.

NOTRUNNING The port monitor is not currently running. (See **sacadm** with **–k**.) This is the normal "not running" state. When a port monitor is killed, all ports it was monitoring are inaccessible. It is not possible for an external user to tell whether a port is not being monitored or the system is down. If the port monitor is not killed but is in the **DISABLED** state, it may be possible (depending on the port monitor being used) to write a message on the inaccessible port telling the user who is trying to access the port that it is disabled. This is the advantage of having a **DISABLED** state as well as the **NOTRUNNING** state.

When a port monitor terminates, the SAC removes the **utmp** entry for that port monitor.

The SAC receives all requests to enable, disable, start, or stop port monitors and takes the appropriate action.

The SAC is responsible for restarting port monitors that terminate. Whether or not the SAC will restart a given port monitor depends on two things:

- The restart count specified for the port monitor when the port monitor was added by **sacadm**; this information is included in **/etc/saf/***pmtag***/_sactab**.
- The number of times the port monitor has already been restarted.

SECURITY

sac uses **pam**(3) for session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the session management module to be used for **sac**. Here is a partial **pam.conf** file with entries for **sac** using the UNIX session management module.

```
sac     session     required     /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **sac** service, then the entries for the "other" service will be used.

OPTIONS

–t *sanity_interval* Sets the frequency (*sanity_interval*) with which **sac** polls the port monitors on the system.

FILES

/etc/saf/_sactab
/etc/saf/_sysconfig
/var/adm/utmp
/var/saf/_log

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pmadm(1M), **sacadm**(1M), **fork**(2) **pam**(3), **pam.conf**(4), **attributes**(5), **pam_unix**(5)

NAME

sacadm – service access controller administration

SYNOPSIS

sacadm –a –p *pmtag* **–t** *type* **–c** *cmd* **–v** *ver* **[–f dx] [–n** *count***] [–y** *comment***] [–z** *script***]**
sacadm –r –p *pmtag*
sacadm –s –p *pmtag*
sacadm –k –p *pmtag*
sacadm –e –p *pmtag*
sacadm –d –p *pmtag*
sacadm –l [–p *pmtag* **| –t** *type***]**
sacadm –L [–p *pmtag* **| –t** *type***]**
sacadm –g –p *pmtag* **[–z** *script***]**
sacadm –G [–z *script***]**
sacadm –x [–p *pmtag***]**

DESCRIPTION

sacadm is the administrative command for the upper level of the Service Access Facility hierarchy (port monitor administration). **sacadm** performs the following functions:

- adds or removes a port monitor
- starts or stops a port monitor
- enables or disables a port monitor
- installs or replaces a per-system configuration script
- installs or replaces a per-port monitor configuration script
- prints requested port monitor information

Requests about the status of port monitors (–**l** and –**L**) and requests to print per-port monitor and per-system configuration scripts (–**g** and –**G** without the –**z** option) may be executed by any user on the system. Other **sacadm** commands may be executed only by the super-user.

OPTIONS

–**a** Add a port monitor. When adding a port monitor, **sacadm** creates the supporting directory structure in **/etc/saf** and **/var/saf** and adds an entry for the new port monitor to **/etc/saf/_sactab**. The file **_sactab** already exists on the delivered system. Initially, it is empty except for a single line, which contains the version number of the Service Access Controller.

Unless the command line that adds the new port monitor includes the –**f** option with the –**x** argument, the new port monitor will be started. Because of the complexity of the options and arguments that follow the –**a** option, it may be convenient to use a command script or the menu system to add port monitors.

–**c** *cmd* Execute the command string *cmd* to start a port monitor. The –**c** option may be used only with a –**a**. A –**a** option requires a –**c**.

–**d** Disable the port monitor *pmtag*.

–e Enable the port monitor *pmtag*.

–f dx The –**f** option specifies one or both of the following two flags which are then included in the flags field of the **_sactab** entry for the new port monitor. If the –**f** option is not included on the command line, no flags are set and the default conditions prevail. By default, a port monitor is started. A –**f** option with no following argument is illegal.

- **d** Do not enable the new port monitor.
- **x** Do not start the new port monitor.

–g The –**g** option is used to request output or to install or replace the per-port monitor configuration script **/etc/saf/***pmtag***/_config**. –**g** requires a –**p** option. The –**g** option with only a –**p** option prints the per-port monitor configuration script for port monitor *pmtag*. The –**g** option with a –**p** option and a –**z** option installs the file **script** as the per-port monitor configuration script for port monitor *pmtag*. Other combinations of options with –**g** are invalid.

–G The –**G** option is used to request output or to install or replace the per-system configuration script **/etc/saf/_sysconfig**. The –**G** option by itself prints the per-system configuration script. The –**G** option in combination with a –**z** option installs the file **script** as the per-system configuration script. Other combinations of options with a –**G** option are invalid.

–k Stop port monitor *pmtag*.

–l The –**l** option is used to request port monitor information. The –**l** by itself lists all port monitors on the system. The –**l** option in combination with the –**p** option lists only the port monitor specified by *pmtag*. A –**l** in combination with the –**t** option lists all port monitors of type *type*. Any other combination of options with the –**l** option is invalid.

–L The –**L** option is identical to the –**l** option except that the output appears in a condensed format.

–n *count* Set the restart count to *count*. If a restart count is not specified, count is set to **0**. A count of **0** indicates that the port monitor is not to be restarted if it fails.

–p *pmtag* Specifies the tag associated with a port monitor.

–r Remove port monitor *pmtag*. **sacadm** removes the port monitor entry from **/etc/saf/_sactab**. If the removed port monitor is not running, then no further action is taken. If the removed port monitor is running, the Service Access Controller (SAC) sends it **SIGTERM** to indicate that it should shut down. Note that the port monitor's directory structure remains intact.

–s Start a port monitor. The SAC starts the port monitor *pmtag*.

–t *type* Specifies the port monitor type.

–v *ver* Specifies the version number of the port monitor. This version number may be given as

–v `***pmspec* **–V`

where *pmspec* is the special administrative command for port monitor *pmtag*. This special command is **ttyadm** for **ttymon** and **nlsadmin** for **listen**. The version stamp of the port monitor is known by the command and is returned when *pmspec* is invoked with a **–V** option.

–x The **–x** option by itself tells the SAC to read its database file (**_sactab**). The **–x** option with the **–p** option tells port monitor *pmtag* to read its administrative file.

–y *comment* Include *comment* in the **_sactab** entry for port monitor *pmtag*.

–z *script* Used with the **–g** and **–G** options to specify the name of a file that contains a configuration script. With the **–g** option, *script* is a per-port monitor configuration script; with **–G** it is a per-system configuration script. Modifying a configuration script is a three-step procedure. First a copy of the existing script is made (**–g** or **–G**). Then the copy is edited. Finally, the copy is put in place over the existing script (**–g** or **–G** with **–z**).

OUTPUT

If successful, **sacadm** will exit with a status of **0**. If **sacadm** fails for any reason, it will exit with a nonzero status. Options that request information will write the information on the standard output. In the condensed format (**–L**), port monitor information is printed as a sequence of colon-separated fields; empty fields are indicated by two successive colons. The standard format (**–l**) prints a header identifying the columns, and port monitor information is aligned under the appropriate headings. In this format, an empty field is indicated by a hyphen. The comment character is **#**.

EXAMPLES

The following command line adds a port monitor. The port monitor tag is **npack**; its type is **listen**; if necessary, it will restart three times before failing; its administrative command is **nlsadmin**; and the configuration script to be read is in the file **script**:

```
sacadm –a –p npack –t listen –c /usr/lib/saf/listen npack
    –v `nlsadmin –V` –n 3 –z script
```

Remove a port monitor whose tag is **pmtag**:

```
sacadm –r –p pmtag
```

Start the port monitor whose tag is **pmtag**:

```
sacadm –s –p pmtag
```

Stop the port monitor whose tag is **pmtag**:

```
sacadm –k –p pmtag
```

Enable the port monitor whose tag is **pmtag**:

sacadm −e −p pmtag

Disable the port monitor whose tag is **pmtag**:

sacadm −d −p pmtag

List status information for all port monitors:

sacadm −l

List status information for the port monitor whose tag is **pmtag**:

sacadm −l −p pmtag

List the same information in condensed format:

sacadm −L −p pmtag

List status information for all port monitors whose type is **listen**:

sacadm −l −t listen

Replace the per-port monitor configuration script associated with the port monitor whose tag is **pmtag** with the contents of the file **file.config**:

sacadm −g −p pmtag −z file.config

FILES

/etc/saf/_sactab
/etc/saf/_sysconfig
/etc/saf/*pmtag***/_config**

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pmadm(1M), **sac**(1M), **doconfig**(3N), **attributes**(5)

NAME

sadmind – distributed system administration daemon

SYNOPSIS

sadmind [**–c** *keywords*] [**–i** *secs*] [**–l** [*logfile*]] [**–O** *OW_path_name*]
[**–S** *security_level*] [**–v**]

DESCRIPTION

sadmind is the daemon used by **Solstice AdminSuite** applications to perform distributed system administration operations.

The **sadmind** daemon is started automatically by the **inetd** daemon whenever a request to invoke an operation is received. The **sadmind** daemon process continues to run for 15 minutes after the last request is completed, unless a different idle-time is specified with the –**i** command line option. The **sadmind** daemon may be started independently from the command line, for example, at system boot time. In this case, the –**i** option has no effect; **sadmind** continues to run, even if there are no active requests.

The **sadmind** daemon process can be configured to write tracing information into a log file by specifying the –**c** and –**l** command line options. The –**c** option specifies a comma-separated list of keywords indicating the types of information to be logged. The following keywords may be useful to you as an administrator:

System-Info Includes messages about when the **sadmind** daemon was started and stopped.

Requests Includes messages about which operations **sadmind** invoked and when.

Errors Includes messages about errors that occurred during the daemon execution.

* Includes all possible log messages.

The –**l** option enables logging and optionally specifies the path and file name of the log file. If no log file is specified, the default log file **/var/adm/admin.log** is used.

OPTIONS

The following command line options may be useful to you as an administrator when starting the **sadmind** daemon:

–**c** *keywords* Specify the types of information to be logged as a comma-separated list of keywords. The default is to log all types of messages.

–**i** *secs* Specify the number of seconds for **sadmind** to stay up after the last request is completed. The default is 15 minutes (900 seconds). If *secs* is 0 or over 10,000,000, **sadmind** stays up forever. –**i** only applies when **sadmind** is started by the **inetd** daemon. You may want **sadmind** to run permanently (or for extended durations) on systems that are frequently administered by applications using **sadmind** (for example, a server managed through **Host Manager**) to improve application performance.

–l[*logfile*] Enable logging and optionally define the path name to the distributed system administration log file. The default log file is:

/var/adm/admin.log

–O *OW_path_name* Define the path name to the OpenWindows home directory. If this option is not specified, the **sadmind** daemon will use the OpenWindows home directory defined in the **OPENWINHOME** environment variable, if defined; the home directory specified in the **/etc/OPENWINHOME** file, if it exists; or the default directory **/usr/openwin**. When the **sadmind** daemon is started by the **inetd** daemon, the environment variable **OPENWINHOME** is typically not defined. If the OpenWindows home directory is not one of the path names specified (**/usr/openwin** or in the file **/etc/OPENWINHOME**), the –**O** option must be added to the **sadmind** entry in the **inetd.conf**(4) configuration file.

–S *security_level* Define the level of security to be used by the **sadmind** daemon when checking a client's right to perform an operation on the server system. Security level specifies the authentication mechanism used to provide and check the client's identity. The client's identity must be authenticated by the specified mechanism for **sadmind** to accept his or her request. The system-wide authentication requirements set by the security level may take precedence over any operation-specific requirements. Consequently, the security level can be used system-wide to ensure that all operations meet minimum authentication requirements, regardless of the requirements assigned specifically to an operation. In addition, the security level determines whether **sadmind** will perform authorization access control checking.

Security level may be one of the following:

0 Set authentication type to **NONE**. All clients' user and group identities are set to the **nobody** identity by **sadmind** (see *Solstice AdminSuite 2.1 User's Guide*). If access is granted to **nobody**, **sadmind** executes the operation. Use this level only for testing.

1 Set authentication type to **WEAK**. Clients' user and group identities are set by **sadmind** from their authentication credentials. Client identities are accepted by **sadmind** when they have satisfied either **AUTH_SYS** or **AUTH_DES** authentication mechanisms. The authenticated client identity is checked by **sadmind** for authorization to execute the operation. If an operation calls for a stronger security level, **sadmind** demotes the user identity to **nobody**, and then checks whether **nobody** is authorized to execute the operation. Since **AUTH_SYS** client credentials are easily forged, this level should be used only in relatively secure environments. No check is done that the user ID of the client represents the same user on the server system as on the client system. It is assumed that user and group identities are set up consistently on the network. This security level is the default.

2 Set authentication type to **STRONG**. Clients' user and group identities are set by **sadmind** from their authentication credential mappings (effectively, user and group IDs from **netid.byname** for **NIS**, or **cred** table for **NIS+**). Client identities are accepted by **sadmind** only when they have satisfied the **AUTH_DES** authentication mechanism. The **sadmind** daemon checks whether the client identity is authorized to execute the operation. This level provides the most secure environment for executing distributed administration operations. It overrides any weaker level specific to an operation. A **DES** credential must exist for the host running the **sadmind** daemon and all administration client user identities.

–v Enable the writing of log messages to the system logger, **syslogd**. Messages logged include fatal errors encountered while attempting to start the **sadmind** daemon process and those specified by the **–c** trace message keywords.

EXAMPLES

By default, the line in **/etc/inetd.conf** that starts **sadmind** appears as follows:

```
100232/10    tli   rpc/udp   wait root /usr/sbin/sadmind sadmind
```

To make a network as secure as possible, change the line to:

```
100232/10    tli   rpc/udp   wait root /usr/sbin/sadmind sadmind –S 2
```

To minimize delays due to starting up **sadmind**, change the line to include the **–i** option:

```
100232/10    tli   rpc/udp   wait root /usr/sbin/sadmind sadmind –i 86400
```

In this example, the duration that **sadmind** remains up after the last operation request was completed is extended to 24 hours (86,400 seconds). Extending the timeout period may enhance performance on servers and workstations that frequently run or are administered by applications that use the **sadmind** daemon (for example, **Solstice AdminSuite** applications such as **Host Manager**).

FILES

/var/adm/admin.log	distributed system administration default log file
/etc/inetd.conf	internet servers database file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmfw

SEE ALSO

inetd(1M), **rpcbind**(1M), **inetd.conf**(4), **attributes**(5)

Solstice AdminSuite 2.1 User's Guide

NOTES

Whenever **inetd** fails to start **sadmind**, re-register the RPC number for **sadmind**, 100232, with **rpcbind** by sending the **inetd** process a **SIGHUP** signal:

```
example% kill –HUP pid
```

or

```
example% kill –1
```

Sometimes **inetd** does not start **sadmind** in response to system administration requests, even though the **inetd.conf** file has the correct entry for the **sadmind** daemon. This can happen when **sadmind** is started manually from the command line and takes over the previous registration of the **sadmind** RPC number, 100232, by **inetd**. When the manually-started **sadmind** daemon is terminated, the **sadmind** RPC number, 100232, is de-registered with **rpcbind**. Consequently, system administration requests are ignored by **inetd**.

NAME

saf – Service Access Facility

DESCRIPTION

The SAF generalizes the procedures for service access so that login access on the local system and network access to local services are managed in similar ways. Under the SAF, systems may access services using a variety of port monitors, including ttymon, the listener, and port monitors written expressly for a user's application. The manner in which a port monitor observes and manages access ports is specific to the port monitor and not to any component of the SAF. Users may therefore extend their systems by developing and installing their own port monitors. One of the important features of the SAF is that it can be extended in this way by users.

Relative to the SAF, a service is a process that is started. There are no restrictions on the functions a service may provide. The SAF consists of a controlling process, the service access controller (SAC), and two administrative levels corresponding to two levels in the supporting directory structure. The top administrative level is concerned with port monitor administration, the lower level with service administration. The SAC is documented in the **sac**(1M) man page. The administrative levels and associated utilities are documented in the *System Administration Guide* - *Volume II*. The requirements for writing port monitors and the functions a port monitor must perform to run under the SAF and the SAC are documented here.

Port Monitors

A port monitor is a process that is responsible for monitoring a set of homogeneous, incoming ports on a machine. A port monitor's major purpose is to detect incoming service requests and to dispatch them appropriately.

A port is an externally seen access point on a system. A port may be an address on a network (TSAP or PSAP), a hardwired terminal line, an incoming phone line, etc. The definition of what constitutes a port is strictly a function of the port monitor itself.

A port monitor performs certain basic functions. Some of these are required to conform to the SAF; others may be specified by the requirements and design of the port monitor itself. Port monitors have two main functions:

- Managing ports
- Monitoring ports for indications of activity

Port Management

The first function of a port monitor is to manage a port. The actual details of how a port is managed are defined by the person who defines the port monitor. A port monitor is not restricted to handling a single port; it may handle multiple ports simultaneously.

Note: Some examples of port management are setting the line speed on incoming phone connections, binding an appropriate network address, reinitializing the port when the service terminates, outputting a prompt, etc.

Activity Monitoring

The second function of a port monitor is to monitor the port or ports for which it is responsible for indications of activity. Two types of activity may be detected.

- The first is an indication to the port monitor to take some port monitor-specific action. Pressing the break key to indicate that the line speed should be cycled is an example of a port monitor activity. Not all port monitors need to recognize and respond to the same indications. The indication used to attract the attention of the port monitor is defined by the person who defines the port monitor.
- The second is an incoming service request. When a service request is received, a port monitor must be able to determine which service is being requested from the port on which the request is received. Note that the same service may be available on more than one port.

Other Port Monitor Functions

This section briefly describes other port monitor functions.

Restricting Access to the System

A port monitor must be able to restrict access to the system without disturbing services that are still running. In order to do this, a port monitor must maintain two internal states: enabled and disabled. The port monitor starts in the state indicated by the ISTATE environment variable provided by the sac. See sac(1M) for details. Enabling or disabling a port monitor affects all ports for which the port monitor is responsible. If a port monitor is responsible for a single port, only that port will be affected. If a port monitor is responsible for multiple ports, the entire collection of ports will be affected. Enabling or disabling a port monitor is a dynamic operation: it causes the port monitor to change its internal state. The effect does not persist across new invocations of the port monitor. Enabling or disabling an individual port, however, is a static operation: it causes a change to an administrative file. The effect of this change will persist across new invocations of the port monitor.

Creating **utmp** Entries

Port monitors are responsible for creating **utmp** entries with the type field set to USER_PROCESS for services they start, if this action has been specified (that is, if **–fu** was specified in the **pmadm** command line that added the service). These **utmp** entries may in turn be modified by the service. When the service terminates, the **utmp** entry must be set to DEAD_PROCESS.

Port Monitor Process IDs and Lock Files

When a port monitor starts, it writes its process id into a file named **_pid** in the current directory and places an advisory lock on the file.

Changing the Service Environment: Running

doconfig(3N) Before invoking the service designated in the port monitor administrative file, **_pmtab**, a port monitor must arrange for the per-service configuration script to be run, if one exists, by calling the library function **doconfig**(3N). Because the per-service configuration script may specify the execution of restricted commands, as well as for other security reasons, port monitors are invoked with root permissions. The details of how services are invoked are specified by the person who defines the port monitor.

Terminating a Port Monitor

A port monitor must terminate itself gracefully on receipt of the signal SIGTERM. The termination sequence is the following:

1. The port monitor enters the stopping state; no further service requests are accepted.
2. Any attempt to re-enable the port monitor will be ignored.
3. The port monitor yields control of all ports for which it is responsible. It must be possible for a new instantiation of the port monitor to start correctly while a previous instantiation is stopping.
4. The advisory lock on the process id file is released. Once this lock is released, the contents of the process id file are undefined and a new invocation of the port monitor may be started.

SAF Files

This section briefly covers the files used by the SAF.

The Port Monitor Administrative File

A port monitor's current directory contains an administrative file named **_pmtab**; **_pmtab** is maintained by the **pmadm** command in conjunction with a port monitor-specific administrative command.

Note: The port monitor administrative command for a listen port monitor is **nlsadmin**(1M); the port monitor administrative command for ttymon is **ttyadm**(1M). Any port monitor written by a user must be provided with an administrative command specific to that port monitor to perform similar functions.

Per-Service Configuration Files

A port monitor's current directory also contains the per-service configuration scripts, if they exist. The names of the per-service configuration scripts correspond to the service tags in the **_pmtab** file.

Private Port Monitor Files

A port monitor may create private files in the directory **/var/saf/***tag*, where *tag* is the name of the port monitor. Examples of private files are log files or temporary files.

The SAC/Port Monitor Interface

The SAC creates two environment variables for each port monitor it starts:

- PMTAG
- ISTATE

This variable is set to a unique port monitor tag by the SAC. The port monitor uses this tag to identify itself in response to **sac** messages. ISTATE is used to indicate to the port monitor what its initial internal state should be. ISTATE is set to "enabled" or "disabled" to indicate that the port monitor is to start in the enabled or disabled state respectively.

The SAC performs a periodic sanity poll of the port monitors. The SAC communicates with port monitors through FIFOs. A port monitor should open _pmpipe, in the current directory, to receive messages from the SAC and **../_sacpipe** to send return messages to the SAC.

Message Formats

This section describes the messages that may be sent from the SAC to a port monitor (**sac** messages), and from a port monitor to the SAC (port monitor messages). These messages are sent through FIFOs and are in the form of C structures.

sac Messages

The format of messages from the SAC is defined by the structure **sacmsg**:

```
struct sacmsg
{
        int sc_size; /* size of optional data portion */
        char sc_type; /* type of message */
};
```

The SAC may send four types of messages to port monitors. The type of message is indicated by setting the **sc_type** field of the **sacmsg** structure to one of the following:

SC_STATUS	status request
SC_ENABLE	enable message
SC_DISABLE	disable message
SC_READDB	message indicating that the port monitor's _pmtab file should be read

The **sc_size** field indicates the size of the optional data part of the message. See "Message Classes." For Solaris, **sc_size** should always be set to 0. A port monitor must respond to every message sent by the sac.

Port Monitor Messages

The format of messages from a port monitor to the SAC is defined by the structure **pmmsg**:

```
struct pmmsg {
        char pm_type;          /* type of message */
        unchar pm_state;       /* current state of port monitor */
        char pm_maxclass;      /* maximum message class this port
                                  monitor understands */
        char pm_tag[PMTAGSIZE + 1];  /* port monitor's tag */
        int pm_size;           /* size of optional data portion */
};
```

Port monitors may send two types of messages to the SAC. The type of message is indicated by setting the **pm_type** field of the **pmmsg** structure to one of the following:

PM_STATUS	state information

PM_UNKNOWN	negative acknowledgment

For both types of messages, the **pm_tag** field is set to the port monitor's tag and the **pm_state** field is set to the port monitor's current state. Valid states are:

PM_STARTING	starting
PM_ENABLED	enabled
PM_DISABLED	disabled
PM_STOPPING	stopping

The current state reflects any changes caused by the last message from the SAC. The status message is the normal return message. The negative acknowledgment should be sent only when the message received is not understood. **pm_size** indicates the size of the optional data part of the message. **pm_maxclass** is used to specify a message class. Both are discussed under "Message Classes." In Solaris, always set **pm_maxclass** to 1 and **sc_size** to 0. Port monitors may never initiate messages; they may only respond to messages that they receive.

Message Classes

The concept of message class has been included to accommodate possible SAF extensions. The messages described above are all class 1 messages. None of these messages contains a variable data portion; all pertinent information is contained in the message header. If new messages are added to the protocol, they will be defined as new message classes (for example, class 2). The first message the SAC sends to a port monitor will always be a class 1 message. Since all port monitors, by definition, understand class 1 messages, the first message the SAC sends is guaranteed to be understood. In its response to the SAC, the port monitor sets the **pm_maxclass** field to the maximum message class number for that port monitor. The SAC will not send messages to a port monitor from a class with a larger number than the value of **pm_maxclass**. Requests that require messages of a higher class than the port monitor can understand will fail. For Solaris, always set **pm_maxclass** to 1.

Note: For any given port monitor, messages of class **pm_maxclass** and messages of all classes with values lower than **pm_maxclass** are valid. Thus, if the **pm_maxclass** field is set to 3, the port monitor understands messages of classes 1, 2, and 3. Port monitors may not generate messages; they may only respond to messages. A port monitor's response must be of the same class as the originating message. Since only the SAC can generate messages, this protocol will function even if the port monitor is capable of dealing with messages of a higher class than the SAC can generate. **pm_size** (an element of the pmmsg structure) and **sc_size** (an element of the **sacmsg** structure) indicate the size of the optional data part of the message. The format of this part of the message is undefined. Its definition is inherent in the type of message. For Solaris, always set both **sc_size** and **pm_size** to 0.

Administrative Interface

This section discusses the port monitor administrative files available under the SAC.

The SAC Administrative File _sactab

The service access controller's administrative file contains information about all the port monitors for which the SAC is responsible. This file exists on the delivered system. Initially, it is empty except for a single comment line that contains the version number of the SAC. Port monitors are added to the system by making entries in the SAC's administrative file. These entries should be made using the administrative command **sacadm**(1M) with a –**a** option. **sacadm**(1M) is also used to remove entries from the SAC's administrative file. Each entry in the SAC's administrative file contains the following information.

PMTAG
A unique tag that identifies a particular port monitor. The system administrator is responsible for naming a port monitor. This tag is then used by the SAC to identify the port monitor for all administrative purposes. PMTAG may consist of up to 14 alphanumeric characters.

PMTYPE
The type of the port monitor. In addition to its unique tag, each port monitor has a type designator. The type designator identifies a group of port monitors that are different invocations of the same entity. ttymon and listen are examples of valid port monitor types. The type designator is used to facilitate the administration of groups of related port monitors. Without a type designator, the system administrator has no way of knowing which port monitor tags correspond to port monitors of the same type. PMTYPE may consist of up to 14 alphanumeric characters.

FLGS
The flags that are currently defined are:

d	When started, do not enable the port monitor.
x	Do not start the port monitor.

If no flag is specified, the default action is taken. By default a port monitor is started and enabled.

RCNT
The number of times a port monitor may fail before being placed in a failed state. Once a port monitor enters the failed state, the SAC will not try to restart it. If a count is not specified when the entry is created, this field is set to 0. A restart count of 0 indicates that the port monitor is not to be restarted when it fails.

COMMAND
A string representing the command that will start the port monitor. The first component of the string, the command itself, must be a full path name.

The Port Monitor Administrative File _pmtab

Each port monitor will have two directories for its exclusive use. The current directory will contain files defined by the SAF (**_pmtab**, **_pid**) and the per-service configuration scripts, if they exist. The directory **/var/saf/***pmtag,* where *pmtag* is the tag of the port monitor, is available for the port monitor's private files. Each port monitor has its own administrative file. The **pmadm**(1M) command should be used to add, remove, or modify service entries in this file. Each time a change is made using **pmadm**(1M), the corresponding port monitor rereads its administrative file. Each entry in a port monitor's administrative file defines how the port monitor treats a specific port and what service is to be invoked on that port. Some fields must be present for all types of port monitors. Each entry must include a service tag to identify the service uniquely and an identity to be assigned to the service when it is started (for example, root).

Note: The combination of a service tag and a port monitor tag uniquely define an instance of a service. The same service tag may be used to identify a service under a different port monitor. The record must also contain port monitor specific data (for example, for a ttymon port monitor, this will include the prompt string which is meaningful to ttymon). Each type of port monitor must provide a command that takes the necessary port monitor-specific data as arguments and outputs these data in a form suitable for storage in the file. The **ttyadm**(1M) command does this for ttymon and **nlsadmin**(1M) does it for listen. For a user-defined port monitor, a similar administrative command must also be supplied. Each service entry in the port monitor administrative file must have the following format and contain the information listed below:

```
svctag:flgs:id:reserved:reserved:reserved:pmspecific# comment
```

SVCTAG is a unique tag that identifies a service. This tag is unique only for the port monitor through which the service is available. Other port monitors may offer the same or other services with the same tag. A service requires both a port monitor tag and a service tag to identify it uniquely. SVCTAG may consist of up to 14 alphanumeric characters. The service entries are defined as:

FLGS
Flags with the following meanings may currently be included in this field:

x
Do not enable this port. By default the port is enabled.

u
Create a utmp entry for this service. By default no utmp entry is created for the service.

ID
The identity under which the service is to be started. The identity has the form of a login name as it appears in **/etc/passwd**.

PMSPECIFIC
Examples of port monitor information are addresses, the name of a process to execute, or the name of a STREAMS pipe to pass a connection through. This information will vary to meet the needs of each different type of port monitor.

COMMENT
A comment associated with the service entry. Note: Port monitors may ignore the *u* flag if creating a utmp entry for the service is not appropriate to the manner in which the service is to be invoked. Some services may not start properly unless utmp entries have been created for them (for example, login). Each port monitor administrative file must contain one special comment of the form:

```
# VERSION=value
```

where *value* is an integer that represents the port monitor's version number. The version number defines the format of the port monitor administrative file. This comment line is created automatically when a port monitor is added to the system. It appears on a line by itself, before the service entries.

Monitor-Specific Administrative Command

Previously, two pieces of information included in the **_pmtab** file were described: the port monitor's version number and the port monitor part of the service entries in the port monitor's **_pmtab** file. When a new port monitor is added, the version number must be known so that the **_pmtab** file can be correctly initialized. When a new service is added, the port monitor part of the **_pmtab** entry must be formatted correctly. Each port monitor must have an administrative command to perform these two tasks. The person who defines the port monitor must also define such an administrative command and its input options. When the command is invoked with these options, the information required for the port monitor part of the service entry must be correctly formatted for inclusion in the port monitor's **_pmtab** file and must be written to the standard output. To request the version number the command must be invoked with a −**V** option; when it is invoked in this way, the port monitor's current version number must be written to the standard output. If the command fails for any reason during the execution of either of these tasks, no data should be written to standard output.

The Port Monitor/Service Interface

The interface between a port monitor and a service is determined solely by the service. Two mechanisms for invoking a service are presented here as examples.

New Service Invocations

The first interface is for services that are started anew with each request. This interface requires the port monitor to first **fork**(2) a child process. The child will eventually become the designated service by performing an **exec**(1). Before the **exec**(1) happens, the port monitor may take some port monitor-specific action; however, one action that must occur is the interpretation of the per-service configuration script, if one is present. This is done by calling the library routine **doconfig**(3N).

Standing Service Invocations

The second interface is for invocations of services that are actively running. To use this interface, a service must have one end of a stream pipe open and be prepared to receive connections through it.

Port Monitor Requirements

To implement a port monitor, several generic requirements must be met. This section summarizes these requirements. In addition to the port monitor itself, an administrative command must be supplied.

Initial Environment

When a port monitor is started, it expects an initial execution environment in which:

- It has no file descriptors open
- It cannot be a process group leader
- It has an entry in **/etc/utmp** of type LOGIN_PROCESS

- An environment variable, ISTATE, is set to "enabled" or "disabled" to indicate the port monitor's correct initial state
- An environment variable, PMTAG, is set to the port monitor's assigned tag
- The directory that contains the port monitor's administrative files is its current directory
- The port monitor is able to create private files in the directory **/var/saf/***tag*, where *tag* is the port monitor's tag
- The port monitor is running with user id 0 (root)

Important Files

Relative to its current directory, the following key files exist for a port monitor.

_config The port monitor's configuration script. The port monitor configuration script is run by the SAC. The SAC is started by **init**(1M) as a result of an entry in **/etc/inittab** that calls **sac**(1M).

_pid The file into which the port monitor writes its process id.

_pmtab The port monitor's administrative file. This file contains information about the ports and services for which the port monitor is responsible.

_pmpipe The FIFO through which the port monitor will receive messages from the SAC.

svctag The per-service configuration script for the service with the tag *svctag*.

../_sacpipe
The FIFO through which the port monitor will send messages to **sac**(1M).

Port Monitor Responsibilities

A port monitor is responsible for performing the following tasks in addition to its port monitor function:

- Write its process id into the file **_pid** and place an advisory lock on the file
- Terminate gracefully on receipt of the signal SIGTERM
- Follow the protocol for message exchange with the SAC

A port monitor must perform the following tasks during service invocation:

- Create a **utmp** entry if the requested service has the **u** flag set in **_pmtab**

  Note: Port monitors may ignore this flag if creating a **utmp** entry for the service does not make sense because of the manner in which the service is to be invoked. On the other hand, some services may not start properly unless utmp entries have been created for them.
- Interpret the per-service configuration script for the requested service, if it exists, by calling the **doconfig**(3N) library routine

Configuration Files and Scripts

The library routine **doconfig**(3N), defined in **libnsl.so**, interprets the configuration scripts contained in the files **/etc/saf/_sysconfig** (the per-system configuration file), and **/etc/saf/***pmtag***/_config** (per-port monitor configuration files); and in **/etc/saf/***pmtag***/svctag** (per-service configuration files). Its syntax is:

```
#include <sac.h>
int doconfig (int fd, char *script, long rflag);
```

script is the name of the configuration script; *fd* is a file descriptor that designates the stream to which stream manipulation operations are to be applied; *rflag* is a bitmask that indicates the mode in which script is to be interpreted. *rflag* may take two values, NORUN and NOASSIGN, which may be or'd. If *rflag* is zero, all commands in the configuration script are eligible to be interpreted. If *rflag* has the NOASSIGN bit set, the assign command is considered illegal and will generate an error return. If *rflag* has the NORUN bit set, the run and runwait commands are considered illegal and will generate error returns. If a command in the script fails, the interpretation of the script ceases at that point and a positive integer is returned; this number indicates which line in the script failed. If a system error occurs, a value of –1 is returned. If a script fails, the process whose environment was being established should not be started. In the example, **doconfig**(3N) is used to interpret a per-service configuration script.

```
. . .
        if ((i = doconfig (fd, svctag, 0)) != 0){
        error ("doconfig failed on line %d of script %s",i,svctag);
}
```

The Per-System Configuration File

The per-system configuration file, **/etc/saf/_sysconfig**, is delivered empty. It may be used to customize the environment for all services on the system by writing a command script in the interpreted language described in this chapter and on the **doconfig**(3N) manpage. When the SAC is started, it calls the **doconfig**(3N) function to interpret the per-system configuration script. The SAC is started when the system enters multiuser mode.

Per-Port Monitor Configuration Files

Per-port monitor configuration scripts (**/etc/saf/***pmtag***/_config)** are optional. They allow the user to customize the environment for any given port monitor and for the services that are available through the ports for which that port monitor is responsible. Per-port monitor configuration scripts are written in the same language used for per-system configuration scripts. The per-port monitor configuration script is interpreted when the port monitor is started. The port monitor is started by the SAC after the SAC has itself been started and after it has run its own configuration script, **/etc/saf/_sysconfig**. The per-port monitor configuration script may override defaults provided by the per-system configuration script.

Per-Service Configuration Files

Per-service configuration files allow the user to customize the environment for a specific service. For example, a service may require special privileges that are not available to the general user. Using the language described in the **doconfig**(3N) manpage, you can write a script that will grant or limit such special privileges to a particular service offered through a particular port monitor. The per-service configuration may override defaults provided by higher-level configuration scripts. For example, the per-service configuration script may specify a set of STREAMS modules other than the default set.

The Configuration Language

The language in which configuration scripts are written consists of a sequence of commands, each of which is interpreted separately. The following reserved keywords are defined: **assign**, **push**, **pop**, **runwait**, and **run**. The comment character is #. Blank lines are not significant. No line in a command script may exceed 1024 characters.

assign *variable=value*

Used to define environment variables; *variable* is the name of the environment variable and *value* is the value to be assigned to it. The value assigned must be a string constant; no form of parameter substitution is available. *value* may be quoted. The quoting rules are those used by the shell for defining environment variables. **assign** will fail if space cannot be allocated for the new variable or if any part of the specification is invalid.

push *module1*[, *module2*, *module3*, . . .]

Used to push STREAMS modules onto the stream designated by **fd**; *module1* is the name of the first module to be pushed, *module2* is the name of the second module to be pushed, and so on. The command will fail if any of the named modules cannot be pushed. If a module cannot be pushed, the subsequent modules on the same command line will be ignored and modules that have already been pushed will be popped.

pop [*module*]

Used to pop STREAMS modules off the designated stream. If **pop** is invoked with no arguments, the top module on the stream is popped. If an argument is given, modules will be popped one at a time until the named module is at the top of the stream. If the named module is not on the designated stream, the stream is left as it was and the command fails. If *module* is the special keyword **ALL**, then all modules on the stream will be popped. Note that only modules above the top-most driver are affected.

runwait *command*

The **runwait** command runs a command and waits for it to complete; *command* is the path name of the command to be run. The command is run with **/bin/sh –c** prepended to it; shell scripts may thus be executed from configuration scripts. The **runwait** command will fail if command cannot be found or cannot be executed, or if *command* exits with a nonzero status.

run *command*

The **run** command is identical to **runwait** except that it does not wait for command to complete; *command* is the path name of the command to be run. **run** will not fail unless it is unable to create a child process to execute the command. Although they are syntactically indistinguishable, some of the commands available to **run** and **runwait** are interpreter built-in commands. Interpreter built-ins are used when it is necessary to alter the state of a process within the context of that process. The **doconfig** interpreter built-in commands are similar to the shell special commands and, like these, they do not spawn another process for execution. See the **sh**(1) man page. The initial set of built-in commands is: **cd**, **ulimit**, **umask**.

Sample Port Monitor Code

This example shows an example of a "null" port monitor that simply responds to messages from the SAC.

```
# include <stdlib.h>
# include <stdio.h>
# include <unistd.h>
# include <fcntl.h>
# include <signal.h>
# include <sac.h>

char Scratch[BUFSIZ]; /* scratch buffer */
char Tag[PMTAGSIZE + 1]; /* port monitor's tag */
FILE *Fp; /* file pointer for log file */
FILE *Tfp; /* file pointer for pid file */
char State; /* portmonitor's current state*/

main(argc, argv)
        int argc;
        char *argv[];
{
        char *istate;
        strcpy(Tag, getenv("PMTAG"));
/*
 * open up a log file in port monitor's private directory
 */
        sprintf(Scratch, "/var/saf/%s/log", Tag);
        Fp = fopen(Scratch, "a+");
        if (Fp == (FILE *)NULL)
                exit(1);
        log(Fp, "starting");
/*
 * retrieve initial state (either "enabled" or "disabled") and set
 * State accordingly
 */
        istate = getenv("ISTATE");
        sprintf(Scratch, "ISTATE is %s", istate);
        log(Fp, Scratch);
        if (!strcmp(istate, "enabled"))
                State = PM_ENABLED;
        else if (!strcmp(istate, "disabled"))
                State = PM_DISABLED;
        else {
                log(Fp, "invalid initial state");
                exit(1);
        }
        sprintf(Scratch, "PMTAG is %s", Tag);
        log(Fp, Scratch);
/*
 * set up pid file and lock it to indicate that we are active
 */
        Tfp = fopen("_pid", "w");
        if (Tfp == (FILE *)NULL) {
                log(Fp, "couldn't open pid file");
                exit(1);
        }
        if (lockf(fileno(Tfp), F_TEST, 0) < 0) {
                log(Fp, "pid file already locked");
                exit(1);
        }

        log(Fp, "locking file");
        if (lockf(fileno(Tfp), F_LOCK, 0) < 0) {
                log(Fp, "lock failed");
                exit(1);
        }
        fprintf(Tfp, "%d", getpid());
        fflush(Tfp);

/*
 * handle poll messages from the sac ... this function never returns
 */
        handlepoll();
        pause();
        fclose(Tfp);
        fclose(Fp);
}

handlepoll()
{
        int pfd; /* file descriptor for incoming pipe */
        int sfd; /* file descriptor for outgoing pipe */
        struct sacmsg sacmsg; /* incoming message */
        struct pmmsg pmmsg; /* outgoing message */
/*
 * open pipe for incoming messages from the sac
 */
        pfd = open("_pmpipe", O_RDONLY | O_NONBLOCK);
        if (pfd < 0) {
                log(Fp, "_pmpipe open failed");
                exit(1);
        }
/*
 * open pipe for outgoing messages to the sac
 */
        sfd = open("../_sacpipe", O_WRONLY);
        if (sfd < 0) {
                log(Fp, "_sacpipe open failed");
                exit(1);
        }
/*
 * start to build a return message; we only support class 1 messages
 */
        strcpy(pmmsg.pm_tag, Tag);
        pmmsg.pm_size = 0;
        pmmsg.pm_maxclass = 1;
/*
 * keep responding to messages from the sac
 */
        for (;;) {
                if (read(pfd, &sacmsg, sizeof(sacmsg)) != sizeof(sacmsg)) {
                        log(Fp, "_pmpipe read failed");
                        exit(1);
                }
/*
 * determine the message type and respond appropriately
 */
                switch (sacmsg.sc_type) {
                        case SC_STATUS:
                                log(Fp, "Got SC_STATUS message");
                                pmmsg.pm_type = PM_STATUS;
                                pmmsg.pm_state = State;
                                break;
                        case SC_ENABLE:
                                /*note internal state change below*/
                                log(Fp, "Got SC_ENABLE message");
                                pmmsg.pm_type = PM_STATUS;
                                State = PM_ENABLED;
                                pmmsg.pm_state = State;
                                break;
                        case SC_DISABLE:
                                /*note internal state change below*/
                                log(Fp, "Got SC_DISABLE message");
                                pmmsg.pm_type = PM_STATUS;
                                State = PM_DISABLED;
                                pmmsg.pm_state = State;
                                break;
                        case SC_READDB:
                                /*
                                 * if this were a fully functional port
                                 * monitor it would read _pmtab here
                                 * and take appropriate action
                                 */
                                log(Fp, "Got SC_READDB message");
                                pmmsg.pm_type = PM_STATUS;
                                pmmsg.pm_state = State;
                                break;
                        default:
                                sprintf(Scratch, "Got unknown message <%d>",
                                sacmsg.sc_type);
                                log(Fp, Scratch);
                                pmmsg.pm_type = PM_UNKNOWN;
                                pmmsg.pm_state = State;
                                break;
                }
/*
 * send back a response to the poll
 * indicating current state
 */
                if (write(sfd, &pmmsg, sizeof(pmmsg)) != sizeof(pmmsg))
                        log(Fp, "sanity response failed");
        }
}
/*
 * general logging function
 */
log(fp, msg)
        FILE *fp;
        char *msg;
{
        fprintf(fp, "%d; %s\n", getpid(), msg);
        fflush(fp);
}
```

The sac.h Header File

The following example shows the sac.h header file.

```
/* length in bytes of a utmp id */
# define IDLEN 4
/* wild character for utmp ids */
# define SC_WILDC 0xff
/* max len in bytes for port monitor tag */
# define PMTAGSIZE 14
/*
 * values for rflag in doconfig()
 */
/* don’t allow assign operations */
# define NOASSIGN 0x1
/* don’t allow run or runwait operations */
# define NORUN 0x2
/*
 * message to SAC (header only). This header is forever fixed. The
 * size field (pm_size) defines the size of the data portion of the
 * message, which follows the header. The form of this optional data
 * portion is defined strictly by the message type (pm_type).
 */
struct pmmsg {
        char pm_type;           /* type of message */
        unchar pm_state;        /* current state of pm */
        char pm_maxclass;       /* max message class this port monitor
                                        understands */
        char pm_tag[PMTAGSIZE + 1]; /* pm’s tag */
        int pm_size;            /* size of opt data portion */
};
/*
 * pm_type values
 */
# define PM_STATUS 1 /* status response */
# define PM_UNKNOWN 2 /* unknown message was received */
/*
 * pm_state values
 */
/*
 * Class 1 responses
 */
# define PM_STARTING 1   /* monitor in starting state */
# define PM_ENABLED 2    /* monitor in enabled state */
# define PM_DISABLED 3   /* monitor in disabled state */
# define PM_STOPPING 4   /* monitor in stopping state */
/*
 * message to port monitor
 */
struct sacmsg {
        int sc_size;      /* size of optional data portion */
        char sc_type;     /* type of message */
};
/*
 * sc_type values
 * These represent commands that the SAC sends to a port monitor.
 * These commands are divided into "classes" for extensibility. Each
 * subsequent "class" is a superset of the previous "classes" plus
 * the new commands defined within that "class". The header for all
 * commands is identical; however, a command may be defined such that
 * an optional data portion may be sent in addition to the header.
 * The format of this optional data piece is self-defining based on
 * the command. Important note: the first message sent by the SAC
 * will always be a class 1 message. The port monitor response
 * indicates the maximum class that it is able to understand. Another
 * note is that port monitors should only respond to a message with
 * an equivalent class response (i.e. a class 1 command causes a
 * class 1 response).
 */
/*
 * Class 1 commands (currently, there are only class 1 commands)
 */
# define SC_STATUS 1    /* status request *
# define SC_ENABLE 2    /* enable request */
# define SC_DISABLE 3   /* disable request */
# define SC_READDB 4    /* read pmtab request */
/*
 * ‘errno’ values for Saferrno, note that Saferrno is used by both
 * pmadm and sacadm and these values are shared between them
 */
# define E_BADARGS 1   /* bad args/ill-formed cmd line */
# define E_NOPRIV 2    /* user not priv for operation */
# define E_SAFERR 3    /* generic SAF error */
# define E_SYSERR 4    /* system error */
# define E_NOEXIST 5   /* invalid specification */
# define E_DUP 6       /* entry already exists */
# define E_PMRUN 7     /* port monitor is running */
# define E_PMNOTRUN 8  /* port monitor is not running */
# define E_RECOVER 9   /* in recovery */
```

Directory Structure

This section gives a description of the SAF files and directories.

/etc/saf/_sysconfig
The per-system configuration script.

/etc/saf/_sactab
The SAC's administrative file. Contains information about the port monitors for which the SAC is responsible.

/etc/saf/*pmtag*
The home directory for port monitor *pmtag*.

/etc/saf/*pmtag***/_config**
The per-port monitor configuration script for port monitor pmtag. **/etc/saf/***pmtag***/_pmtab** Port monitor pmtag's administrative file. Contains information about the services for which *pmtag* is responsible. **/etc/saf/***pmtag*/*svctag* The file in which the per-service configuration script for service *svctag* (available through port monitor **pmtag**) is placed. **/etc/saf/***pmtag***/_pid** The file in which a port monitor writes its process id in the current directory and places an advisory lock on the file. **/etc/saf/ pmtag /_pmpipe** The file in which the port monitor receives messages from the SAC and **../_sacpipe** and sends return messages to the SAC. **/var/saf/_log** The SAC's log file. **/var/saf/***pmtag* The directory for files created by port monitor *pmtag*, for example its log file.

LIST OF COMMANDS

The following administrative commands relate to SAF.

sacadm(1M)
port monitor administrative command

pmadm(1M)
service administration command

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

exec(1), **sh**(1), **init**(1M), **nlsadmin**(1M), **pmadm**(1M), **sac**(1M), **sacadm**(1M), **ttyadm**(1M), **fork**(2), **doconfig**(3N), **attributes**(5)

NAME

sar, sa1, sa2, sadc – system activity report package

SYNOPSIS

/usr/lib/sa/sadc [*t n*] [*ofile*]

/usr/lib/sa/sa1 [*t n*]

/usr/lib/sa/sa2 [**–aAbcdgkmpqruvwy**] [**–e** *time*] [**–f** *filename*] [**–i** *sec*] [**–s** *time*]

DESCRIPTION

System activity data can be accessed at the special request of a user (see **sar**(1)) and automatically, on a routine basis, as described here. The operating system contains several counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, and paging. For more general system statistics, use **iostat**(1M), **sar**(1), or **vmstat**(1M).

See *Solaris 1.x to 2.x Transition Guide* for device naming conventions for disks.

sadc and two shell procedures, **sa1** and **sa2**, are used to sample, save, and process this data.

sadc, the data collector, samples system data *n* times, with an interval of *t* seconds between samples, and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than 5 seconds; otherwise, the activity of **sadc** itself may affect the sample. If *t* and *n* are omitted, a special record is written. This facility can be used at system boot time, when booting to a multi-user state, to mark the time at which the counters restart from zero. For example, when accounting is enabled, the **/etc/init.d/perf** file writes the restart mark to the daily data file using the command entry:

```
su sys –c "/usr/lib/sa/sadc /var/adm/sa/sa`date +%d`"
```

The shell script **sa1**, a variant of **sadc**, is used to collect and store data in the binary file **/var/adm/sa/sa***dd*, where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The following entries in **/var/spool/cron/crontabs/sys** will produce records every 20 minutes during working hours and hourly otherwise:

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8–17 * * 1–5 /usr/lib/sa/sa1
```

See **crontab**(1) for details.

The shell script **sa2**, a variant of **sar**, writes a daily report in the file **/var/adm/sa/sar***dd*. See the **OPTIONS** section in **sar**(1) for an explanation of the various options. The following entry in **/var/spool/cron/crontabs/sys** will report important activities hourly during the working day:

```
5 18 * * 1–5 /usr/lib/sa/sa2 –s 8:00 –e 18:01 –i 1200 –A
```

FILES

/etc/init.d/perf	
/tmp/sa.adrfl	address file
/var/adm/sa/sa*dd*	daily data file
/var/adm/sa/sar*dd*	daily report file
/var/spool/cron/crontabs/sys	

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

SEE ALSO

crontab(1), **sag**(1), **sar**(1), **timex**(1), **iostat**(1M), **vmstat**(1M), **attributes**(5)

System Administration Guide
Solaris 1.x to 2.x Transition Guide

NAME

savecore – save a core dump of the operating system

SYNOPSIS

/usr/bin/savecore [**–v**] *directory* [*system-name*]

DESCRIPTION

The **savecore** utility saves a core dump of the kernel (assuming that one was made) and writes a reboot message in the shutdown log. It is meant to be called near the end of the **/etc/init.d/sysetup** file after the system boots. However, it is not normally run by default; you must edit that file to enable it.

The **savecore** utility checks the core dump to be certain it corresponds with the version of the operating system currently running. If it does, **savecore** saves the core image in the file *directory*/**vmcore.***n* and the kernel's namelist in *directory*/**unix.***n*. The trailing **.***n* in the pathnames is replaced by a number which grows every time **savecore** is run in that directory.

Before writing out a core image, **savecore** reads a number from the file *directory*/**minfree**. This is the minimum number of kilobytes that must remain free on the file system containing *directory*. If there is less free space on the file system containing *directory* than the number of kilobytes specified in **minfree**, the core dump is not saved. If the **minfree** file does not exist, **savecore** always writes out the core file (assuming that a core dump was taken).

The **savecore** utility also logs a reboot message using facility **LOG_AUTH** (see **syslog**(3)). If the system crashed as a result of a panic, **savecore** logs the panic string too.

If the core dump corresponds to a namelist other than the currently running kernel (represented by **/dev/ksyms**) that namelist must be supplied as *system-name*.

OPTIONS

The following options are supported:

–v Verbose. Enable verbose error messages from **savecore**.

FILES

directory/**vmcore.***n*	
directory/**unix.***n*	
directory/**bounds**	
directory/**minfree**	
/dev/ksyms	the kernel namelist
/etc/init.d/sysetup	
/var/crash/‘uname –n‘	normal core image directory

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtoo

SEE ALSO

compress(1), **crash**(1M), **syslog**(3), **attributes**(5)

BUGS

The **savecore** utility can be fooled into thinking a core dump is the wrong size.

You must run **savecore** very soon after booting, before the swap space containing the crash dump is overwritten by programs currently running.

NAME

sendmail – send mail over the internet

SYNOPSIS

/usr/lib/sendmail [–**ba**] [–**bd**] [–**bi**] [–**bm**] [–**bp**] [–**bs**] [–**bt**] [–**bv**]
[–**B** *type*] [–**C** *file*] [–**d** *X*] [–**F** *fullname*] [–**f** *name*] [–**h** *N*] [–**M** *id*] [–**n**]
[–**o** **x***value*] [–**p** *protocol*] [–**q** [*time*]] [–**q** *Xstring*] [–**r** *name*] [–**t**] [–**v**]
[–**X** *logfile*] [*address*. . .]

DESCRIPTION

sendmail sends a message to one or more people, routing the message over whatever networks are necessary. **sendmail** does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends. **sendmail** is used only to deliver pre-formatted messages.

With no flags, **sendmail** reads its standard input up to an EOF, or a line with a single dot, and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in the local **aliases**(4) file, or by using the YP name service, and aliased appropriately. In addition, if there is a **.forward** file in a recipient's home directory, **sendmail** forwards a copy of each message to the list of recipients that file contains. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in alias expansions. For example, if "john" sends to "group", and "group" includes "john" in the expansion, then the letter will not be delivered to "john".

sendmail will also route mail directly to other known hosts in a local network. The list of hosts to which mail is directly sent is maintained in the file **/usr/lib/mailhosts**.

If a letter is found to be undeliverable, it is returned to the sender with diagnostics that indicate the location and nature of the failure; or, the letter is placed in a **dead.letter** file in the sender's home directory.

OPTIONS

–**ba** Go into ARPANET mode. All input lines must end with a RETURN-LINEFEED, and all messages will be generated with a RETURN-LINEFEED at the end. Also, the **From:** and **Sender:** fields are examined for the name of the sender.

–**bd** Run as a daemon, waiting for incoming SMTP connections.

–**bi** Initialize the **aliases**(4) database.

–**bm** Deliver mail in the usual way (default).

–**bp** Print a summary of the mail queue.

–**bs** Use the SMTP protocol as described in RFC 821. This flag implies all the operations of the –**ba** flag that are compatible with SMTP.

–bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

–bv Verify names only; do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

–B *type* Indicate body *type* (**7BIT** or **8BITMIME**).

–C *file* Use alternate configuration file.

–d *X* Set debugging value to *X*.

–F *fullname* Set the full name of the sender.

–f *name* Sets the name of the "from" person (that is, the sender of the mail).

–h *N* Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

–M *id* Attempt to deliver the queued message with message-id *id*. This option is supported for backward compatibility and the –**qI** option is preferred.

–n Do not do aliasing.

–o *xvalue* Set option *x* to the specified *value*. **Processing Options** are described below.

–p *protocol* Set the sending protocol. Programs are encouraged to set this. The *protocol* field can be in form *protocol*:*host* to set both the sending protocol and the sending host. For example:

–pUUCP:uunet

sets the sending *protocol* to **UUCP** and the sending host to **uunet**. (Some existing programs use –**oM** to set the *r* and *s* macros; this is equivalent to using –**p**).

–q[*time*] Process saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with **s** being seconds, **m** being minutes, **h** being hours, **d** being days, and **w** being weeks. For example, –**q1h30m** or –**q90m** would both set the timeout to one hour thirty minutes.

–q *Xstring* Run the queue once, limiting the jobs to those matching *Xstring*. The key letter *X* can be:

I to limit based on queue identifier (see –**M** option).
R to limit based on recipient (see –**R** option).
S to limit based on sender.

A particular queued job is accepted if one of the corresponding addresses contains the indicated *string*.

–r *name* An alternate and obsolete form of the –**f** flag.

–R *string* Go through the queue of pending mail and attempt to deliver any message with a recipient containing the specified string. This is useful for clearing out mail directed to a machine which has been down for awhile. This option is supported for backward compatibility and the **–qR** option is preferred.

–t Read message for recipients. **To:**, **Cc:**, and **Bcc:** lines will be scanned for people to send to. The **Bcc:** line will be deleted before transmission. Any addresses in the argument list will be suppressed.

However, if neither the **To:** or **Cc:** lines have an entry, the recipients on the **Bcc:** line will be identified with an **Apparently-To:** header line when the message is received, thereby violating the presumed privacy of the original **Bcc:** header. To avoid this, make sure that messages always include a minimal **To:** line, even if it is a dummy line with a comment, such as

To: (names withheld)

–v Go into verbose mode. Alias expansions will be announced, and so forth.

–X *logfile* Log all traffic in and out of **sendmail** in the indicated *logfile* for debugging mailer problems. This produces a lot of data very quickly and should be used sparingly.

Processing Options

There are a number of "random" options that can be set from a configuration file. Options are represented by single characters. The syntax of this line is:

O*ovalue*

This sets option *o* to be *value*. Depending on the option, *value* may be

- a string
- an integer
- a boolean (with legal values **t**, **T**, **f**, or **F**; the default is **TRUE**)
- a time interval

The options supported are:

a*N* If set, wait up to *N* minutes for an "@:@" entry to exist in the **aliases**(4) database before starting up. If it does not appear in *N* minutes, rebuild the database (if the **D** option is also set) or issue a warning.

A*file* Specify possible alias file(s).

b*N*/*M* Insist on at least *N* blocks free on the filesystem that holds the queue files before accepting email via SMTP. If there is insufficient space, **sendmail** gives a **452** response to the **MAIL** command. This invites the sender to try again later. The optional *M* is a maximum message size advertised in the **ESMTP EHLO** response. It is currently otherwise unused.

B*c* Set the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character. Defaults to SPACE (that is, no change is made).

c If an outgoing mailer is marked as being expensive, don't connect immediately.

C*N* Checkpoints the queue every *N* (default **10**) addresses sent. If your system crashes during delivery to a large list, this prevents retransmission to any but the last *N* recipients.

d*x* Deliver in mode *x*. Legal modes are:

i	Deliver interactively (synchronously).
b	Deliver in background (asynchronously).
q	Just queue the message (deliver during queue run).

Defaults to **b** if no option is specified, **i** if it is specified but given no argument (that is, **Od** is equivalent to **Odi**).

D If set, rebuild the **/etc/mail/aliases** database if necessary and possible. If this option is not set, **sendmail** will never rebuild the **aliases** database unless explicitly requested using −**bi**, or **newaliases**(1) is invoked.

e*x* Dispose of errors using mode *x*. The values for *x* are:

p	Print error messages (default).
q	No messages, just give exit status.
m	Mail back errors.
w	Write back errors (mail if user not logged in).
e	Mail back errors and give **0** exit status always.

E*file/message* Prepend error messages with the indicated message. If it begins with a slash, it is assumed to be the pathname of a file containing a message (this is the recommended setting). Otherwise, it is a literal message. The error file might contain the name, email address, and/or phone number of a local postmaster who could provide assistance to end users. If the option is missing or **NULL**, or if it names a file which does not exist or which is not readable, no message is printed.

f Save Unix-style **From** lines at the front of headers. Normally they are assumed redundant and discarded.

F*mode* The file mode for queue files.

g*n* Set the default group ID for mailers to run in to *n*. Defaults to **1**. The value can also be given as a symbolic group name.

h*N* The maximum hop count. Messages that have been processed more than *N* times are assumed to be in a loop and are rejected. Defaults to **25**.

H*file* Specify the help file for SMTP.

i Ignore dots in incoming messages. This is always disabled (that is, dots are always accepted) when reading SMTP mail.

I
Insist that the name server be running to resolve host names. If this is not set and the name server is not running, the **/etc/hosts** file (see **hosts**(4)) will be considered complete. In general, you do want to set this option if your **/etc/hosts** file does not include all hosts known to you or if you are using the MX (mail forwarding) feature of the name server. The name server will still be consulted even if this option is not set, but **sendmail** will feel free to resort to reading **/etc/hosts** if the name server is not available. Thus, you should *never* set this option if you do not run the name server.

j
If set, send error messages in MIME format (see RFC 1341 and RFC 1344 for details).

J*path*
Set the path for searching for users' **.forward** files. The default is **$z/.forward**. Some sites that use the automounter may prefer to change this to **/var/forward/$u** to search a file with the same name as the user in a system directory. It can also be set to a sequence of paths separated by colons; **sendmail** stops at the first file it can successfully and safely open. For example, **/var/forward/$u:$z/.forward** will search first in **/var/forward/** *username* and then in ~*username*/**.forward** (but only if the first file does not exist).

k*N*
The maximum number of open connections that will be cached at a time. The default is **1**. This delays closing the current connection until either this invocation of **sendmail** needs to connect to another host or it terminates. Setting it to **0** defaults to the old behavior, that is, connections are closed immediately.

K*timeout*
The maximum amount of time a cached connection will be permitted to idle without activity. If this time is exceeded, the connection is immediately closed. This value should be small (on the order of ten minutes). Before **sendmail** uses a cached connection, it always sends a **NOOP** (no operation) command to check the connection; if this fails, it reopens the connection. This keeps your end from failing if the other end times out. The point of this option is to be a good network neighbor and avoid using up excessive resources on the other end. The default is five minutes.

l
If there is an **Errors-To**: header, send error messages to the addresses listed there. They normally go to the envelope sender. Use of this option causes **sendmail** to violate RFC 1123.

L*n*
Set the default log level to *n*. Defaults to **9**.

m
Send to me too, even if I am in an alias expansion.

M*x value*
Set the macro *x* to *value*. This is intended only for use from the command line.

n
Validate the RHS of aliases when rebuilding the **aliases**(4) database.

o
Assume that the headers may be in old format, that is, spaces delimit

names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

O*options* Set server SMTP options. The options are *key*=*value* pairs. Known keys are:

Port	Name/number of listening port (defaults to **smtp**)
Addr	Address mask (defaults **INADDR_ANY**)
Family	Address family (defaults to **INET**)
Listen	Size of listen queue (defaults to **10**)

The **Addr**ess mask may be a numeric address in dot notation or a network name.

p *opt*,*opt*,. . . Set the privacy *opt*ions. ‘‘Privacy’’ is really a misnomer; many of these are just a way of insisting on stricter adherence to the SMTP protocol. The *opt*ions can be selected from:

public	Allow open access.
needmailhelo	Insist on **HELO** or **EHLO** command before **MAIL**.
needexpnhelo	Insist on **HELO** or **EHLO** command before **EXPN**.
noexpn	Disallow **EXPN** entirely.
needvrfyhelo	Insist on **HELO** or **EHLO** command before **VRFY**.
novrfy	Disallow **VRFY** entirely.
restrictmailq	Restrict **mailq** command.
restrictqrun	Restrict –**q** command line flag.
goaway	Disallow essentially all SMTP status queries.
authwarnings	Put X-Authentication-Warning: headers in messages.

The **goaway** pseudo-flag sets all flags except **restrictmailq** and **restrictqrun**. If **mailq** is restricted, only people in the same group as the queue directory can print the queue. If queue runs are restricted, only root and the owner of the queue directory can run the queue. Authentication Warnings add warnings about various conditions that may indicate attempts to spoof the mail system, such as using a non-standard queue directory.

P*postmaster* If set, copies of error messages will be sent to the named *postmaster*. Only the header of the failed message is sent. Since most errors are user problems, this is probably not a good idea on large sites, and arguably contains all sorts of privacy violations, but it seems to be popular with certain operating systems vendors.

q*factor* Use *factor* as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (**x** flag) to determine the maximum message priority that will be sent. Defaults to **600000**.

Q*dir* Use the named *dir* as the queue directory.

r *timeouts* Timeout reads after *time* interval. The *timeouts* argument is a list of *keyword*=*value* pairs. The recognized timeouts and their default values, and their minimum values specified in RFC 1123 section 5.3.2 are:

initial	wait for initial greeting message [5m, 5m]
helo	reply to **HELO** or **EHLO** command [5m, none]
mail	reply to **MAIL** command [10m, 5m]
rcpt	reply to **RCPT** command [1h, 5m]
datainit	reply to **DATA** command [5m, 2m]
datablock	data block read [1h, 3m]
datafinal	reply to final ‘‘.’’ in data [1h, 10m]
rset	reply to **RSET** command [5m, none]
quit	reply to **QUIT** command [2m, none]
misc	reply to **NOOP** and **VERB** commands [2m, none]
command	command read [1h, 5m]
ident	**IDENT** protocol timeout [30s, none]

All but **command** apply to client SMTP. For backward compatibility, a timeout with no *keyword*= part will set all of the longer values.

s Be super-safe when running things, that is, always instantiate the queue file, even if you are going to attempt immediate delivery. **sendmail** always instantiates the queue file before returning control to the client under any circumstances.

S*file* Log statistics in the named *file*.

t*tzinfo* Set the local time zone info to *tzinfo*, for example, "PST8PDT ". Actually, if this is not set, the **TZ** environment variable is cleared (so the system default is used); if set but null, the user’s **TZ** variable is used, and if set and non-null, the **TZ** variable is set to this value.

T*rtime*/*wtime* Set the queue timeout to *rtime*. After this interval, messages that have not been successfully sent will be returned to the sender. Defaults to five days (**5d**). The optional *wtime* is the time after which a warning message is sent. If it is missing or **0**, then no warning messages are sent.

u*n* Set the default userid for mailers to *n*. Mailers without the *S* flag in the mailer definition will run as this user. Defaults to **1**. The value can also be given as a symbolic user name.

v Run in verbose mode. If this is set, **sendmail** adjusts options **c** (do not connect to expensive mailers) and **d** (delivery mode) so that all mail is delivered completely in a single job so that you can see the entire delivery process. Option **v** should *never* be set in the configuration file; it is intended for command line use only.

V*fallbackhost* If specified, the *fallbackhost* acts like a very low priority MX on every host. This is intended to be used by sites with poor network connectivity.

w If you are the "best" (that is, lowest preference) MX for a given host, you should normally detect this situation and treat that condition specially, by forwarding the mail to a UUCP feed, treating it as local, or whatever. However, in some cases (such as Internet firewalls) you may want to try to connect directly to that host as though it had no MX records at all. Setting this option causes **sendmail** to try this. The downside is that errors in your configuration are likely to be diagnosed as "host unknown" or "message timed out" instead of something more meaningful. This option is deprecated.

x*LA* When the system load average exceeds *LA*, just queue messages (that is, do not try to send them). Defaults to **8**.

X*LA* When the system load average exceeds *LA*, refuse incoming SMTP connections. Defaults to **12**.

y*fact* The indicated factor *fact* is added to the priority (thus *lowering* the priority of the job) for each recipient, that is, this value penalizes jobs with large numbers of recipients. Defaults to **30000**.

Y If set, deliver each job that is run from the queue in a separate process. Use this option if you are short of memory, since the default tends to consume considerable amounts of memory while the queue is being processed.

z*fact* The indicated factor *fact* is multiplied by the message class (determined by the **Precedence:** field in the user header and the **P** lines in the configuration file) and subtracted from the priority. Thus, messages with a higher **Priority:** will be favored. Defaults to **1800**.

Z*fact* The indicated factor *fact* is added to the priority every time a job is processed. Thus, each time a job is processed, its priority will be decreased by the indicated value. In most environments this should be positive, since hosts that are down are all too often down for a long time. Defaults to **90000**.

7 Strip input to seven bits for compatibility with old systems. This should not be necessary.

| If set, **:include:** and **.forward** files that are group writable are considered "unsafe", that is, they cannot reference programs or write directly to files. World writable **:include:** and **.forward** files are always unsafe.

All options can be specified on the command line using the **–o** flag, but most will cause **sendmail** to relinquish its setuid permissions. The options that will not cause this are **b**, **d**, **e**, **E**, **i**, **L**, **m**, **o**, **p**, **r**, **s**, **v**, **C**, and **7**. Also considered "safe" is **M** (define macro) when defining the **r** or **s** macros.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep **sendmail** from suppressing the blanks from between arguments.

If invoked as **newaliases**, **sendmail** rebuilds the alias database. If invoked as *mailq*, **sendmail** prints the contents of the mail queue.

OPERANDS

address address of an intended recipient of the message being sent.

USAGE

See **largefile**(5) for the description of the behavior of **sendmail** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

sendmail returns an exit status describing what it did. The codes are defined in **/usr/include/sysexits.h**.

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall. Necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
X_OSERR	Temporary operating system error, such as “cannot fork”.
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

FILES

dead.letter	unmailable text
/etc/mail/sendmail.cf	defines environment for **sendmail**
/var/spool/mqueue/*	temp files and queued mail
˜/.forward	list of recipients for forwarding messages

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

biff(1B), **mail**(1), **mailx**(1), **newaliases**(1), **aliases**(4), **hosts**(4), **attributes**(5), **largefile**(5)

Su, Zaw-Sing, and Jon Postel, *The Domain Naming Convention for Internet User Applications*, RFC 819, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Postel, Jon, *Simple Mail Transfer Protocol*, RFC 821, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Crocker, Dave, *Standard for the Format of* ARPA-Internet *Text* Messages, RFC 822, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

NAME

server_upgrade – upgrade clients of a heterogeneous OS server

SYNOPSIS

server_upgrade –d <*install_image_dir*> [**–p** <*profile*>]

DESCRIPTION

Use this command to upgrade clients of a heterogeneous OS server that have different platforms (for example, SPARC or x86) or platform groups (for example, sun4d, sun4L) from the OS server. This command is necessary because clients of an OS server are not upgraded during a standard upgrade if the installation image does not support their platform or platform group.

Following are the steps for performing a standard upgrade of an OS server and upgrading clients, followed by the steps for upgrading clients with different platforms and platform groups using the **server_upgrade** command. The steps assume you are mounting the Solaris CD locally, remotely, or otherwise making it available.

1. Using the Solaris CD that matches the platform of the OS server, boot the OS server and perform a standard upgrade. Only clients that have the same platform and platform group supported on the installation image are upgraded. For example, when you boot a SPARC based server using a Solaris SPARC CD; all clients that are SPARC based and share the same platform group are upgraded.
2. Reboot the OS server.
3. Insert a platform-specific CD into the CD-ROM drive. For example, if the OS server is a SPARC based system, which shares services for both SPARC and x86 based clients, you would load the x86 Solaris CD at this point.
4. At the root prompt, type:

   server_upgrade –d <*install_image_dir*> [**–p** <*profile*>]

   The command upgrades the platform-specific services for clients on the OS server from the installation image.
5. Reboot the OS server.
6. Repeat steps 3-5 to upgrade platforms or platform groups of other clients.

OPTIONS

–p <*profile*>
Specify the full path to a custom JumpStart profile (a text file that defines how to install Solaris software on a system). For information on setting up a profile, see Installing Solaris Software.

NOTE: The profile must have the keyword set to upgrade.

–d <*install_image_dir*>
Specify the path to the installation image. For example, **/cdrom/cdrom0**.

EXAMPLES

The following example shows the states of a SPARC based OS server and its clients each time the **server_upgrade** command is used to upgrade clients. The scenario uses the **server_upgrade** command once to upgrade a client with an x86 platform, and once to upgrade a client with a different platform group (sun4L).

Initial State of Solaris

The OS server is a sparc.sun4d, running Solaris 2.4, sharing the following services:

> Solaris 2.4 for sparc.{sun4c, sun4d, sun4e, sun4m, and sun4L}
> Solaris 2.4 for i386.i86pc
> Solaris 2.3 for sparc.{sun4c, sun4d, sun4e, sun4m}

Initial client states are:

Host name	Is A ...	Running ...
red	sparc.sun4c	Solaris 2.4
blue	sparc.sun4e	Solaris 2.4
yellow	sparc.sun4L	Solaris 2.4
green	i386.i86pc	Solaris 2.4
purple	sparc.sun4c	Solaris 2.3
brown	sparc.sun4e	Solaris 2.3

Upgrade the 2.4 OS server to 2.5

Use the SPARC Solaris 2.5 CD to upgrade the Solaris 2.4 OS server, then reboot the OS server.

After rebooting the OS server, it is running Solaris 2.5 and sharing the following services:

> Solaris 2.5 for sparc.{sun4c, sun4d, sun4m}
> Solaris 2.3 for sparc.{sun4c, sun4d, sun4e, sun4m}

Client states are:

Host name	Is A ...	Running ...	And is Now...
* red	sparc.sun4c	Solaris 2.5	bootable
blue	sparc.sun4e	Solaris 2.4	not bootable
yellow	sparc.sun4L	Solaris 2.4	not bootable
green	i386.i86pc	Solaris 2.4	not bootable
* purple	sparc.sun4c	Solaris 2.5	bootable
* brown	sparc.sun4e	Solaris 2.3	bootable

NOTE: Client brown can still be booted because it is running Solaris 2.3, which is supported by the OS server, and because Solaris 2.3 supports sun4e.

Upgrade the x86 clients and services

Insert the x86 Solaris 2.5 CD and type:

```
server_upgrade -d /cdrom/cdrom0
```

After rebooting the OS server, it is running Solaris 2.5 and sharing the following services:

> Solaris 2.5 for sparc.{sun4c, sun4d, sun4m}
> Solaris 2.5 for i386.i86pc
> Solaris 2.3 for sparc.{sun4c, sun4d, sun4e, sun4m}

Client states are:

Host name	Is A ...	Running ...	And is Now...
red	sparc.sun4c	Solaris 2.5	bootable
blue	sparc.sun4e	Solaris 2.4	not bootable
yellow	sparc.sun4L	Solaris 2.4	not bootable
* green	i386.i86pc	Solaris 2.5	bootable

purple	sparc.sun4c	Solaris 2.5	bootable
brown	sparc.sun4e	Solaris 2.3	bootable

Upgrade the sun4L (Hardware Partner) client

Insert the Hardware Edition Solaris 2.5 CD and type:

```
server_upgrade -d /cdrom/cdrom0
```

After rebooting the OS server, it is running Solaris 2.5 and sharing the following services:

> Solaris 2.5 for sparc.{sun4c, sun4d, sun4m, sun4L}
> Solaris 2.5 for i386.i86pc
> Solaris 2.3 for sparc.{sun4c, sun4d, sun4e, sun4m}

Client states are:

Host name	Is A ...	Running ...	And is Now...
red	sparc.sun4c	Solaris 2.5	bootable
blue	sparc.sun4e	Solaris 2.4	not bootable
* yellow	sparc.sun4L	Solaris 2.5	bootable
green	i386.i86pc	Solaris 2.5	bootable
purple	sparc.sun4c	Solaris 2.5	bootable
brown	sparc.sun4e	Solaris 2.3	bootable

State of sun4e clients

Client blue is not bootable because sun4e systems are not supported by Solaris 2.5. However, it can be made bootable again by using the Solstice Host Manager and adding the Solaris 2.4 services to the OS server.

Client states are:

Host name	Is A ...	Running ...	And is Now...
red	sparc.sun4c	Solaris 2.5	bootable
* blue	sparc.sun4e	Solaris 2.4	not bootable
yellow	sparc.sun4L	Solaris 2.5	bootable
green	i386.i86pc	Solaris 2.5	bootable
purple	sparc.sun4c	Solaris 2.5	bootable
brown	sparc.sun4e	Solaris 2.3	bootable

NAME

setmnt – establish mount table

SYNOPSIS

/usr/sbin/setmnt

DESCRIPTION

setmnt creates the **/etc/mnttab** table which is needed for both the **mount** and **umount** commands. **setmnt** reads standard input and creates a **mnttab** entry for each line. Input lines have the format:

filesys node

where *filesys* is the name of the file system's ''special file'' (such as **/dev/dsk/c?d?s?**) and *node* is the root name of that file system. Thus *filesys* and *node* become the first two strings in the mount table entry.

FILES

/etc/mnttab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mount(1M), **attributes**(5)

BUGS

Problems may occur if *filesys* or *node* are longer than 32 characters.
setmnt silently enforces an upper limit on the maximum number of **mnttab** entries.

NAME

setuname – changes machine information

SYNOPSIS

setuname [–**t**][–**n** *node*][–**s** *name*]

DESCRIPTION

setuname changes the parameter value for the system name and node name. Each parameter can be changed using **setuname** and the appropriate option.

Either or both the –**s** and –**n** options must be given when invoking **setuname**.

The system architecture may place requirements on the size of the system and network node name. The command will issue a fatal warning message and an error message if the name entered is incompatible with the system requirements.

OPTIONS

–**t** Temporary change. No attempt will be made to create a permanent change.

–**n** *node* Changes the node name. *node* specifies the new network node name and can consist of alphanumeric characters and the special characters dash, underbar, and dollar sign.

–**s** *name* Changes the system name. *name* specifies new system name and can consist of alphanumeric characters and the special characters dash, underbar, and dollar sign.

NOTES

setuname attempts to change the parameter values in two places: the running kernel and, as necessary per implementation, to cross system reboots. A temporary change changes only the running kernel.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5)

NAME

share – make local resource available for mounting by remote systems

SYNOPSIS

share [–**F** *FSType*] [–**o** *specific_options*] [–**d** *description*] [*pathname*]

DESCRIPTION

The **share** command exports, or makes a resource available for mounting, through a remote file system of type *FSType*. If the option –**F** *FSType* is omitted, the first file system type listed in **/etc/dfs/fstypes** is used as default. For a description of NFS specific options, see **share_nfs**(1M). *pathname* is the pathname of the directory to be shared. When invoked with no arguments, **share** displays all shared file systems.

OPTIONS

–**F** *FSType* Specify the filesystem type.

–**o** *specific_options* The *specific_options* are used to control access of the shared resource. (See **share_nfs**(1M) for the NFS specific options.) They may be any of the following:

- **rw** *pathname* is shared read / write to all clients. This is also the default behavior.
- **rw=***client*[*:client*]... *pathname* is shared read / write only to the listed clients. No other systems can access *pathname*.
- **ro** *pathname* is shared read-only to all clients.
- **ro=***client*[*:client*]... *pathname* is shared read-only only to the listed clients. No other systems can access *pathname*.

–**d** *description* The –**d** flag may be used to provide a description of the resource being shared.

EXAMPLES

This line will share the **/disk** file system read-only at boot time.

```
share –F nfs –o ro /disk
```

FILES

/etc/dfs/dfstab	list of **share** commands to be executed at boot time
/etc/dfs/fstypes	list of file system types, NFS by default
/etc/dfs/sharetab	system record of shared file systems

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mountd(1M), **nfsd**(1M), **share_nfs**(1M), **shareall**(1M), **unshare**(1M), **attributes**(5)

NOTES

Export (old terminology): file system sharing used to be called exporting on SunOS 4.x, so the **share** command used to be invoked as **exportfs**(1B) or **/usr/sbin/exportfs**.

If **share** commands are invoked multiple times on the same filesystem, the last **share** invocation supersedes the previous—the options set by the last **share** command replace the old options. For example, if read-write permission was given to **usera** on **/somefs**, then to give read-write permission also to **userb** on **/somefs**:

```
example% share -F nfs -o rw=usera:userb /somefs
```

This behavior is not limited to sharing the root filesystem, but applies to all filesystems.

NAME

shareall, unshareall – share, unshare multiple resources

SYNOPSIS

shareall [–**F** *FSType*[,*FSType*...]] [– | *file*]
unshareall [–**F** *FSType*[,*FSType*...]]

DESCRIPTION

When used with no arguments, **shareall** shares all resources from *file*, which contains a list of **share** command lines. If the operand is a hyphen (–), then the **share** command lines are obtained from the standard input. Otherwise, if neither a *file* nor a hyphen is specified, then the file **/etc/dfs/dfstab** is used as the default.

Resources may be shared by specific file system types by specifying the file systems in a comma-separated list as an argument to –**F**.

unshareall unshares all currently shared resources. Without a –**F** flag, it unshares resources for all distributed file system types.

OPTIONS

–**F** *FSType* Specify file system type. Defaults to the first entry in **/etc/dfs/fstypes**.

FILES

/etc/dfs/dfstab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

share(1M), **unshare**(1M), **attributes**(5)

NAME

share_nfs – make local NFS file systems available for mounting by remote systems

SYNOPSIS

share [**–F nfs**] [**–o** *specific_options*] [**–d** *description*] *pathname*

DESCRIPTION

The **share** utility makes local file systems available for mounting by remote systems.

If no argument is specified, then **share** displays all file systems currently shared, including NFS file systems and file systems shared through other distributed file system packages.

OPTIONS

–F nfs Share NFS file system type.

–o *specific_options* Specify *specific_options* in a comma-separated list of keywords and attribute-value-assertions for interpretation by the file-system-type-specific command. If *specific_options* is not specified, then by default sharing will be read-write to all clients. *specific_options* can be any combination of the following:

aclok Allows the NFS server to do access control for NFS Version 2 clients (running SunOS 2.4 or earlier). When **aclok** is set on the server, maximal access is given to all clients. For example, with **aclok** set, if anyone has read permissions, then everyone does. If **aclok** is not set, minimal access is given to all clients.

anon=*uid*
Set *uid* to be the effective user ID of unknown users. By default, unknown users are given the effective user ID **UID_NOBODY**. If *uid* is set to –**1**, access is denied.

index=*file*
Load *file* rather than a listing of the directory containing this file when the directory is referenced by an NFS URL.

kerberos This option has been deprecated in favor of the **sec=***krb4* option.

nosub Prevents clients from mounting subdirectories of shared directories. For example, if **/export** is shared with the **nosub** option on server *fooey* then a NFS client will not be able to do:

```
mount -F nfs fooey:/export/home /mnt
```

nosuid By default, clients are allowed to create files on the shared file system with the setuid or setgid mode enabled. Specifying **nosuid** causes the server file system to silently ignore any attempt to enable the setuid or setgid mode bits.

public Enables NFS browsing of the file system by a WebNFS-enabled browser. Only one file system per server many use this option. The **-ro=**list and **-rw=**list options can be included with this option.

ro Sharing will be read-only to all clients.

ro=*client*[**:***client*]. . .
Sharing will be read-only to the listed clients; overrides the **rw** suboption for the clients specified. Netgroup names may be used in place of client names.

root=*host*[**:***host*] . . .
Only root users from the specified hosts will have root access. By default, no host has root access, so root users are mapped to an anonymous user ID (see the **anon=***uid* option described above). Netgroups can be used if the file system shared is using UNIX authentication (**AUTH_SYS**).

rw Sharing will be read-write to all clients.

rw=*client*[**:***client*]. . .
Sharing will be read-write to the listed clients; overrides the **ro** suboption for the clients specified. Netgroup names may be used in place of client names.

sec=*mode*[**:***mode*]. . .
Sharing will use one or more of the specified security modes. The *mode* in the **sec=***mode* option must be a node name supported on the client. If the **sec=** option is not specified, the default security mode used is **AUTH_SYS**. Multiple **sec=** options can be specified on the command line, although each mode can appear only once. The security modes are defined in **nfssec**(5).

Each **sec=** option specifies modes that apply to any subsequent **window=, rw, ro, rw=, ro=** and **root=** options that are provided before another **sec=**option. Each additional **sec=** resets the security mode context, so that more **window=, rw, ro, rw=, ro=** and **root=** options can be supplied for additional modes.

sec=*none* If the option **sec=***none* is specified when the client uses **AUTH_NONE**, or if the client uses a security mode that is not one that the file system is shared with, then the credential of each NFS request is treated as unauthenticated. See the **anon=***uid* option for a description of how unauthenticated requests are handled.

secure This option has been deprecated in favor of the **sec=***dh* option.

window=*value*
When sharing with **sec=***dh* or **sec=***krb4* set the maximum life time (in seconds) of the RPC request's credential (in the authentication header) that the NFS server will allow. If a credential arrives with a life time larger than what is allowed, the NFS server will reject the request. The default value is 30000 seconds (8.3 hours).

–d *description* Provide a comment that describes the file system to be shared.

OPERANDS

The following operands are supported:

pathname The pathname of the file system to be shared.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

FILES

/etc/dfs/fstypes	list of system types, NFS by default
/etc/dfs/sharetab	system record of shared file systems

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mount(1M), **mountd**(1M), **nfsd**(1M), **share**(1M), **unshare**(1M), **attributes**(5), **nfssec**(5)

NOTES

If the **sec=** option is presented at least once, all uses of the **window=, rw, ro, rw=, ro=** and **root=** options must come **after** the first **sec=** option. If the **sec=** option is not presented, then **sec=***sys* is implied.

If one or more explicit **sec=** options are presented, *sys* must appear in one of the options mode lists for accessing using the **AUTH_SYS** security mode to be allowed. For example:

```
share –F nfs /var
share –F nfs –o sec=sys /var
```

will grant read-write access to any host using **AUTH_SYS**, but

```
share –F nfs –o sec=dh /var
```

will grant no access to clients that use **AUTH_SYS**.

Unlike previous implementations of **share_nfs**(1M), access checking for the **window=, rw, ro, rw=,** and **ro=** options is done per NFS request, instead of per mount request.

Combining multiple security modes can be a security hole is situations where the **ro=** and **rw=** options are used to control access to weaker security modes. In this example,

```
share –F nfs –o sec=dh,rw,sec=sys,rw=hosta /var
```

an intruder can forge the IP address for **hosta** (albeit on each NFS request) to side-step the stronger controls of **AUTH_DES**. Something like:

```
share –F nfs –o sec=dh,rw,sec=sys,ro /var
```

is safer, because any client (intruder or legitimate) that avoids **AUTH_DES** will only get read-only access. In general, multiple security modes per **share** command should only be used in situations where the clients using more secure modes get stronger access than clients using less secure modes.

If **rw=,** and **ro=** options are specified in the same **sec=** clause, and a client is in both lists, the order of the two options determines the access the client gets. If client **hosta** is in two netgroups **-** **group1** and **group2** **-** in this example, the client would get read-only access:

```
share –F nfs –o ro=group1,rw=group2 /var
```

In this example **hosta** would get read-write access:

```
share –F nfs –o rw=group2,ro=group1 /var
```

If within a **sec=** clause, both the **ro** and **rw=** options are specified, for compatibility, the order of the options rule is not enforced. All hosts would get read-only access, with the exception to those in the read-write list. Likewise, if the **ro=** and **rw** options are specified, all hosts get read-write access with the exceptions of those in the read-only list.

The **ro=** and **rw=** options are guaranteed to work over UDP and TCP but may not work over other transport providers.

The **root=** option with **AUTH_SYS** is guaranteed to work over UDP and TCP but may not work over other transport providers.

The **root=** option with **AUTH_DES** and **AUTH_KERB** is guaranteed to work over any transport provider.

There are no interactions between the **root=** option and the **rw, ro, rw=,** and **ro=** options. Putting a host in the **root** list does not override the semantics of the other options. The access the host gets is the same as when the **root=** options is absent. For example, the following **share** command will deny access to **hostb:**

```
share –F nfs –o ro=hosta,root=hostb /var
```

The following will give read-only permissions to **hostb:**

```
share –F nfs –o ro=hostb,root=hostb /var
```

The following will give read-write permissions to **hostb:**

```
share –F nfs –o ro=hosta,rw=hostb,root=hostb /var
```

If the file system being shared is a symbolic link to a valid pathname, the canonical path (the path which the symbolic link follows) will be shared. For example, if **/export/foo** is a symbolic link to **/export/bar** (**/export/foo -> /export/bar**), the following **share** command will result in **/export/bar** as the shared pathname (and not **/export/foo**).

```
example# share –F nfs /export/foo
```

Note that an NFS mount of **server:/export/foo** will result in **server:/export/bar** really being mounted.

This line in the **/etc/dfs/dfstab** file will share the **/disk** file system read-only at boot time:

```
share –F nfs –o ro /disk
```

Note that the same command entered from the command line will not share the **/disk** file system unless there is at least one file system entry in the **/etc/dfs/dfstab** file. The **mountd**(1M) and **nfsd**(1M) daemons only run if there is a file system entry in **/etc/dfs/dfstab** when starting or rebooting the system.

NAME

showmount – show all remote mounts

SYNOPSIS

/usr/sbin/showmount [–**ade**] [*hostname*]

DESCRIPTION

showmount lists all the clients that have remotely mounted a filesystem from *host*. This information is maintained by the **mountd**(1M) server on *host*, and is saved across crashes in the file **/etc/rmtab**. The default value for *host* is the value returned by **hostname**(1).

OPTIONS

–**a** Print all remote mounts in the format:

hostname **: directory**

where *hostname* is the name of the client, and **directory** is the root of the file system that has been mounted.

–**d** List directories that have been remotely mounted by clients.

–**e** Print the list of shared file systems.

FILES

/etc/rmtab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

hostname(1), **mountd**(1M), **attributes**(5)

Solaris Advanced Installation Guide

BUGS

If a client crashes, its entry will not be removed from the list of remote mounts on the server.

NAME

showrev – show machine and software revision information

SYNOPSIS

/usr/bin/showrev [**–a**] [**–p**] [**–w**] [**–c** *command*] [**–s** *hostname*]

DESCRIPTION

showrev displays revision information for the current hardware and software. With no arguments, **showrev** shows the system revision information including hostname, hostid, release, kernel architecture, application architecture, hardware provider, domain, and kernel version.

If a command is supplied with the **–c** option, **showrev** shows the **PATH** and **LD_LIBRARY_PATH** and finds out all the directories within the **PATH** that contain it. For each file found, its file type, revision, permissions, library information, and checksum are printed as well.

OPTIONS

–a Print all system revision information available. Window system and patch information are added.

–p Print only the revision information about patches.

–w Print only the OpenWindows revision information.

–c *command* Print the revision information about *command*.

–s *hostname* Perform this operation on the specified *hostname*. The **–s** operation completes correctly only when *hostname* is running Solaris 2.5 or later.

OUTPUT

Varies, based on flags passed. If no flags are passed, output similar to the following appears:

```
Hostname: system1
Hostid: 7233808e
Release: 5.4
Kernel architecture: sun4m
Application architecture: sparc
Hardware provider: Sun_Microsystems
Domain: a.network.COM
Kernel version: SunOS 5.4 generic July 1994
```

EXIT STATUS

The following error values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmc

SEE ALSO

arch(1), **ldd**(1), **mcs**(1), **sum**(1), **attributes**(5)

BUGS

For the −**s** option to work when *hostname* is running a version of Solaris prior to 2.5, the Solstice AdminSuite must be installed on *hostname*.

NAME

shutdown – shut down system, change system state

SYNOPSIS

/usr/sbin/shutdown [**–y**] [**–g** *grace-period*] [**–i** *init-state*] [*message*]

DESCRIPTION

shutdown is executed by the super-user to change the state of the machine. In most cases, it is used to change from the multi-user state (state 2) to another state.

By default, **shutdown** brings the system to a state where only the console has access to the operating system. This state is called single-user.

Before starting to shut down daemons and killing processes, **shutdown** sends a warning message and, by default, a final message asking for confirmation. *message* is a string that is sent out following the standard warning message "The system will be shut down in . . ." If the string contains more than one word, it should be contained within single (**'**) or double (") quotation marks.

The warning message and the user provided *message* are output when there are 7200, 3600, 1800, 1200, 600, 300, 120, 60, and 30 seconds remaining before **shutdown** begins. See **EXAMPLES**.

System state definitions are:

state 0 Stop the operating system.

state 1 State 1 is referred to as the administrative state. In state 1 file systems required for multi-user operations are mounted, and logins requiring access to multi-user file systems can be used. When the system comes up from firmware mode into state 1, only the console is active and other multi-user (state 2) services are unavailable. Note that not all user processes are stopped when transitioning from multi-user state to state 1.

state s, S State s (or S) is referred to as the single-user state. All user processes are stopped on transitions to this state. In the single-user state, file systems required for multi-user logins are unmounted and the system can only be accessed through the console. Logins requiring access to multi-user file systems cannot be used.

state 5 Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible. The **rc0** procedure is called to perform this task.

state 6 Stop the operating system and reboot to the state defined by the **initdefault** entry in **/etc/inittab**. The **rc6** procedure is called to perform this task.

OPTIONS

–y Pre-answer the confirmation question so the command can be run without user intervention.

–g *grace-period* Allow the super-user to change the number of seconds from the 60-second default.

–i *init-state* If there are warnings, *init-state* specifies the state **init** is to be in. By default, system state λq**s**ρq is used.

EXAMPLES

In the following example, **shutdown** is being executed on host **foo** and is scheduled in 120 seconds. The warning message is output 2 minutes, 1 minute, and 30 seconds before the final confirmation message.

```
example# shutdown -i S -g 120 "===== disk replacement ====="



Shutdown started.   Tue Jun   7  14:51:40 PDT  1994



Broadcast Message from root (pts/1) on foo Tue Jun  7 14:51:41...
The system will be shut down in 2 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:52:41...
The system will be shut down in 1 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:53:41...
The system will be shut down in 30 seconds
===== disk replacement =====
Do you want to continue? (y or n):
```

FILES

/etc/inittab controls process dispatching by **init**

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

boot(1M), **halt**(1M), **init**(1M), **killall**(1M), **reboot**(1M), **ufsdump**(1M), **init.d**(4), **inittab**(4), **nologin**(4), **attributes**(5)

NAME

snmpdx – Sun Solstice Enterprise Master Agent

SYNOPSIS

/usr/lib/snmp/snmpdx [−**hy**] [−**a** *filename*] [−**c** *config-dir*] [−**d** *debug-level*]
[−**i** *filename*] [−**m GROUP** | −**m SPLIT**] [−**o** *filename*] [−**p** *port*] [−**r** *filename*]

DESCRIPTION

The Master Agent, **snmpdx**, is the main component of Solstice Enterprise Agent technology. It runs as a daemon process and listens to User Datagram Protocol (UDP) port 161 for SNMP requests. The Master Agent also opens another port to receive SNMP trap notifications from various subagents. These traps are forwarded to various managers, as determined by the configuration file.

Upon invocation, **snmpdx** reads its various configuration files and takes appropriate actions by activating subagents, determining the subtree Object Identifier (OID) for various subagents, populating its own Management Information Bases (MIBs), and so forth. The Master Agent invokes subagents, registers subagents, sends requests to subagents, receives responses from subagents, and traps notifications from subagents.

OPTIONS

The following options are supported:

−**a** *filename* Specify the full path of the access control file used by the Master Agent. The default access control file is **/etc/snmp/conf/snmpdx.acl**.

−**c** *config-dir* Specify the full path of the directory containing the Master Agent configuration files. The default directory is **/etc/snmp/conf**.

−**d** *debug-level* Debug. Levels from 0 to 4 are supported, giving various levels of debug information. The default is 0 which means no debug information is given.

−**h** Help. Print the command line usage.

−**i** *filename* Specify the full path of the enterprise-name OID map. This file contains the PID used by the Master Agent for recovery after a crash. It contains tuples of the UNIX process ID, port number, resource name, and agent name. The default file is **/var/snmp/snmpdx.st**.

−**m GROUP** | −**m SPLIT**
Specify the mode to use for forwarding of SNMP requests.

GROUP Multiple variables can be included in each request from the Master Agent to the subagents. This results in, at mose, one send-request per agent.

SPLIT Each variable in the incoming request results in one send-request to each subagent.

The default is **GROUP**.

−**o** *filename* Specify the full path of the file containing the tuple (enterprise-name, OID). For example, (Sun Microsystems, 1.3.1.6.1.4.32). The Master Agent uses this file as a base for look-up in the trap-filtering and forwarding process. The default file is **/etc/snmp/conf/enterprises.oid**.

−p *port* Specify the port number. The default port number is 161.

−r *filename* Specify the full path of the resource file to be used by the Master Agent. This file stores information about the subagents that the Master Agent invokes and manages. The default resource file is **/etc/snmp/conf/snmpdx.rsrc**.

−y Set a recovery indicator to invoke the recovery module. The recovery process discovers which subagents in the previous session are still active; those subagents not active are re-spawned by the Master Agent.

FILES

/var/snmp/conf/enterprises.oid	Enterprise-name OID map
/var/snmp/conf/snmpdx.acl	Access control file
/var/snmp/conf/snmpdx.rsrc	Resource configuration file
/var/snmp/snmpdx.st	Master Agent status file
/var/snmp/mib/snmpdx.mib	Master Agent MIB file

EXIT STATUS

The following error values are returned:

0 Successful completion.

non-zero An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsasnm

SEE ALSO

snmpd(1M), **snmpXdmid**(1M), **attributes**(5)

NAME

snmpXdmid – Sun Solstice Enterprise SNMP-DMI mapper subagent

SYNOPSIS

/usr/lib/dmi/snmpXdmid –s *hostname* [**–h**] [**–c** *config-dir*] [**–d** *debug-level*]

DESCRIPTION

The **snmpXdmid** utility is a subagent in the Solstice Enterprise Agent Desktop Management Interface package. It maps the SNMP requests forwarded by the Master Agent (**snmpdx**(1M)) into one or more equivalent DMI requests. Further, it remaps the DMI response into SNMP response back to **snmpdx**. By default, **snmpXdmid** also forwards the DMI indications as SNMP traps to **snmpdx**. The feature is configurable and can be disabled by setting **TRAP_FORWARD_TO_MAGENT=0** in the **snmpXdmid** configuration file, **snmpXdmid.conf**.

This subagent runs as a daemon in the system. The subagent uses a set of .MAP files located in **/var/dmi/map** to map the SNMP Object Identifier (OID) into a corresponding DMI component. The map files are generated using the MIF-to-MIB utility, **miftomib**. They are read by **snmpXdmid** when a corresponding MIF file gets registered with the DMI Service Provider (**dmispd**(1M)).

The **snmpXdmid.conf** file is used for configuration information. Each entry in the file consists of a keyword followed by an equal sign (=), followed by a parameter string. The keyword must begin in the first position. A line beginning with a pound sign (#) is treated as a comment and the subsequent characters on that line are ignored. The keywords currently supported are:

WARNING_TIMESTAMP
Indication subscription expiration, warning time.

EXPIRATION_TIMESTAMP
Indication subscription expiration timestamp.

FAILURE_THRESHOLD
DMISP retries before dropping indication due to comm errors.

TRAP_FORWARD_TO_MAGENT

0	Drop indication at the subagent level.
non-zero	Forward indications as SNMP traps to **snmpdx**.

By default, the configuration file **snmpXdmid.conf** is located in the **/etc/dmi/conf** directory. You can specify an alternative directory with the –**c** option.

OPTIONS

The following options are supported:

–**c** *config-dir* Specify the directory where **snmpXdmid.conf** file is located.

–**d** *debug-level* Debug. Levels from 1 to 5 are supported, giving various levels of debug information.

–**h** Help. Print the command line usage.

–**s** *hostname* Specify the host on which **dmispd** is running.

FILES

/etc/dmi/conf/snmpXdmid.conf DMI mapper configuration file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi

SEE ALSO

dmispd(1M), **snmpd**(1M), **snmpdx**(1M), **attributes**(5)

NAME

snoop – capture and inspect network packets

SYNOPSIS

snoop [–**aCDNPSvV**] [–**t** [**r** | **a** | **d**]] [–**c** *maxcount*] [–**d** *device*]
[–**i** *filename*] [–**n** *filename*] [–**o** *filename*] [–**p** *first* [, *last*]] [–**s** *snaplen*]
[–**x** *offset* [, *length*]] [*expression*]

DESCRIPTION

snoop captures packets from the network and displays their contents. **snoop** uses both the network packet filter and streams buffer modules to provide efficient capture of packets from the network. Captured packets can be displayed as they are received, or saved to a file for later inspection.

snoop can display packets in a single-line summary form or in verbose multi-line forms. In summary form, only the data pertaining to the highest level protocol is displayed. For example, an NFS packet will have only NFS information displayed. The underlying RPC, UDP, IP, and ethernet frame information is suppressed but can be displayed if either of the verbose options are chosen.

snoop requires an interactive interface.

OPTIONS

–**a** Listen to packets on **/dev/audio** (warning: can be noisy).

–**C** List the code generated from the filter expression for either the kernel packet filter, or **snoop**'s own filter.

–**D** Display number of packets dropped during capture on the summary line.

–**N** Create an IP address-to-name file from a capture file. This must be set together with the –**i** option that names a capture file. The address-to-name file has the same name as the capture file with **.names** appended. This file records the IP address to hostname mapping at the capture site and increases the portability of the capture file. Generate a **.names** file if the capture file is to be analyzed elsewhere. Packets are not displayed when this flag is used.

–**P** Capture packets in non-promiscuous mode. Only broadcast, multicast, or packets addressed to the host machine will be seen.

–**S** Display size of the entire ethernet frame in bytes on the summary line.

–**v** Verbose mode. Print packet headers in lots of detail. This display consumes many lines per packet and should be used only on selected packets.

–**V** Verbose summary mode. This is halfway between summary mode and verbose mode in degree of verbosity. Instead of displaying just the summary line for the highest level protocol in a packet, it displays a summary line for each protocol layer in the packet. For instance, for an NFS packet it will display a line each for the ETHER, IP, UDP, RPC and NFS layers. Verbose summary mode output may be easily piped through **grep** to extract packets of interest. For example to view only RPC summary lines:

```
example# snoop –i rpc.cap –V | grep RPC
```

–t [**r** | **a** | **d**] Time-stamp presentation. Time-stamps are accurate to within 4 microseconds. The default is for times to be presented in **d** (delta) format (the time since receiving the previous packet).
Option **a** (absolute) gives wall-clock time.
Option **r** (relative) gives time relative to the first packet displayed. This can be used with the –**p** option to display time relative to any selected packet.

–c *maxcount* Quit after capturing *maxcount* packets. Otherwise keep capturing until there is no disk left or until interrupted with CTRL-C.

–d *device* Receive packets from the network using the interface specified by *device*. Usually **le0** or **ie0**. The program **netstat**(1M), when invoked with the –**i** flag, lists all the interfaces that a machine has. Normally, **snoop** will automatically choose the first non-loopback interface it finds.

–i *filename* Display packets previously captured in *filename*. Without this option, **snoop** reads packets from the network interface. If a *filename***.names** file is present, it is automatically loaded into **snoop**'s IP address-to-name mapping table (See –**N** flag).

–n *filename* Use *filename* as an IP address-to-name mapping table. This file must have the same format as the **/etc/hosts** file (IP address followed by the hostname).

–o *filename* Save captured packets in *filename* as they are captured. During packet capture, a count of the number of packets saved in the file is displayed. If you wish just to count packets without saving to a file, name the file **/dev/null**.

–p *first* [, *last*]
Select one or more packets to be displayed from a capture file. The *first* packet in the file is packet #1.

–s *snaplen* Truncate each packet after *snaplen* bytes. Usually the whole packet is captured. This option is useful if only certain packet header information is required. The packet truncation is done within the kernel giving better utilization of the streams packet buffer. This means less chance of dropped packets due to buffer overflow during periods of high traffic. It also saves disk space when capturing large traces to a capture file. To capture only IP headers (no options) use a *snaplen* of 34. For UDP use 42, and for TCP use 54. You can capture RPC headers with a *snaplen* of 80 bytes. NFS headers can be captured in 120 bytes.

–x *offset* [, *length*]
Display packet data in hexadecimal and ASCII format. The *offset* and *length* values select a portion of the packet to be displayed. To display the whole packet, use an *offset* of 0. If a *length* value is not provided, the rest of the packet is displayed.

OPERANDS

expression
Select packets either from the network or from a capture file. Only packets for which the expression is true will be selected. If no expression is provided it is assumed to be true.

Given a filter expression, **snoop** generates code for either the kernel packet filter or for its own internal filter. If capturing packets with the network interface, code for the kernel packet filter is generated. This filter is implemented as a streams module, upstream of the buffer module. The buffer module accumulates packets until it becomes full and passes the packets on to **snoop**. The kernel packet filter is very efficient, since it rejects unwanted packets in the kernel before they reach the packet buffer or **snoop**. The kernel packet filter has some limitations in its implementation — it is possible to construct filter expressions that it cannot handle. In this event, **snoop** generates code for its own filter. The –**C** flag can be used to view generated code for either the kernel's or **snoop 's** own packet filter. If packets are read from a capture file using the –**i** option, only **snoop**'s packet filter is used.

A filter *expression* consists of a series of one or more boolean primitives that may be combined with boolean operators (AND, OR, and NOT). Normal precedence rules for boolean operators apply. Order of evaluation of these operators may be controlled with parentheses. Since parentheses and other filter expression characters are known to the shell, it is often necessary to enclose the the filter expression in quotes. The primitives are:

host *hostname*
True if the source or destination address is that of *hostname*. The keyword **host** may be omitted if the name does not conflict with the name of another expression primitive e.g. "**pinky**" selects packets transmitted to or received from the host **pinky** whereas "**pinky and dinky**" selects packets exchanged between hosts **pinky** AND **dinky**. Normally the IP address is used. With the **ether** qualifier the ethernet address is used, for instance, "**ether pinky**".

ipaddr or *etheraddr*
Literal addresses, both IP dotted and ethernet colon are recognized. For example, "**129.144.40.13**" matches all packets with that IP address as source or destination, and similarly, "**8:0:20:f:b1:51**" matches all packets with the ethernet address as source or destination. An ethernet address beginning with a letter is interpreted as a hostname. To avoid this, prepend a zero when specifying the address. For example, if the ethernet address is "aa:0:45:23:52:44", then specify it by add a leading zero to

make it "0aa:0:45:23:52:44".

from or **src**
A qualifier that modifies the following **host**, **net**, *ipaddr*, *etheraddr*, **port** or **rpc** primitive to match just the source address, port, or RPC reply.

to or **dst**
A qualifier that modifies the following **host**, **net**, *ipaddr*, *etheraddr*, **port** or **rpc** primitive to match just the destination address, port, or RPC call.

ether A qualifier that modifies the following **host** primitive to resolve a name to an ethernet address. Normally, IP address matching is performed.

ethertype *number*
True if the ethernet type field has value *number*.
Equivalent to "**ether[12:2] =** *number*".

ip, **arp**, **rarp**
True if the packet is of the appropriate ethertype.

broadcast
True if the packet is a broadcast packet.
Equivalent to "**ether[2:4] = 0xffffffff**".

multicast
True if the packet is a multicast packet.
Equivalent to "**ether[0] & 1 = 1**".

apple True if the packet is an Apple Ethertalk packet.
Equivalent to "**ethertype 0x809b or ethertype 0x803f**".

decnet True if the packet is a DECNET packet.

greater *length*
True if the packet is longer than *length*.

less *length*
True if the packet is shorter than *length*.

udp, **tcp**, **icmp**
True if the IP protocol is of the appropriate type.

net *net* True if either the IP source or destination address has a network number of *net*. The **from** or **to** qualifier may be used to select packets for which the network number occurs only in the source or destination address.

port *port*
True if either the source or destination port is *port*. The *port* may be either a port number or name from **/etc/services**. The **tcp** or **udp** primitives may be used to select TCP or UDP ports only. The **from** or **to** qualifier may be used to select packets for which the *port* occurs only as the source or destination.

rpc *prog* [, *vers* [, *proc*]]
True if the packet is an RPC call or reply packet for the protocol identified by *prog*. The *prog* may be either the name of an RPC protocol from

/etc/rpc or a program number. The *vers* and *proc* may be used to further qualify the program *version* and *procedure* number, for example, "**rpc nfs,2,0**" selects all calls and replies for the NFS null procedure. The **to** or **from** qualifier may be used to select either call or reply packets only.

gateway *host*
True if the packet used *host* as a gateway, that is, the ethernet source or destination address was for *host* but not the IP address.
Equivalent to "**ether host** *host* and not host *host*".

nofrag True if the packet is unfragmented or is the first in a series of IP fragments.
Equivalent to "**ip[6:2] & 0x1fff = 0**".

expr relop expr
True if the relation holds, where *relop* is one of **>**, **<**, **>=**, **<=**, **=**, **!=**, and *expr* is an arithmetic expression composed of numbers, packet field selectors, the **length** primitive, and arithmetic operators **+**, **–**, ∗, **&**, **|**, **^**, and **%**. The arithmetic operators within *expr* are evaluated before the relational operator and normal precedence rules apply between the arithmetic operators, such as multiplication before addition. Parentheses may be used to control the order of evaluation. To use the value of a field in the packet use the following syntax:

base[*expr* [**:** *size*]]

where *expr* evaluates the value of an offset into the packet from a *base* offset which may be **ether**, **ip**, **udp**, **tcp**, or **icmp**. The *size* value specifies the size of the field. If not given, 1 is assumed. Other legal values are 2 and 4.
Examples:

"**ether[0] & 1 = 1**" is equivalent to **multicast**.

"**ether[2:4] = 0xffffffff** " is equivalent to **broadcast**.

"**ip[ip[0] & 0xf** ∗ **4 : 2] = 2049**" is equivalent to "**udp[0:2] = 2049**".

"**ip[0] & 0xf > 5**" selects IP packets with options.

"**ip[6:2] & 0x1fff = 0**" eliminates IP fragments.

"**udp and ip[6:2]&0x1fff = 0 and udp[6:2] != 0**" finds all packets with UDP checksums.

The **length** primitive may be used to obtain the length of the packet. For instance "**length > 60**" is equivalent to "**greater 60**", and "**ether[length – 1]**" obtains the value of the last byte in a packet.

and Perform a logical AND operation between two boolean values. The AND operation is implied by the juxtaposition of two boolean expressions, for example "**dinky pinky**" is the same as "**dinky AND pinky**".

or or **,** Perform a logical OR operation between two boolean values. A comma may be used instead, for example, "**dinky,pinky**" is the same as "**dinky OR pinky**".

not or **!** Perform a logical NOT operation on the following boolean value. This operator is evaluated before AND or OR.

EXAMPLES

Capture all packets and display them as they are received:

```
example# snoop
```

Capture packets with host **funky** as either the source or destination and display them as they are received:

```
example# snoop funky
```

Capture packets between **funky** and **pinky** and save them to a file. Then inspect the packets using times (in seconds) relative to the first captured packet:

```
example# snoop –o cap  funky pinky
example$ snoop –i cap  –t r | more
```

Look at selected packets in another capture file:

```
example$ snoop –i pkts  –p99,108
 99   0.0027   boutique -> sunroof     NFS C GETATTR FH=8E6C
100   0.0046   sunroof -> boutique     NFS R GETATTR OK
101   0.0080   boutique -> sunroof     NFS C RENAME FH=8E6C MTra00192 to .nfs08
102   0.0102   marmot -> viper         NFS C LOOKUP FH=561E screen.r.13.i386
103   0.0072   viper -> marmot         NFS R LOOKUP No such file or directory
104   0.0085   bugbomb -> sunroof      RLOGIN C PORT=1023 h
105   0.0005   kandinsky -> sparky     RSTAT C Get Statistics
106   0.0004   beeblebrox -> sunroof   NFS C GETATTR FH=0307
107   0.0021   sparky -> kandinsky     RSTAT R
108   0.0073   office -> jeremiah      NFS C READ FH=2584 at 40960 for 8192
```

Packet 101 Looks interesting. Take a look in more detail:

```
example$ snoop –i pkts  –v –p101
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 101 arrived at 16:09:53.59
ETHER:  Packet size = 210 bytes
ETHER:  Destination = 8:0:20:1:3d:94, Sun
ETHER:  Source      = 8:0:69:1:5f:e,  Silicon Graphics
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4, header length = 20 bytes
IP:   Type of service = 00
IP:         ..0. .... = routine
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 196 bytes
IP:   Identification 19846
IP:   Flags = 0X
IP:   .0.. .... = may fragment
IP:   ..0. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
```

```
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 18DC
IP:   Source address = 129.144.40.222, boutique
IP:   Destination address = 129.144.40.200, sunroof
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 1023
UDP:  Destination port = 2049 (Sun RPC)
UDP:  Length = 176
UDP:  Checksum = 0
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 665905
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100003 (NFS), version = 2, procedure = 1
RPC:  Credentials: Flavor = 1 (Unix), len = 32 bytes
RPC:     Time = 06-Mar-90 07:26:58
RPC:     Hostname = boutique
RPC:     Uid = 0, Gid = 1
RPC:     Groups = 1
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:
NFS:  ----- SUN NFS -----
NFS:
NFS:  Proc = 11 (Rename)
NFS:  File handle = 000016430000000100080000305A1C47
NFS:                597A0000000800002046314AFC450000
NFS:  File name = MTra00192
NFS:  File handle = 000016430000000100080000305A1C47
NFS:                597A0000000800002046314AFC450000
NFS:  File name = .nfs08
NFS:
```

View just the NFS packets between **sunroof** and **boutique**:

```
example$ snoop –i pkts  rpc nfs and sunroof and boutique
  1   0.0000   boutique -> sunroof    NFS C GETATTR FH=8E6C
  2   0.0046    sunroof -> boutique   NFS R GETATTR OK
  3   0.0080   boutique -> sunroof    NFS C RENAME FH=8E6C MTra00192 to .nfs08
```

Save these packets to a new capture file:

```
$ snoop –i pkts –o pkts.nfs rpc nfs sunroof boutique
```

EXIT STATUS

0	Successful completion.
1	An error occurred.

FILES

/dev/audio	symbolic link to the system's primary audio device
/dev/null	the null file
/etc/hosts	host name database
/etc/rpc	rpc program number data base
/etc/services	Internet services and aliases

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

netstat(1M), **hosts**(4), **rpc**(4), **services**(4), **attributes**(5), **audio**(7I), **bufmod**(7M), **dlpi**(7P), **ie**(7D), **le**(7D), **pfmod**(7M)

WARNINGS

The processing overhead is much higher for realtime packet interpretation. Consequently, the packet drop count may be higher. For more reliable capture, output raw packets to a file using the **–o** option and analyze the packets off-line.

Unfiltered packet capture imposes a heavy processing load on the host computer— particularly if the captured packets are interpreted realtime. This processing load further increases if verbose options are used. Since heavy use of **snoop** may deny computing resources to other processes, it should not be used on production servers. Heavy use of **snoop** should be restricted to a dedicated computer.

snoop does not reassemble IP fragments. Interpretation of higher level protocol halts at the end of the first IP fragment.

snoop may generate extra packets as a side-effect of its use. For example it may use a network name service (NIS or NIS+) to convert IP addresses to host names for display. Capturing into a file for later display can be used to postpone the address-to-name mapping until after the capture session is complete. Capturing into an NFS-mounted file may also generate extra packets.

Setting the **snaplen** (**–s** option) to small values may remove header information that is needed to interpret higher level protocols. The exact cutoff value depends on the network and protocols being used. For NFS Version 2 traffic using UDP on 10 Mb / s ethernet, do not set **snaplen** less than 150 bytes. For NFS Version 3 traffic using TCP on 100 Mb / s ethernet, **snaplen** should be 250 bytes or more.

snoop requires information from an RPC request to fully interpret an RPC reply. If an RPC reply in a capture file or packet range does not have a request preceding it, then only the RPC reply header will be displayed.

NAME

soconfig – configure transport providers for use by sockets

SYNOPSIS

/sbin/soconfig –f *file*
/sbin/soconfig *family type protocol* [*path*]

DESCRIPTION

The **soconfig** utility configures the transport provider driver for use with sockets. It specifies how the family, type, and protocol parameters in the **socket**(3N) call are mapped to the name of a transport provider such as **/dev/tcp**. This utility can be used to add an additional mapping or remove a previous mapping.

The **init**(1M) utility uses **soconfig** with the **sock2path**(4) file during the booting sequence.

OPTIONS

The following options are supported:

–f *file* Set up the **soconfig** configuration for each driver according to the information stored in *file*. A **soconfig** file consists of lines of at least the first three fields listed below, separated by spaces:

family type protocol path

These fields are described in the **OPERANDS** section below.

An example of *file* can be found in the **EXAMPLES** section below.

OPERANDS

The following operands are supported:

family The protocol family as listed in the **/usr/include/sys/socket.h** file, expressed as an integer.

type The socket type as listed in the **/usr/include/sys/socket.h** file, expressed as an integer.

protocol The protocol number as specified in the family-specific **include** file, expressed as an integer. For example, for **AF_INET** this number is specified in **/usr/include/netinet/in.h**. An unspecified protocol number is denoted with the value zero.

path The string that specifies the path name of the device that corresponds to the transport provider. If this parameter is specified, the configuration will be added for the specified family, type, and protocol. If this parameter is not specified, the configuration will be removed.

EXAMPLES

The following example sets up **/dev/tcp** for family **AF_INET** and type **SOCK_STREAM**:

```
example# soconfig 2 2 0 /dev/tcp
```

The following is a sample file used with the **–f** option. Comment lines begin with a number sign (**#**):

```
#    Family  Type  Protocol    Path
         2       2              0          /dev/tcp
         2       2              6          /dev/tcp

         2       1              0          /dev/udp
         2       1              17         /dev/udp

         1       2              0          /dev/ticotsord
         1       1              0          /dev/ticlts

         2       4              0          /dev/rawip
```

FILES

/etc/sock2path file containing mappings from sockets to transport providers

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

init(1M), **socket**(3N), **sock2path**(4), **attributes**(5)

Network Interfaces Programmer's Guide

NAME

soladdapp – add an application to the Solstice application registry

SYNOPSIS

/usr/snadm/bin/soladdapp [**–r** *registry*] **–n** *name* **–i** *icon* **–e** *executable* [*args*]

DESCRIPTION

soladdapp adds an application to the Solstice application registry. After it is added, the application is displayed in the Solstice Launcher main window (see **solstice**(1M)).

OPTIONS

–r *registry*	Define the full path name of the Solstice registry file.
–n *name*	Define the name of the tool to be registered.
–i *icon*	Define the full path name of the tool icon.
–e *executable*	Define the full path name of the tool.
args	Specify any arguments to use with the tool.

When executed without options, **soladdapp** uses **/opt/SUNWadm/etc/.solstice_registry** (the default registry path).

RETURN VALUES

0	on success
1	on failure
2	if the registry is locked
3	if the entry is a duplicate.

EXAMPLES

The following adds an application called Disk Manager to the Solstice application registry for display in the Solstice Launcher main window.

```
# soladdapp -r /opt/SUNWadm/etc/.solstice_registry -n "Disk Manager"
-i /opt/SUNWdsk/etc/diskmgr.xpm -e /opt/SUNWdsk/bin/diskmgr
```

FILES

/opt/SUNWadm/etc/.solstice_registry	The default registry path.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

SEE ALSO

soldelapp(1M), **solstice**(1M), **attributes**(5)

NOTES

Globally registered applications are used by local and remote users sharing the software in a particular **/opt** directory. They can be added only using **soladdapp**.

NAME

soldelapp – remove an application from the Solstice application registry

SYNOPSIS

/usr/snadm/bin/soldelapp [**–r** *registry*] **–n** *name*

DESCRIPTION

soldelapp removes an application from the Solstice application registry. After removal, the application is no longer displayed in the Solstice Launcher main window (see **solstice**(1M)).

OPTIONS

–r *registry* Define the full path name of the Solstice registry file.

–n *name* Define the name of the tool to be removed.

When executed without options, **soldelapp** uses **/opt/SUNWadm/etc/.solstice_registry** (the default registry path).

RETURN VALUES

0 on success

1 on failure

2 if the registry is locked

3 if *name* is not found in the registry

4 if the named registry or default registry is not found

EXAMPLES

The following removes an application called Disk Manager from the Solstice application registry and the Solstice Launcher main window.

```
# soldelapp -r /opt/SUNWadm/etc/.solstice_registry -n "Disk Manager"
```

FILES

/opt/SUNWadm/etc/.solstice_registry The default registry file.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

SEE ALSO

soladdapp(1M), **solstice**(1M), **attributes**(5)

NOTES

Globally registered applications are used by local and remote users sharing the software in a particular **/opt** directory. They can be removed only using **soldelapp**.

NAME

solstice – access system administration tools with a graphical user interface

SYNOPSIS

/bin/solstice

DESCRIPTION

solstice used on a system presents the **Solstice Launcher,** a graphical user interface that provides access to the Solstice AdminSuite product family of system administration tools. The tools that appear in the launcher depend on what Solstice products you installed on your system.

Help is available by using the Help button.

USAGE

The **Solstice Launcher** allows you to do the following tasks:

Launch applications Use the **Solstice Launcher** to launch system administration tools.

Register applications Use the **Solstice Launcher** to add and register applications locally with the launcher.

Remove applications Use the **Solstice Launcher** to remove locally registered applications.

Customize application properties
Use the **Solstice Launcher** to show, hide, or remove applications in the launcher, reorder the icons, change the launcher window width, modify applications properties, and add applications.

FILES

/$HOME/.solstice_registry Local registry information.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

SEE ALSO

soladdapp(1M), **soldelapp**(1M), **attributes**(5)

NOTES

The **Solstice Launcher** adds or removes local applications that are private to the user (not local to the system) only. The properties of globally registered applications that are used by local and remote users sharing the software from a particular **/opt** directory cannot be modified from the **Solstice Launcher.** To register global applications for use by local and remote users, use the **soladdapp**(1M) command. To remove globally registered applications, use the **soldelapp**(1M) command.

NAME

spray – spray packets

SYNOPSIS

/usr/sbin/spray [**–c** *count*] [**–d** *delay*] [**–l** *length*] [**–t** *nettype*] *host*

DESCRIPTION

spray sends a one-way stream of packets to *host* using RPC, and reports how many were received, as well as the transfer rate. The *host* argument can be either a name or an Internet address.

spray is not useful as a networking benchmark as it uses unreliable connectionless transports, (upd for example). **spray** can report a large number of packets dropped when the drops were caused by **spray** sending packets faster than they can be buffered locally (before the packets get to the network medium).

OPTIONS

–c *count* Specify how many packets to send. The default value of *count* is the number of packets required to make the total stream size 100000 bytes.

–d *delay* Specify how many microseconds to pause between sending each packet. The default is **0**.

–l *length* The *length* parameter is the numbers of bytes in the Ethernet packet that holds the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32 bit quantities, not all values of *length* are possible, and **spray** rounds up to the nearest possible value. When *length* is greater than 1514, then the RPC call can no longer be encapsulated in one Ethernet packet, so the *length* field no longer has a simple correspondence to Ethernet packet size. The default value of *length* is **86 bytes** (the size of the RPC and UDP headers).

–t *nettype* Specify class of transports. Defaults to **netpath**. See **rpc**(3N) for a description of supported classes.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

rpc(3N), **attributes**(5)

NAME

ssaadm – administration program for the SPARCstorage Array

SYNOPSIS

ssaadm [**–v**] *subcommand* [*subcommand_option* . . .] *pathname* . . .

DESCRIPTION

The **ssaadm** program is an administrative command that manages the SPARCstorage Array. **ssaadm** performs a variety of control and query tasks depending on the command line arguments and options used.

The command line must contain a *subcommand* (listed under **USAGE**) and at least one *pathname*. It may also contain options and other parameters depending on the subcommand. The subcommand is applied to each of the pathnames on the command line.

pathname specifies the SPARCstorage Array controller or a disk in the SPARCstorage Array. The controller name is specified by its physical name, for example,

/devices/. . . /. . . /SUNW,soc@3,0/SUNW,pln@*axxxxxxx***,***xxxxxxxx***:ctlr**

or by a name of the form

c*N*

where *N* is the logical controller number. **ssaadm** uses the **c***N* name to find an entry in the **/dev/rdsk** directory of a disk that is attached to the SPARCstorage Array controller. The **/dev/rdsk** entry is then used to determine the physical name of the SPARCstorage Array controller. A disk in the SPARCstorage Array is specified by its logical or physical device name, for example,

/dev/rdsk/c1t0d0s2

or

/devices/. . . /. . . /SUNW,soc@3,0/SUNW,pln@*axxxxxxx***,***xxxxxxxx***/ssd@0,0:c,raw**

See **disks**(1M) for more information on logical names for disks and controllers.

OPTIONS

The following options are supported:

–v Verbose mode.

Subcommands and their options are described below.

OPERANDS

The following operands are supported:

pathname The SPARCstorage Array controller or a disk in the SPARCstorage Array.

USAGE

Subcommands

display [**–p**] *pathname* . . .

Display configuration information for the specified unit(s) or display performance information for the specified SPARCstorage Array controller. If *pathname* specifies the controller, the configuration information is displayed for all disks in the SPARCstorage Array. For each drive that has fast write enabled, “(FW)” will be displayed after the drive identification.

–p Display performance information for the specified SPARCstorage Array controller. The accumulation of the performance statistics must be enabled using the **perf_statistics** subcommand before displaying the performance information. If not enabled, all of the I/Os per second will be displayed as zeros.

The performance display reports the following information:

BUSY How busy the controller in the SPARCstorage Array is, expressed as a percentage.

IOPS The total I/Os per second for the SPARCstorage Array.

entries for each disk
The total number of I/Os per second.

download –**f** *filename pathname*
download –**w** *wwn pathname*
Download an image to the SPARCstorage Array controller.

–**f** Download the prom image specified by *filename* to the SPARCstorage Array controller FEPROMs. When the download is complete, the SPARCstorage Array must be reset in order to use the downloaded code. Note that the **download** subcommand modifies the FEPROM on the SPARCstorage Array and should be used *with caution*.

–**w** Change the SPARCstorage Array controller's World Wide Name. *wwn* is a 12 digit hex number, leading zeros required. The new SPARCstorage Array controller's image will have the least significant 6 bytes of the 8-byte World Wide Name modified to *wwn*.

fast_write [–**s**] –**c** *pathname*
fast_write [–**s**] –**d** *pathname*
fast_write [–**s**] –**e** *pathname*
Enable or disable the use of the NVRAM to enhance the performance of writes in the SPARCstorage Array. *pathname* may refer to the SPARCstorage Array controller or to an individual disk.

–**s** Save the state that is currently being requested so it will persist across power-cycles.

–**c** Enable fast writes for synchronous writes only.

–**e** Enable fast writes.

–**d** Disable fast writes.

fc_s_download [–**f** *fcode-file*]
Download the fcode contained in the file *fcode-file* into *all* the FC/S Sbus Cards. This command is interactive and expects user confirmation before downloading the fcode. When invoked without the [–**f** *fcode-file*] option, the current version of the fcode in each FC/S Sbus card is printed.
Note that the **fc_s_download** subcommand should be used *only in single-user mode*; otherwise the FC/S card could be reset.

perf_statistics –**e** *pathname*
perf_statistics –**d** *pathname*

Enable or disable the accumulation of performance statistics for the specified SPARCstorage Array controller. The accumulation of performance statistics must be enabled before using the **display** –**p** subcommand. This subcommand can be issued only to the SPARCstorage Array controller.

–**e** Enable the accumulation of performance statistics.

–**d** Disable the accumulation of performance statistics.

purge *pathname*

Purge any fast write data from NVRAM for one disk, or all disks if the controller is specified. This option should be used with caution, usually only when a drive has failed.

reserve *pathname*

Reserve the specified controller(s) or disk(s) for exclusive use by the issuing host. When HA (High_Availability) Software is running on a system, do not use this subcommand to reserve a disk on an SSA. Doing so could cause problems for the HA software.

release *pathname*

Release a reservation held on the specified controller(s) or disk(s). When HA (High_Availability) Software is running on a system, do not use this subcommand to release a disk on an SSA. Doing so could cause problems for the HA software.

set_boot_dev [–**y**] *pathname*

Set the boot-device variable in the PROM to the physical device name specified by *pathname* which can be a block special device or a mount-point. The command normally runs interactively and requests confirmation for setting the default boot-device in the PROM. The –**y** option can be used to run it in non-interactive mode, in which case no confirmation is requested or required.

start [–**t** *tray-number*] *pathname*

Spin up the specified disk(s). If *pathname* specifies the controller, this action applies to all disks in the SPARCstorage Array.

–**t** Spin up all disks in the tray specified by *tray-number*. *pathname* must specify the controller.

stop [–**t** *tray-number*] *pathname*

Spin down the specified disk(s). If *pathname* specifies the controller, this action applies to all disks in the SPARCstorage Array.

–**t** Spin down all disks in the tray specified by *tray-number*. *pathname* must specify the controller.

sync_cache *pathname*
Flush all outstanding writes for the specified disk from NVRAM to the media. If *pathname* specifies the controller, this action applies to all disks in the SPARCstorage Array.

SCSI Enclosure Services (SES) Commands

The SPARCstorage RSM tray is addressed by the using the logical or physical path of the SES device or specifying the controller followed by the tray number if that controller has multiple trays. The controller is addressed by **c***N* or the physical path to the SPARCstorage Array's controller.

See **ses**(7D) for more information about environmental sensor cards and associated devices.

These commands also work with RSM trays directly attached to wide differential SCSI controllers.

env_display *pathname* | **controller** *tray-number*
Display the environmental information for the specified unit.

alarm *pathname* | **controller** *tray-number*
Display the current state of the audible alarm.

alarm_on *pathname* | **controller** *tray-number*
alarm_off *pathname* | **controller** *tray-number*
Enable or disable the audible alarm for this enclosure.

alarm_set *pathname* | **controller** *tray-number* [*seconds*]
Set the audible alarm setting to seconds.

led *pathname*
Display the current state of the led for the specified disk.

led_on *pathname*
led_off *pathname*
Turn on or off the led for this disk.

power_off *pathname* | **controller** *tray-number*
Power down this RSM. The RSM will need to be powered back on manually. This command does not work with RSMs directly attached to wide differential SCSI controllers.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWssaop

SEE ALSO

disks(1M), **luxadm**(1M), **attributes**(5), **ses**(7D)

SPARCstorage Array User’s Guide

NAME

statd – network status monitor

SYNOPSIS

/usr/lib/nfs/statd

DESCRIPTION

statd is an intermediate version of the status monitor. It interacts with **lockd**(1M) to provide the crash and recovery functions for the locking services on NFS. **statd** keeps track of the clients with processes which hold locks on a server. When the server reboots after a crash, **statd** sends a message to the **statd** on each client indicating that the server has rebooted. The client **statd** processes then inform the **lockd** on the client that the server has rebooted. The client **lockd** then attempts to reclaim the lock(s) from the server.

statd on the client host also informs the **statd** on the server(s) holding locks for the client when the client has rebooted. In this case, the **statd** on the server informs its **lockd** that all locks held by the rebooting client should be released, allowing other processes to lock those files.

FILES

/var/statmon/sm lists hosts to be contacted after a reboot

/var/statmon/sm.bak lists hosts that could not be contacted after last reboot

/var/statmon/state includes a number which changes during a reboot

/usr/include/rpcsvc/sm_inter.x
contains the rpcgen source code for the interface services provided by the statd daemon.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

lockd(1M), **attributes**(5)

NOTES

The crash of a server is only detected upon its recovery.

NAME

strace – print STREAMS trace messages

SYNOPSIS

strace [*mid sid level*] ...

DESCRIPTION

strace without arguments writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages are obtained from the STREAMS log driver (see **log**(7D)). If arguments are provided, they must be in triplets of the form *mid, sid, level*, where *mid* is a STREAMS module ID number, *sid* is a sub-ID number, and *level* is a tracing priority level. Each triplet indicates that tracing messages are to be received from the given module / driver, sub-ID (usually indicating minor device), and priority level equal to, or less than the given level. The token **all** may be used for any member to indicate no restriction for that attribute.

The format of each trace message output is:

<seq> <time> <ticks> <level> <flags> <mid> <sid> <text>

<seq> trace sequence number

<time> time of message in *hh:mm:ss*

<ticks> time of message in machine ticks since boot

<level> tracing priority level

<flags> E : message is also in the error log
F : indicates a fatal error
N : mail was sent to the system administrator (hardcoded as root)

<mid> module ID number of source

<sid> sub-ID number of source

<text> formatted text of the trace message

Once initiated, **strace** will continue to execute until terminated by the user.

EXAMPLES

Output all trace messages from the module or driver whose module ID is 41:

strace 41 all all

Output those trace messages from driver / module ID 41 with sub-IDs 0, 1, or 2:

strace 41 0 1 41 1 1 41 2 0

Messages from sub-IDs 0 and 1 must have a tracing level less than or equal to 1. Those from sub-ID 2 must have a tracing level of 0.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5), **log**(7D)

STREAMS Programming Guide

NOTES

- There is no restriction to the number of **strace** processes opening the STREAMS log driver at a time.
- The log-driver records the list of the triplets specified in the command invocation, and compares each potential trace message against this list to decide if it should be formatted and sent up to the **strace** process. Hence, long lists of triplets will have a greater impact on overall STREAMS performance. Running **strace** will have the most impact on the timing of the modules and drivers generating the trace messages that are sent to the **strace** process. If trace messages are generated faster than the **strace** process can handle them, some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

NAME

strclean – STREAMS error logger cleanup program

SYNOPSIS

strclean [–**a** *age*] [–**d** *logdir*]

DESCRIPTION

strclean is used to clean up the STREAMS error logger directory on a regular basis (for example, by using **cron**. By default, all files with names matching **error.*** in **/var/adm/streams** that have not been modified in the last three days are removed.

OPTIONS

–**a** *age* The maximum age in days for a log file can be changed using the –**a** option.

–**d** *logdir* A directory other than **/var/adm/streams** can be specified using the –**d** option.

EXAMPLES

This example has the same result as running **strclean** with no arguments:

```
example% strclean –d /var/adm/streams –a 3
```

FILES

/var/adm/streams/error.*

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cron(1M), **strerr**(1M), **attributes**(5)

STREAMS Programming Guide

NOTES

strclean is typically run from **cron** on a daily or weekly basis.

NAME

strerr – STREAMS error logger daemon

SYNOPSIS

strerr

DESCRIPTION

strerr receives error log messages from the STREAMS log driver (see **log**(7D)) and appends them to a log file. The resultant error log files reside in the directory **/var/adm/streams**, and are named **error.***mm-dd*, where *mm* is the month and *dd* is the day of the messages contained in each log file.

The format of an error log message is:

<seq> <time> <ticks> <flags> <mid> <sid> <text>

<seq> error sequence number

<time> time of message in hh:mm:ss

<ticks> time of message in machine ticks since boot priority level

<flags> **T** : the message was also sent to a tracing process
F : indicates a fatal error
N : send mail to the system administrator (hardcoded as root)

<mid> module ID number of source

<sid> sub-ID number of source

<text> formatted text of the error message

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the system administrator. Those messages which indicate the total failure of a STREAMS driver or module should have the **F** flag set. Those messages requiring the immediate attention of the administrator will have the **N** flag set, which causes the error logger to send the message to the system administrator using **mail**. The priority level usually has no meaning in the error log but will have meaning if the message is also sent to a tracer process.

Once initiated, **strerr** continues to execute until terminated by the user. It is commonly executed asynchronously.

FILES

/var/adm/streams/error.*mm-dd* error log file.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5), **log**(7D)

STREAMS Programming Guide

NOTES

- There is no restriction to the number of **strerr** processes opening the STREAMS log driver at a time.
- If a module or driver is generating a large number of error messages, running the error logger will cause a degradation in STREAMS performance. If a large burst of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

NAME

sttydefs – maintain line settings and hunt sequences for TTY ports

SYNOPSIS

/usr/sbin/sttydefs –a *ttylabel* [**–b**] [**–f** *final-flags*] [**–i** *initial-flags*] [**–n** *nextlabel*]
/usr/sbin/sttydefs –l [*ttylabel*]
/usr/sbin/sttydefs –r *ttylabel*

DESCRIPTION

sttydefs is an administrative command that maintains the line settings and hunt sequences for the system's TTY ports by making entries in, and deleting entries from the **/etc/ttydefs** file.

sttydefs with a **–a** or **–r** option may be invoked only by the super-user. **sttydefs** with **–l** may be invoked by any user on the system.

OPTIONS

–a *ttylabel* Add a record to the **ttydefs** file, using *ttylabel* as its label. The following describes the effect of the **–b**, **–n**, **–i**, or **–f** options when used in conjunction with the **–a** option:

–b Enable autobaud. Autobaud allows the system to set the line speed of a given TTY port to the line speed of the device connected to the port without the user's intervention.

–f *final-flags* Specify the value to be used in the *final-flags* field in **/etc/ttydefs**. *final-flags* must be in a format recognized by the **stty** command. *final-flags* are the **termio**(7I) settings used by **ttymon** after receiving a successful connection request and immediately before invoking the service on the port. If this option is not specified, **sttydefs** will set *final-flags* equal to the **termio**(7I) flags **9600** and **sane**.

–i *initial-flags* Specify the value to be used in the *initial-flags* field in **/etc/ttydefs**. *initial-flags* must be in a format recognized by the **stty** command. These flags are used by **ttymon** when searching for the correct baud rate. They are set prior to writing the prompt.

If this option is not specified, **sttydefs** will set *initial-flags* equal to the **termio**(7I) flag **9600**.

–n *nextlabel* Specify the value to be used in the *nextlabel* field in **/etc/ttydefs**. If this option is not specified, **sttydefs** will set *nextlabel* equal to *ttylabel*.

–l If a *ttylabel* is specified, **sttydefs** displays the record from **/etc/ttydefs** whose TTY label matches the specified *ttylabel*. If no *ttylabel* is specified, **sttydefs** displays the entire contents of **/etc/ttydefs**. **sttydefs** verifies that each entry it displays is correct and that the entry's *nextlabel* field references an existing

–r *ttylabel* Remove any record in the **ttydefs** file that has *ttylabel* as its label.

OUTPUT

If successful, **sttydefs** will exit with a status of **0**. **sttydefs –l** will generate the requested information and send it to standard output.

EXAMPLES

The following command lists all the entries in the **ttydefs** file and prints an error message for each invalid entry that is detected.

```
example# sttydefs –l
```

The following shows a command that requests information for a single label and its output:

```
example# sttydefs –l 9600

----------------------------------------------------------------
9600:9600 hupcl erase ^h:9600 sane ixany tab3 hupcl erase ^h::4800
----------------------------------------------------------------

ttylabel:              9600
initial flags:         9600 hupcl erase ^h
final flags:           9600 sane ixany tab3 hupcl erase ^h
autobaud:              no
nextlabel:             4800
```

The following sequence of commands will add the labels **1200**, **2400**, **4800**, and **9600** and put them in a circular list:

```
sttydefs –a 1200 –n 2400 –i 1200 –f "1200 sane"
sttydefs –a 2400 –n 4800 –i 2400 –f "2400 sane"
sttydefs –a 4800 –n 9600 –i 4800 –f "4800 sane"
sttydefs –a 9600 –n 1200 –i 9600 –f "9600 sane"
```

FILES

/etc/ttydefs

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

attributes(5), **termio**(7I)

NAME

su – become super user or another user

SYNOPSIS

su [–] [*username* [*arg* . . .]]

DESCRIPTION

su allows one to become another user without logging off. The default user *name* is **root** (super user).

To use **su**, the appropriate password must be supplied (unless the invoker is already **root**). If the password is correct, **su** creates a new shell process that has the real and effective user ID, group IDs, and supplementary group list set to those of the specified *username*. The new shell will be the shell specified in the shell field of *username's* password file entry (see **passwd**(4)). If no shell is specified, **/usr/bin/sh** is used (see **sh**(1)). To return to normal user ID privileges, type an EOF character (CTRL-D) to exit the new shell.

Any additional arguments given on the command line are passed to the new shell. When using programs such as **sh**, an *arg* of the form –**c** *string* executes *string* using the shell and an *arg* of –**r** gives the user a restricted shell.

The following statements are true only if either **/usr/bin/sh** or **NULL** is named in the specified user's password file entry. If the first argument to **su** is a ' **-** ' (dash), the environment is passed along unchanged, as if the user actually logged in as the specified user. Otherwise, the environment is passed along, with the exception of **$PATH**, which is controlled by **PATH** and **SUPATH** in **etc/default/su**.

All attempts to become another user using **su** are logged in the log file **/var/adm/sulog** (see **sulog**(4)).

SECURITY

su uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **su**. Here is a partial **pam.conf** file with entries for the **su** command using the UNIX authentication, account management, and session management module.

```
su    auth       required    /usr/lib/security/pam_unix.so.1
su    account    required    /usr/lib/security/pam_unix.so.1
su    session    required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **su** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords.

EXAMPLES

To become user **bin** while retaining your previously exported environment, execute:

```
example% su bin
```

To become user **bin** but change the environment to what would be expected if **bin** had originally logged in, execute:

```
example% su – bin
```

To execute *command* with the temporary environment and permissions of user **bin**, type:

```
example% su – bin –c "command args"
```

ENVIRONMENT

If any of the **LC_*** variables (**LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC,** and **LC_MONETARY**) (see **environ**(5)) are not set in the environment, the operational behavior of **su** for each corresponding locale category is determined by the value of the **LANG** environment variable. If **LC_ALL** is set, its contents are used to override both the **LANG** and the other **LC_*** variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how **su** behaves.

LC_CTYPE Determines how **su** handles characters. When **LC_CTYPE** is set to a valid value, **su** can display and handle text and filenames containing valid characters for that locale. **su** can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. **su** can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

FILES

$HOME/.profile	user's login commands for sh and ksh
/etc/passwd	system's password file
/etc/profile	system-wide sh and ksh login commands
/var/adm/sulog	log file
/etc/default/su	the default parameters that reside here are:

SULOG: If defined, all attempts to **su** to another user are logged in the indicated file.

CONSOLE: If defined, all attempts to **su** to **root** are logged on the console.

PATH: Default path. (**/usr/bin:**)

SUPATH: Default path for a user invoking **su** to **root**. (**/usr/sbin:/usr/bin**)

SYSLOG: Determines whether the **syslog**(3) **LOG_AUTH** facility should be used to log all **su** attempts. **LOG_NOTICE** messages are generated for **su**'s to root, **LOG_INFO** messages are generated for **su**'s to other users, and **LOG_CRIT** messages are generated for failed **su** attempts.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

csh(1), **env**(1), **ksh**(1), **login**(1), **sh**(1), **syslogd**(1M), **pam**(3), **syslog**(3), **pam.conf**(4), **passwd**(4), **profile**(4), **sulog**(4), **attributes**(5), **environ**(5), **pam_unix**(5)

NAME

sulogin – access single-user mode

SYNOPSIS

sulogin

DESCRIPTION

sulogin is automatically invoked by **init** when the system is first started. It prompts the user to type the root password to enter system maintenance mode (single-user mode) or to type EOF (typically CTRL-D) for normal startup (multi-user mode). **sulogin** should never be directly invoked by the user.

FILES

/etc/sulogin

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

SEE ALSO

init(1M), **attributes**(5)

NAME

suninstall – install the Solaris environment

SYNOPSIS

suninstall

DESCRIPTION

suninstall is a forms-based subsystem for installing the operating system.

suninstall only exists on the Solaris CD-ROM and should only be invoked from there. Refer to the installation manual for more details.

suninstall allows installation of the operating system onto any stand-alone system. **suninstall** loads the software available on the CD-ROM. Refer to the installation manual for disk space requirements.

To abort the installation procedure, use the interrupt character (typically, CTRL-C).

USAGE

Refer to the installation manual for more information on the various menus and selections.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcdrom (Solaris CD)

SEE ALSO

pkginfo(1), **install**(1M), **pkgadd**(1M), **attributes**(5)

Solaris Advanced Installation Guide

NOTES

It is advisable to exit **suninstall** through the exit options from the **suninstall** menus.

NAME

swap – swap administrative interface

SYNOPSIS

/usr/sbin/swap –a *swapname* [*swaplow*] [*swaplen*]
/usr/sbin/swap –d *swapname* [*swaplow*]
/usr/sbin/swap –l
/usr/sbin/swap –s

DESCRIPTION

swap provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager.

OPTIONS

–**a** *swapname*

Add the specified swap area. This option can only be used by the super-user. *swapname* is the name of the swap file: for example, **/dev/dsk/c0t0d0s1** or a regular file. *swaplow* is the offset in 512-byte blocks into the file where the swap area should begin. *swaplen* is the desired length of the swap area in 512-byte blocks. The value of *swaplen* can not be less than **16**. For example, if *n* blocks are specified, then (*n*-1) blocks would be the actual swap length. *swaplen* must be at least one page in length. One page of memory is equivalent to eight 512-byte blocks. The size of a page of memory can be determined by using the **pagesize** command. See **pagesize**(1). Since the first page of a swap file is automatically skipped, and a swap file needs to be at least one page in length, the minimum size should be a factor of 2 pagesize bytes. The size of a page of memory is machine dependent.

swaplow + *swaplen* must be less than or equal to the size of the swap file. If *swaplen* is not specified, an area will be added starting at *swaplow* and extending to the end of the designated file. If neither *swaplow* nor *swaplen* are specified, the whole file will be used except for the first page. Swap areas are normally added automatically during system startup by the **/sbin/swapadd** script. This script adds all swap areas which have been specified in the **/etc/vfstab** file; for the syntax of these specifications, see **vfstab**(4).

To use an NFS or local file-system *swapname*, you should first create a file using **mkfile**(1M). A local file-system swap file can now be added to the running system by just running the **swap -a** command. For NFS mounted swap files, the server needs to export the file. Do this by performing the following steps:

1. Add the following line to **/etc/dfs/dfstab**:

   ```
   share -F nfs -o rw=clientname,root=clientname path-to-swap-file
   ```

2. Run **shareall**(1M).

3. Have the client add the following lines to **/etc/vfstab**:

   ```
   server:path-to-swap-file - local-path-to-swap-file nfs ---
   local-path-to-swap-file -- swap ---
   ```

4. Have the client run **mount**:

```
# mount local-path-to-swap-file
```

5. The client can then run **swap -a** to add the swap space:

```
# swap -a local-path-to-swap-file
```

–**d** *swapname*
Delete the specified swap area. This option can only be used by the super-user. *swapname* is the name of the swap file: for example, **/dev/dsk/c0t0d0s1** or a regular file. *swaplow* is the offset in 512-byte blocks into the swap area to be deleted. If *swaplow* is not specified, the area will be deleted starting at the second page. When the command completes, swap blocks can no longer be allocated from this area and all swap blocks previously in use in this swap area have been moved to other swap areas.

–**l**
List the status of all the swap areas. The output has five columns:

path The path name for the swap area.

dev The major / minor device number in decimal if it is a block special device; zeroes otherwise.

swaplo The *swaplow* value for the area in 512-byte blocks.

blocks The *swaplen* value for the area in 512-byte blocks.

free The number of 512-byte blocks in this area that are not currently allocated.

The list does not include swap space in the form of physical memory because this space is not associated with a particular swap area.

If **swap -l** is run while *swapname* is in the process of being deleted (by **swap -d**), the string INDEL will appear in a sixth column of the swap stats.

–**s**
Print summary information about total swap space usage and availability:

allocated The total amount of swap space (in 1024-byte blocks) currently allocated for use as backing store.

reserved The total amount of swap space (in 1024-bytes blocks) not currently allocated, but claimed by memory mappings for possible future use.

used The total amount of swap space (in 1024-byte blocks) that is either allocated or reserved.

available The total swap space (in 1024-byte blocks) that is currently available for future reservation and allocation.

These numbers include swap space from all configured swap areas as listed by the –**l** option, as well swap space in the form of physical memory.

USAGE

See **largefile**(5) for the description of the behavior of **swap** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

pagesize(1), **mkfile**(1M), **shareall**(1M), **getpagesize**(3C), **vfstab**(4), **attributes**(5), **largefile**(5)

WARNINGS

No check is done to see if a swap area being added overlaps with an existing file system.

NAME

swmtool – install, upgrade, and remove software packages

SYNOPSIS

swmtool [–**d** *directory*]

DESCRIPTION

The **swmtool** command invokes the **admintool**(1M) application, which is preselected to add or remove software. Using the **swmtool** command, you can add software from a product CD or hard disk to an installed system, or you can remove software from an installed system.

Once logged in, you may run **swmtool** to examine the packages on your local system.

Membership in the sysadmin group (gid 14) is used to restrict access to administrative tasks. Members of the sysadmin group can use **swmtool** to add or remove software packages. Non-members have read-only permissions (where applicable).

Help is available by using the Help button.

OPTIONS

–**d** *directory*
Specify the directory containing the software to be installed.

EXAMPLES

The following example starts the **admintool**( ) application and tells it to look for software packages in the local directory **/cdrom/cdrom0/s0** (the default directory for a CD when running Volume Manager).

```
example% /usr/sbin/swmtool –d /cdrom/cdrom0/s0
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap

SEE ALSO

admintool(1M), **pkgadd**(1M), **pkgrm**(1M), **attributes**(5)

Solaris Advanced User's Guide

NAME

sxconfig – configure contiguous memory for the SX video subsystem

SYNOPSIS

/usr/platform/*platform-name***/sbin/sxconfig –c**
/usr/platform/*platform-name***/sbin/sxconfig –d**
/usr/platform/*platform-name***/sbin/sxconfig** [**–f** | **–n**] [**–l** *limit*] [**–s** *size*]

DESCRIPTION

The **sxconfig** command configures contiguous memory parameters for exclusive use by the **SX** video system on the Desktop SPARCsystems with graphics option. *platform-name* can be found using the **–i** option of **uname**(1).

After configuring the physically contiguous memory, using the various options described below, the system must be rebooted for the changes to take effect. If this command is being used to configure physically contiguous memory for the first time after the system software has been installed, then the system must be rebooted using the reconfiguration option (**–r**) of **boot**(1M).

The amount of memory to be reserved depends upon the type of application. Applications that benefit from the availability of contiguous memory are those that are written to the XGL and XIL graphics and imaging foundation library APIs.

The *Platform Notes: SPARCstation 10SX System Configuration Guide* provides more detailed information regarding how much memory to reserve for various types of graphics and imaging applications.

sxconfig is supported only on Desktop SPARCsystems with SX graphics option.

The interface, output, and command location are uncommitted and subject to change in future releases.

OPTIONS

The following options are supported:

–c Display the current configuration parameters in the driver configuration file. If the system was not rebooted after previously changing the configuration parameters, then the displayed values do not reflect the actual system setup.

–d Restore all configuration parameters to the default values. By default, 0 megabytes of physically contiguous memory are reserved, fragmentation is not allowed, and 32 megabytes of memory are reserved for system use.

–f Allow fragmentation. If no single chunk of memory of at least the requested size is found, allow the request to span multiple chunks. This flag also specifies that less than *size* megabytes of data may be reserved if there are not enough contiguous chunks available. If this flag is not specified, then the memory reserved must be exactly one chunk of the requested size for the request to succeed.

–n Fragmentation not allowed.

–l *limit* Specify that at least *limit* megabytes of total memory must remain for system use after the contiguous memory has been reserved.

–s *size* Reserve *size* megabytes of contiguous memory for exclusive use by the **SX** video subsystem.

EXAMPLES

The following example reserves 16 megabytes of contiguous memory without fragmentation and indicates 32 megabytes of memory should remain for system use after reserving the contiguous memory:

```
example# sxconfig –s 16 –l 32
```

The following example is identical to the one described above except that fragmentation is allowed:

```
example# sxconfig –s 16 –f –l 32
```

The following example reports current configuration parameters in the driver configuration file:

```
example# sxconfig –c
```

This example restores all configuration parameters to the default values:

```
example# sxconfig –d
```

The following example disables fragmentation:

```
example# sxconfig –n
```

EXIT STATUS

sxconfig returns **0** on success, and a positive integer on failure.

1 Permission denied. Only root can run this command.

2 Configuration file **sx_cmem.conf** does not exist.

3 Illegal option.

4 Illegal combination of options.

5 Illegal argument for **–s** option. Should be an integer.

6 Illegal argument for **–l** option. Should be an integer.

FILES

/platform/*platform-name***/kernel/drv/sx_cmem**
contiguous memory device driver

/platform/*platform-name***/kernel/drv/sx_cmem.conf**
configuration file for contiguous memory driver

/etc/init.d/sxcmem contiguous memory startup script

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm

SEE ALSO

uname(1), **boot**(1M), **init**(1M), **attributes**(5)

Platform Notes: SPARCstation 10SX System Configuration Guide

NAME

sync – update the super block

SYNOPSIS

sync

DESCRIPTION

sync executes the **sync** system primitive. If the system is to be stopped, **sync** must be called to insure file system integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See **sync**(2) for details.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

sync(2), **attributes**(5)

NOTES

If you have done a write to a file on a remote machine in a Remote File Sharing environment, you cannot use **sync** to force buffers to be written out to disk on the remote machine. **sync** will only write local buffers to local disks.

NAME

syncinit – set serial line interface operating parameters

SYNOPSIS

/usr/sbin/syncinit *device* [[*baud_rate*] | [*keyword*=*value*,. . .] | [*single-word option*]]

DESCRIPTION

The **syncinit** utility allows the user to modify some of the hardware operating modes common to synchronous serial lines. This can be useful in troubleshooting a link, or necessary to the operation of a communications package.

If run without options, **syncinit** reports the options as presently set on the port. If options are specified, the new settings are reported after they have been made.

OPTIONS

Options to **syncinit** normally take the form of a keyword, followed by an equal sign and a value. The exception is that a baud rate may be specified as a decimal integer by itself. Keywords must begin with the value shown in the options table, but may contain additional letters up to the equal sign. For example, **loop=** and **loopback=** are equivalent.

Recognized options are listed in the table below.

Keyword	**Value**	**Effect**
loop	yes	Set the port to operate in **internal loopback** mode. The receiver is electrically disconnected from the DCE receive data input and tied to the outgoing transmit data line. Transmit data is available to the DCE. The Digital Phase-Locked Loop (DPLL) may not be used as a clock source in this mode. If no other clocking options have been specified, perform the equivalent of **txc=baud** and **rxc=baud**.
	no	Disable internal loopback mode. If no other clocking options have been specified, perform the equivalent of **txc=txc** and **rxc=rxc**.
echo	yes	Set the port to operate in **auto-echo** mode. The transmit data output is electrically disconnected from the transmitter and tied to the receive data input. Incoming receive data is still visible. Use of this mode in combination with local loopback mode has no value, and should be rejected by the device driver. The **auto-echo** mode is useful to make a system become the endpoint of a remote loopback test.
	no	Disable **auto-echo** mode.
nrzi	yes	Set the port to operate with **NRZI** data encoding.
	no	Set the port to operate with **NRZ** data encoding.
txc	**txc**	Transmit clock source will be the **TxC** signal (pin 15).
	rxc	Transmit clock source will be the **RxC** signal (pin 17).
	baud	Transmit clock source will be the internal **baud rate generator**.
	pll	Transmit clock source will be the output of the **DPLL** circuit.

rxc	**rxc**	Receive clock source will be the **RxC** signal (pin 17).
	txc	Receive clock source will be the **TxC** signal (pin 15).
	baud	Receive clock source will be the internal **baud rate generator**.
	pll	Receive clock source will be the output of the **DPLL** circuit.
speed	*integer*	Set the baud rate to *integer* bits per second.

There are also several single-word options that set one or more paramaters at a time:

Keyword	**Equivalent to Options:**
external	**txc=txc rxc=rxc loop=no**
sender	**txc=baud rxc=rxc loop=no**
internal	**txc=pll rxc=pll loop=no**
stop	**speed=0**

EXAMPLES

The following command sets the first CPU port to loop internally, use internal clocking and operate at 38400 baud:

```
example# syncinit zsh0 38400 loop=yes
device: /dev/zsh  ppa: 0
speed=38400, loopback=yes, echo=no, nrzi=no, txc=baud, rxc=baud
```

The following command sets the same port's clocking, local loopback and baud rate settings to their default values:

```
example# syncinit zsh0 stop loop=no
device: /dev/zsh  ppa: 0
speed=0, loopback=no, echo=no, nrzi=no, txc=txc, rxc=rxc
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

syncloop(1M), **syncstat**(1M), **intro**(2), **ioctl**(2), **attributes**(5), **zsh**(7D)

DIAGNOSTICS

device **missing minor device number**

The name *device* does not end in a decimal number that can be used as a minor device number.

bad speed: *arg*

The string *arg* that accompanied the **speed=** option could not be interpreted as a decimal integer.

Bad arg: *arg*

The string *arg* did not make sense as an option.

ioctl failure code = *errno*

An **ioctl**(2) system called failed. The meaning of the value of *errno* may be found in **intro**(2).

WARNINGS

syncinit should not be used on an active serial link, unless needed to resolve an error condition. It should not be run casually, or if the user is unsure of the consequences of its use.

NAME

syncloop – synchronous serial loopback test program

SYNOPSIS

/usr/sbin/syncloop [**–cdlstv**] *device*

DESCRIPTION

The **syncloop** command performs several loopback tests that are useful in exercising the various components of a serial communications link.

Before running a test, **syncloop** opens the designated port and configures it according to command line options and the specified test type. It announces the names of the devices being used to control the hardware channel, the channel number (ppa) corresponding to the *device* argument, and the parameters it has set for that channel. It then runs the loopback test in three phases.

The first phase is to listen on the port for any activity. If no activity is seen for at least four seconds, **syncloop** proceeds to the next phase. Otherwise, the user is informed that the line is active and that the test cannot proceed, and the program exits.

In the second phase, called the "first-packet" phase, **syncloop** attempts to send and receive one packet. The program will wait for up to four seconds for the returned packet. If no packets are seen after five attempts, the test fails with an excoriating message. If a packet is returned, the result is compared with the original. If the length and content do not match exactly, the test fails.

The final phase, known as the "multiple-packet" phase, attempts to send many packets through the loop. Because the program has verified the integrity of the link in the first-packet phase, the test will not fail after a particular number of timeouts. If a packet is not seen after four seconds, a message is displayed. Otherwise, a count of the number of packets received is updated on the display once per second. If it becomes obvious that the test is not receiving packets during this phase, the user may wish to stop the program manually. The number and size of the packets sent during this phase is determined by default values, or by command line options. Each returned packet is compared with its original for length and content. If a mismatch is detected, the test fails. The test completes when the required number of packets have been sent, regardless of errors.

After the multiple-packet phase has completed, the program displays a summary of the hardware event statistics for the channel that was tested. The display takes the following form:

```
CRC errors   Aborts   Overruns   Underruns        In <-Drops-> Out
         0        0          0           0         0           0
```

This is followed by an estimated line speed, which is an approximation of the bit rate of the line, based on the number of bytes sent and the actual time that it took to send them.

OPTIONS

The options for **syncloop** are described in the following table:

Option	**Parameter**	**Default**	**Description**
–c	*packet_count*	100	Specifies the number of packets to be sent in the multiple-packet phase.
–d	*hex_data_byte*	*random*	Specifies that each packet will be filled with bytes with the value of *hex_data_byte*.
–l	*packet_length*	100	Specifies the length of each packet in bytes.
–s	*line_speed*	9600	Bit rate in bits per second.
–v			Sets verbose mode. If data errors occur, the expected and received data is displayed.
–t	*test_type*	*none*	A number, from 1 to 4, that specifies which test to perform. The values for *test_type* are as follows: **1** Internal loopback test. Port loopback is on. Transmit and receive clock sources are internal (baud rate generator). **2** External loopback test. Port loopback is off. Transmit and receive clock sources are internal. Requires a loopback plug suitable to the port under test. **3** External loopback test. Port loopback is off. Transmit and receive clock sources are external (modem). Requires that one of the local modem, the remote modem, or the remote system be set in a loopback configuration. **4** Test using predefined parameters. User defines hardware configuration and may select port parameters using the **syncinit**(1M) command.

All numeric options except **–d** are entered as decimal numbers (for example, **–s 19200**). If you do not provide the **–t** *test_type* option, **syncloop** prompts for it.

EXAMPLES

In the following command **syncloop** uses a packet length of 512 bytes over the first CPU port:

```
example# syncloop –l 512 zsh0
```

In response to the above command, **syncloop** prompts you for the test option you want.

The following command performs an internal loopback test on the first CPU port, using 5000 packets and a bit rate of 56Kbps:

```
example# syncloop –t 1 –s 56000 –c 5000 zsh0
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

syncinit(1M), **syncstat**(1M), **attributes**(5), **zsh**(7D)

DIAGNOSTICS

device **missing minor device number**
The name *device* does not end in a decimal number that can be used as a minor device number.

invalid packet length: *nnn*
The packet length was specified to be less than zero or greater than 4096.

poll: nothing to read

poll: nothing to read or write.
The **poll**(2) system call indicates that there is no input pending and / or that output would be blocked if attempted.

len *xxx* **should be** *yyy*
The packet that was sent had a length of *yyy*, but was received with a length of *xxx*.

nnn **packets lost in outbound queueing**

nnn **packets lost in inbound queueing**
A discrepancy has been found between the number of packets sent by **syncloop** and the number of packets the driver counted as transmitted, or between the number counted as received and the number read by the program.

WARNINGS

To allow its tests to run properly, as well as prevent disturbance of normal operations, **syncloop** should only be run on a port that is not being used for any other purpose at that time.

NAME

syncstat – report driver statistics from a synchronous serial link

SYNOPSIS

/usr/sbin/syncstat [**–c**] *device* [*interval*]

DESCRIPTION

The **syncstat** command reports the event statistics maintained by a synchronous serial device driver. The report may be a single snapshot of the accumulated totals, or a series of samples showing incremental changes. Prior to these it prints the device name being used to query a particular device driver, along with a number indicating the channel number (ppa) under control of that driver.

Event statistics are maintained by a driver for each physical channel that it supports. They are initialized to zero at the time the driver module is loaded into the system, which may be either at boot time or when one of the driver's entry points is first called.

The *device* argument is the name of the serial device as it appears in the **/dev** directory. For example, **zsh0** specifies the first on-board serial device.

The following is a breakdown of **syncstat** output:

speed The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.

ipkts The total number of input packets.

opkts The total number of output packets.

undrun The number of transmitter underrun errors.

ovrrun The number of receiver overrun errors.

abort The number of aborted received frames.

crc The number of received frames with CRC errors.

isize The average size (in bytes) of input packets.

osize The average size (in bytes) of output packets.

OPTIONS

–c Clear the accumulated statistics for the device specified. This may be useful when it is not desirable to unload a particular driver, or when the driver is not capable of being unloaded.

interval **syncstat** samples the statistics every *interval* seconds and reports incremental changes. The output reports line utilization for input and output in place of average packet sizes. These are the relationships between bytes transferred and the baud rate, expressed as percentages. The loop repeats indefinitely, with a column heading printed every twenty lines for convenience.

EXAMPLES

```
example# syncstat zsh0
speed   ipkts   opkts   undrun   ovrrun   abort   crc   isize   osize
 9600   15716   17121      0        0       1      3     98      89
```

```
example# syncstat –c zsh0
speed    ipkts    opkts    undrun    ovrrun    abort    crc    isize    osize
 9600      0        0        0         0         0       0       0        0
example# syncstat zsh0 5
ipkts    opkts    undrun    ovrrun    abort    crc    iutil    outil
  12       10       0         0         0       0      5%       4%
  22       60       0         0         0       0      3%      90%
  36       14       0         0         0       1     51%       2%
```

In this final example a new line of output is generated every five seconds.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

syncinit(1M), **syncloop**(1M), **attributes**(5), **zsh**(7D)

DIAGNOSTICS

bad interval: *arg*

The argument *arg* is expected to be an interval and could not be understood.

device **missing minor device number**

The name *device* does not end in a decimal number that can be used as a minor device number.

baud rate not set

The *interval* option is being used and the baud rate on the device is zero. This would cause a divide-by-zero error when computing the line utilization statistics.

WARNINGS

Underrun, overrun, frame-abort, and CRC errors have a variety of causes. Communication protocols are typically able to handle such errors and initiate recovery of the transmission in which the error occurred. Small numbers of such errors are not a significant problem for most protocols. However, because the overhead involved in recovering from a link error can be much greater than that of normal operation, high error rates can greatly degrade overall link throughput. High error rates are often caused by problems in the link hardware, such as cables, connectors, interface electronics or telephone lines. They may also be related to excessive load on the link or the supporting system.

The percentages for input and output line utilization reported when using the *interval* option may occasionally be reported as slightly greater than 100% because of inexact sampling times and differences in the accuracy between the system clock and the modem clock. If the percentage of use greatly exceeds 100%, or never exceeds 50%, then the baud rate set for the device probably does not reflect the speed of the modem.

NAME

sysdef – output system definition

SYNOPSIS

/usr/sbin/sysdef [–**n** *namelist*]
/usr/sbin/sysdef [–**h**] [–**d**] [–**D**]

DESCRIPTION

sysdef outputs the current system definition in tabular form. It lists all hardware devices, as well as pseudo devices, system devices, loadable modules, and the values of selected kernel tunable parameters.

It generates the output by analyzing the named bootable operating system file (*namelist*) and extracting the configuration information from it.

The default system *namelist* is **/dev/kmem**.

OPTIONS

–**n** *namelist* Specifies a *namelist* other than the default (**/dev/kmem**). The *namelist* specified must be a valid bootable operating system.

–**h** Prints the identifier of the current host in hexadecimal. This numeric value is unique across all Sun hosts.

–**d** The output includes the configuration of system peripherals formatted as a device tree.

–**D** For each system peripheral in the device tree, display the name of the device driver used to manage the peripheral.

EXAMPLES

The following example displays the format of the **sysdef** –**d** output:

```
example% sysdef –d
Node ’Sun 4/60’, unit #0 (no driver)
        Node ’options’, unit #0 (no driver)
        Node ’zs’, unit #0
        Node ’zs’, unit #1
        Node ’fd’, unit #0
        Node ’audio’, unit #0
        Node ’sbus’, unit #0
                Node ’dma’, unit #0
                Node ’esp’, unit #0
                        Node ’st’, unit #1 (no driver)
                        Node ’st’, unit #0
                        Node ’sd’, unit #2
                        Node ’sd’, unit #1
                        Node ’sd’, unit #0
                Node ’le’, unit #0
                Node ’bwtwo’, unit #0
        Node ’auxiliary-io’, unit #0
        Node ’interrupt-enable’, unit #0
        Node ’memory-error’, unit #0
        Node ’counter-timer’, unit #0
```

Node 'eeprom', unit #0

FILES

/dev/kmem default operating system image

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

hostid(1), **prtconf**(1M), **nlist**(3E), **attributes**(5)

NAME

sysidconfig – execute system configuration applications, or define set of system configuration applications

SYNOPSIS

sysidconfig [–**lv**] [–**a** *application*] [–**b** *basedir*] [–**r** *application*]

DESCRIPTION

Invoked without any options, the **sysidconfig** program executes a list of applications. An application on this list is referred to as a "system configuration application." Every application on this list will be passed one command-line argument, –**c**. This flag will cause the system configuration application to perform its configuration function. Without options, **sysidconfig** should only be invoked by startup scripts, which occurs during the initial installation and during a reconfigure reboot.

All applications on the list will be executed, if possible. All activity taken by the **sysidconfig** program is logged in the **sysidconfig** log file, **/var/log/sysidconfig.log**. If one or more of the applications on the list are either not present at execution time, are not executable, or execute but return a failure code upon completion, then that information will be logged as well. Successful completion of the program can be assumed if no error message is present in the log file. Programs are executed sequentially, with only one configuration application active at a time.

Executed with the –**l**, –**a**, or –**r** options, the **sysidconfig** program allows the super-user to list the defined configuration applications, and to add items to or remove items from that list. Running **sysidconfig** with options is the only way to view or manipulate the list. Only the super-user can execute the **sysidconfig** program with options.

The –**b** and **-v** options change the behavior of **sysidconfig**, and can be used with or without the list manipulation options discussed above. The –**b** *basedir* option is used to specify a reference **root** directory other than the default, **/**. The –**v** option duplicates the log file output on **stdout**.

By default, no SPARC based applications exist on this list. However, the x86 based systems are delivered with one application, **kdmconfig**(1M), on the list. **kdmconfig** is not delivered on SPARC based systems.

This application is an extension of the **sysidtool**(1M) suite of programs. It is executed during initial installation and during a reconfigure reboot, before the window system has been started. Graphical User Interface (GUI) applications will not execute successfully if they are added to the list of configuration applications via **sysidconfig** –**a**.

This program is referenced, but not fully described, in the **sysidtool**(1M) manual page.

OPTIONS

The valid options are:

–**a** *application* Add the named application to the list of defined applications. When next invoked without arguments, **sysidconfig** will run this newly added application after all previously defined applications. *application* must be a fully qualified path name that is not currently on the list of applications to execute.

–**b** *basedir* Specify an alternate base directory (**/** is defined as the default base directory if no other is specified). The specified directory is used as the **root** directory when adding, listing, removing, or executing configuration applications. The log file where information is recorded is in **/var/log**, relative to the specified *basedir*. In the log file, the *basedir* is not noted. This means, for example, that if the super-user on a diskless client's server executes:

fig –b /export/root/*client* –a /sbin/*someapp*"

then the diskless client *client* would have **/sbin/***someapp* executed upon reconfigure reboot. The diskless client's log file would note that **/sbin/***someapp* was added, not **/export/root/client/sbin/***someapp*.

–l List defined configuration applications. Applications will be executed one at a time, in the order shown in the list.

–r *application* Remove the named application from the list of defined applications. *application* must be a fully qualified path name and it must be on the existing list of applications to execute.

–v Verbose mode. This option echoes all information sent to the log file to **stdout**. Such information includes timestamp information about when the program was executed, the names of applications being executed, and results of those executions.

RETURN VALUES

The **sysidconfig** program will return **0** if it completes successfully.

When executed with the –**r** or –**a** options, error conditions or warnings will be reported on **stderr**. If the requested action completes successfully, an exit code of **0** will be returned.

ERRORS

EPERM The program was executed by a user other than the super-user.

EINVAL Option –**l**, –**a**, or –**r** was passed and the action could not be completed successfully.

FILES

/var/log/sysidconfig.log **sysidconfig** log file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap

SEE ALSO

sys-unconfig(1M), **sysidtool**(1M), **attributes**(5)

x86 Only

kdmconfig(1M)

DIAGNOSTICS

When run without options, a log of the **sysidconfig** program's activity can be found in **/var/log/sysidconfig.log**. This file contains a timestamp log of each program executed, its resulting **stderr** output, and its exit code. If an application in the list was not found or is not executable, that will also be noted.

NAME

sysidtool, sysidnet, sysidnis, sysidsys, sysidroot, sysidpm – system configuration

SYNOPSIS

/usr/sbin/sysidnet
/usr/sbin/sysidnis
/usr/sbin/sysidsys
/usr/sbin/sysidroot
/usr/sbin/sysidpm

DESCRIPTION

sysidtool is a suite of five programs that configure a new system, or one that has been unconfigured with **sys-unconfig**(1M). The **sysidtool** programs run automatically at system installation, or during the first boot after a machine has been successfully unconfigured.

These programs have no effect except at such times, and should never be run manually.

The **sysidtool** programs set up the appropriate information in the machine's configuration files, in the kernel, and on the machine's network interface. They may prompt for the following information:

sysidnet: network configuration

- Machine's default locale
- Machine's console type
- Machine's host name
- Machine's IP address

sysidnis: NIS / NIS+ client configuration

- Name service choice: NIS+, NIS, or none
- Machine's IP subnet mask (if no NIS / NIS+ server can automatically be located on the machine's sub-network)
- NIS / NIS+ domain name
- Hostname and IP address of an NIS / NIS+ server in the domain

sysidsys: miscellaneous system configuration

- Machine's IP subnet mask (if an NIS / NIS+ server was automatically located on the machine's sub-network)
- Machine's time zone
- Date and time

sysidroot: control superuser information

- Machine's root password

sysidpm: Power Management configuration

- Auto-shutdown confirmation if the system is Energystar-V2 compliant, that is, a new system model shipped after October 1, 1995.

sysidconfig: host or platform-specific configuration

- This command controls specification and execution of custom configuration applications which may be specified for a particular host or a particular platform. (See **sysidconfig**(1M)).

The **sysidtool** programs attempt to obtain system configuration information from various name service databases (for example, NIS) or from the **sysidcfg**(4) file, and you are prompted to provide the information if it cannot be found. However, you can avoid one or more of the prompts by preconfiguring the appropriate configuration information in the name service databases or in the **sysidcfg**(4) file.

To preconfigure the information in the name service databases, you must use the name service commands or the Solstice AdminSuite tools. See *Solaris Advanced Installation Guide* for more details about how to preconfigure the system configuration information.

The machine's configuration information is set up in its **/etc** and **/var** files.

FILES

/etc/.UNCONFIGURED	
/etc/nodename	
/etc/hostname.??[0-9]	
/etc/default/init	
/etc/defaultdomain	
/etc/passwd	password file. See **passwd**(4).
/etc/shadow	shadow password file. See **shadow**(4).
/etc/inet/hosts	
/etc/inet/netmasks	
/etc/net/*/hosts	
/var/nis/NIS_COLD_START	
/var/yp/aliases	
/var/yp/binding/*/ypservers	
/etc/.sysIDtool.state	
/etc/power.conf	Power management configuration file. See **power.conf**(4).
/etc/.PM_RECONFIGURE	If this file is present during system reboot, the **sysidpm** program will be run. This file will be removed by **sysidpm**.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap SUNWpmu

SEE ALSO

powerd(1M), **sys-unconfig**(1M), **sysidconfig**(1M), **passwd**(4), **power.conf**(4), **shadow**(4), **sysidcfg**(4), **attributes**(5)

Solaris Advanced Installation Guide

NOTES

If a system has more than one network interface, you can use **sysidtool** to configure only the primary interface on the system. All other interfaces on the system must be configured manually.

You cannot use the name service databases or the **sysidcfg**(4) file to suppress the Power Management configuration prompt; however, you can suppress it by creating either the **/autoshutdown** or **/noautoshutdown** file before installation reboot. Accordingly, the auto-shutdown feature is silently configured. The **/autoshutdown** or **/noautoshutdown** files are removed by **sysidpm** before it exits.

NAME

syslogd – log system messages

SYNOPSIS

/usr/sbin/syslogd [**–d**] [**–f** *configfile*] [**–m** *markinterval*] [**–p** *path*]

DESCRIPTION

syslogd reads and forwards system messages to the appropriate log files and / or users, depending upon the priority of a message and the system facility from which it originates. The configuration file **/etc/syslog.conf** (see **syslog.conf**(4)) controls where messages are forwarded. **syslogd** logs a mark (timestamp) message every *markinterval* minutes (default **20**) at priority **LOG_INFO** to the facility whose name is given as **mark** in the **syslog.conf** file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (**< >**); priorities are defined in **<sys/syslog.h>**.

syslogd reads from the STREAMS log driver, **/dev/log**, and from any transport provider specified in **/etc/netconfig**, **/etc/net/transport/hosts**, and **/etc/net/transport/services**.

syslogd reads the configuration file when it starts up, and again whenever it receives a **HUP** signal (see **signal**(5), at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. **syslogd** exits when it receives a **TERM** signal.

As it starts up, **syslogd** creates the file **/etc/syslog.pid**, if possible, containing its process identifier (PID).

OPTIONS

The following options are supported:

–d Turn on debugging. This option should only be used interactively in a root shell once the system is in multi-user mode. It should *not* be used in the system start-up scripts, as this will cause the system to hang at the point where **syslogd** is started.

–f *configfile* Specify an alternate configuration file.

–m *markinterval* Specify an interval, in minutes, between mark messages.

–p *path* Specify an alternative log device name. The default is **/dev/log**.

FILES

/etc/syslog.conf	configuration file
/etc/syslog.pid	process ID
/dev/log	STREAMS log driver
/etc/netconfig	specifies the transport providers available on the system
/etc/net/transport/hosts	network hosts for each transport
/etc/net/transport/services	network services for each transport

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO **logger**(1), **syslog**(3), **syslog.conf**(4), **attributes**(5), **signal**(5), **log**(7D)

NAME

sys-unconfig – undo a system’s configuration

SYNOPSIS

/usr/sbin/sys-unconfig

DESCRIPTION

sys-unconfig packs up a machine to make it ready to be configured again.

It restores a system’s configuration to an “as-manufactured” state. A system’s configuration consists of hostname, Network Information Service (NIS) domain name, timezone, IP address, IP subnet mask, and root password. This operation is the inverse of those performed by the **sysidnet**(1M), **sysidnis**(1M), and **sysidsys**(1M) programs run at boot. See **sysidtool**(1M).

sys-unconfig does the following:

- Saves current **/etc/inet/hosts** file information in **/etc/inet/hosts.saved**.
- Restores the default **/etc/inet/hosts** file.
- Removes the default hostname in **/etc/hostname.??[0-9]** and **/etc/nodename**.
- Removes the default domainname in **/etc/defaultdomain**.
- Restores the timezone to PST8PDT in **/etc/TIMEZONE**.
- Disables the Network Information Service (NIS) and Network Information Service Plus (NIS+) if either NIS or NIS+ was configured.
- Removes the entries for this host in **/etc/net/*/hosts**.
- Removes the file **/etc/inet/netmasks**.
- Removes the password set for root in **/etc/shadow**.
- Removes the file **/etc/.rootkey**.
- Execute all system configuration applications. These applications are defined by prior executions of a **sysidconfig** **−a** *application*. (See **sysidconfig**(1M)). When **sys-unconfig** is run, all system configuration applications are passed one argument, **−u**.

When **sys-unconfig** is finished, it performs a system shutdown.

sys-unconfig is a potentially dangerous utility and can only be run by the super-user.

FILES

/etc/default/init	process control initialization
/etc/defaultdomain	
/etc/hostname.??[0-9]	
/etc/inet/hosts	host name database
/etc/inet/netmasks	network mask database
/etc/net/*/hosts	
/etc/nodename	
/etc/.rootkey	super-user’s secret key
/etc/shadow	shadow password file
/var/nis/NIS_COLD_START	
/var/yp/binding/*/ypservers	

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap

SEE ALSO

init(1M), **kdmconfig**(1M), **sysidconfig**(1M), **sysidtool**(1M), **hosts**(4), **netmasks**(4), **shadow**(4), **attributes**(5)

NOTES

sys-unconfig is not available on diskless clients.

NAME

tapes – creates /dev entries for tape drives attached to the system

SYNOPSIS

/usr/sbin/tapes [**–r** *rootdir*]

DESCRIPTION

tapes creates symbolic links in the **/dev/rmt** directory to the actual tape device special files under the **/devices** directory tree. **tapes** searches the kernel device tree to see what tape devices are attached to the system. For each equipped tape drive, the following steps are performed:

1. The **/dev/rmt** directory is searched for a **/dev/rmt/***n* entry that is a symbolic link to the **/devices** special node of the current tape drive. If one is found, this determines the logical controller number of the tape drive.
2. The rest of the special devices associated with the drive are checked, and incorrect symbolic links are removed and necessary ones added.
3. If none are found, a new logical controller number is assigned (the lowest-unused number), and new symbolic links are created for all the special devices associated with the drive.

tapes does not remove links to non-existent devices; these must be removed by hand.

tapes is run each time a reconfiguration-boot is performed, or when **add_drv**(1M) is executed. When invoking **tapes**(1M) manually, first run **drvconfig**(1M)
to ensure **/devices** is consistent with the current device configuration.

Notice to Driver Writers

tapes(1M) considers all devices with the node type **DDI_NT_TAPE** to be tape devices; these devices must have their minor name created with a specific format. The minor name encodes operational modes for the tape device and consists of an ASCII string of the form [**l**,**m**,**h**,**c**,**u**][**b**][**n**].

The first character set is used to specify the tape density of the device, and are named low (**l**), medium (**m**), high (**h**), compressed (**c**), and ultra (**u**). These specifiers only express a relative density; it is up to the driver to assign specific meanings as needed. For example, 9 track tape devices interpret these as actual bits-per-inch densities, where **l** means 800 BPI, **m** means 1600 BPI , and **h** means 6250 BPI, whereas 4mm DAT tapes defines **l** as standard format, and **m**, **h**, **c** and **u** as compressed format. Drivers may choose to implement any or all of these format types.

During normal tape operation (non-BSD behavior), once an EOF mark has been reached, subsequent reads from the tape device return an error. An explicit IOCTL must be issued to space over the EOF mark before the next file can be read. **b** instructs the device to observe BSD behavior, where reading at EOF will cause the tape device to automatically space over the EOF mark and begin reading from the next file.

n or no-rewind-on-close instructs the driver to not rewind to the beginning of tape when the device is closed. Normal behavior for tape devices is to reposition to BOT when closing. See **mtio**(7I).

The minor number for tape devices should be created by encoding the device's instance number using the tape macro **MTMINOR** and ORing in the proper combination of density, BSD behavior, and no-rewind flags. See **mtio**(7I).

To prevent **disks** from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using a node type: **DDI_NT_BLOCK**, **DDI_NT_BLOCK_CHAN**, **DDI_NT_CD**, or **DDI_NT_CD_CHAN** when calling **ddi_create_minor_node**(9F).

OPTIONS

−r *rootdir* Causes **disks** to presume that the **/dev/dsk**, **/dev/rdsk** and **/devices** directory trees are found under *rootdir*, not directly under **/**.

ERRORS

If **tapes** finds entries of a particular logical controller linked to different physical controllers, it prints an error message and exits without making any changes to the **/dev** directory, since it cannot determine which of the two alternative logical to physical mappings is correct. The links should be manually corrected or removed before another reconfiguration boot is performed.

EXAMPLES

This example demonstrates creating tape device nodes from within the **xktape** driver's **attach**(9E) function.

```
#include    <sys/mtio.h>

struct tape_minor_info {
   char *minor_name;
   int   minor_mode;
};
/*
 * create all combinations of logical tapes
 */
static struct tape_minor_info example_tape[] = {
   {"",    0},                  /* default tape */
   {"l",   MT_DENSITY1},
   {"lb",  MT_DENSITY1 | MT_BSD},
   {"lbn", MT_DENSITY1 | MT_BSD | MT_NOREWIND},
   {"m",   MT_DENSITY2},
   {"mb",  MT_DENSITY2 | MT_BSD},
   {"mbn", MT_DENSITY2 | MT_BSD | MT_NOREWIND},
   {"h",   MT_DENSITY3},
   {"hb",  MT_DENSITY3 | MT_BSD},
   {"hbn", MT_DENSITY3 | MT_BSD | MT_NOREWIND},
   {"c",   MT_DENSITY4},
   {"cb",  MT_DENSITY4 | MT_BSD},
   {"cbn", MT_DENSITY4| MT_BSD | MT_NOREWIND},
   {NULL,  0},
};
```

```
    int
    xktapeattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
    {
        int instance;
        struct tape_minor_info *mdp;
                /* other stuff in attach... */

        instance = ddi_get_instance(dip);

        for (mdp = example_tape; mdp->minor_name != NULL; mdp++) {
              ddi_create_minor_node(dip, mdp->minor_name, S_IFCHR,
                  (MTMINOR(instance) | mdp->minor_mode), DDI_NT_TAPE, 0);
        }
    }
```

Installing the **xktape** driver on a SPARCstation 20, with the driver controlling a SCSI tape (target 4 attached to an **esp**(7D) SCSI HBA) and performing a reconfiguration-boot creates the following special files in **/devices**.

```
# ls -l /devices/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/

crw-rw-rw-   1 root sys   33,136 Aug 29 00:02  xktape@4,0:
crw-rw-rw-   1 root sys   33,200 Aug 29 00:02  xktape@4,0:b
crw-rw-rw-   1 root sys   33,204 Aug 29 00:02  xktape@4,0:bn
crw-rw-rw-   1 root sys   33,152 Aug 29 00:02  xktape@4,0:c
crw-rw-rw-   1 root sys   33,216 Aug 29 00:02  xktape@4,0:cb
crw-rw-rw-   1 root sys   33,220 Aug 29 00:02  xktape@4,0:cbn
crw-rw-rw-   1 root sys   33,156 Aug 29 00:02  xktape@4,0:cn
crw-rw-rw-   1 root sys   33,144 Aug 29 00:02  xktape@4,0:h
crw-rw-rw-   1 root sys   33,208 Aug 29 00:02  xktape@4,0:hb
crw-rw-rw-   1 root sys   33,212 Aug 29 00:02  xktape@4,0:hbn
crw-rw-rw-   1 root sys   33,148 Aug 29 00:02  xktape@4,0:hn
crw-rw-rw-   1 root sys   33,128 Aug 29 00:02  xktape@4,0:l
crw-rw-rw-   1 root sys   33,192 Aug 29 00:02  xktape@4,0:lb
crw-rw-rw-   1 root sys   33,196 Aug 29 00:02  xktape@4,0:lbn
crw-rw-rw-   1 root sys   33,132 Aug 29 00:02  xktape@4,0:ln
crw-rw-rw-   1 root sys   33,136 Aug 29 00:02  xktape@4,0:m
crw-rw-rw-   1 root sys   33,200 Aug 29 00:02  xktape@4,0:mb
crw-rw-rw-   1 root sys   33,204 Aug 29 00:02  xktape@4,0:mbn
crw-rw-rw-   1 root sys   33,140 Aug 29 00:02  xktape@4,0:mn
crw-rw-rw-   1 root sys   33,140 Aug 29 00:02  xktape@4,0:n
```

/dev/rmt will contain the logical tape devices (symbolic links to tape devices in **/devices**).

```
# ls -l /dev/dsk
/dev/rmt/0    -> ../../devices/[....]/xktape@4,0:
/dev/rmt/0b   -> ../../devices/[....]/xktape@4,0:b
/dev/rmt/0bn  -> ../../devices/[....]/xktape@4,0:bn
```

```
/dev/rmt/0c   -> ../../devices/[....]/xktape@4,0:c
/dev/rmt/0cb  -> ../../devices/[....]/xktape@4,0:cb
/dev/rmt/0cbn -> ../../devices/[....]/xktape@4,0:cbn
/dev/rmt/0cn  -> ../../devices/[....]/xktape@4,0:cn
/dev/rmt/0h   -> ../../devices/[....]/xktape@4,0:h
/dev/rmt/0hb  -> ../../devices/[....]/xktape@4,0:hb
/dev/rmt/0hbn -> ../../devices/[....]/xktape@4,0:hbn
/dev/rmt/0hn  -> ../../devices/[....]/xktape@4,0:hn
/dev/rmt/0l   -> ../../devices/[....]/xktape@4,0:l
/dev/rmt/0lb  -> ../../devices/[....]/xktape@4,0:lb
/dev/rmt/0lbn -> ../../devices/[....]/xktape@4,0:lbn
/dev/rmt/0ln  -> ../../devices/[....]/xktape@4,0:ln
/dev/rmt/0m   -> ../../devices/[....]/xktape@4,0:m
/dev/rmt/0mb  -> ../../devices/[....]/xktape@4,0:mb
/dev/rmt/0mbn -> ../../devices/[....]/xktape@4,0:mbn
/dev/rmt/0mn  -> ../../devices/[....]/xktape@4,0:mn
/dev/rmt/0n   -> ../../devices/[....]/xktape@4,0:n
```

FILES

/dev/rmt/*	logical tape devices
/devices/*	tape device nodes

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

add_drv(1M), **devlinks**(1M), **disks**(1M), **drvconfig**(1M), **ports**(1M), **attributes**(5), **esp**(7D), **mtio**(7I), **attach**(9E), **ddi_create_minor_node**(9F)

Writing Device Drivers

BUGS

tapes silently ignores malformed minor device names.

NAME

taskstat – prints ASET tasks status

SYNOPSIS

/usr/aset/util/taskstat [–**d** *aset_dir*]

DESCRIPTION

taskstat is located in the **/usr/aset/util** directory. **/usr/aset** is the default operating directory of the Automated Security Enhancement Tool (ASET). An alternative working directory can be specified by the administrators through the **aset** –**d** command or the **ASETDIR** environment variable. See **aset**(1M). Because **aset** dispatches its tasks to run in the background, when it returns, these tasks may or may not have completed. **taskstat** prints the status of the tasks, listing those that are completed and those that are still executing.

The ASET reports, which are located in the **/usr/aset/reports** directory (see the –**d** option), are not complete until all the tasks finish executing.

OPTIONS

–**d** *aset_dir* Specify the working directory for ASET. By default, this directory is **/usr/aset**. With this option, the **reports** directory will be located under *aset_dir*.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWast

SEE ALSO

aset(1M), **attributes**(5)

System Administration Guide

NAME

tcxconfig – configure the default linearity of the 24-bit TrueColor Visual for OpenWindows on a system with an S24 frame buffer (TCX)

SYNOPSIS

/usr/sbin/tcxconfig [**linear** | **nonlinear**]

DESCRIPTION

The **tcxconfig** script changes the default linearity of a 24-bit TrueColor Visual for OpenWindows on a system with an S24 frame buffer. When the S24 graphics driver for OpenWindows is installed, the default 24-bit TrueColor Visual is nonlinear. You can run **tcxconfig** with an argument that specifies the setting you want.

OpenWindows should not be running when you execute the **tcxconfig** script with an option. Start OpenWindows after **tcxconfig** has set the linearity you desire.

OPTIONS

If you specify no option, **tcxconfig** displays the current default setting.

You must become superuser before you can execute **tcxconfig** with one of the following options.

linear Set linear visual to be the default 24-bit TrueColor Visual. This means colors will be gamma-corrected.

nonlinear Set nonlinear visual to be the default 24-bit TrueColor Visual.

EXIT STATUS

The following exit values are returned:

0 success

1 an error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtcxow

SEE ALSO

attributes(5)

NAME

tic – terminfo compiler

SYNOPSIS

tic [–**v**[*n*]] [–**c**] *file*

DESCRIPTION

The command **tic** translates a **terminfo** file from the source format into the compiled format. The results are placed in the directory **/usr/share/lib/terminfo**. The compiled format is necessary for use with the library routines in **curses**(3X).

If the environment variable **TERMINFO** is set, the compiled results are placed there instead of **/usr/share/lib/terminfo**.

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes. Terminal names exceeding 14 characters will be truncated to 14 characters and a warning message will be printed.

OPTIONS

–**v***n* Specify that (verbose) output be written to standard error trace information showing **tic**'s progress. The optional integer *n* is a number from 1 to 10, indicating the desired level of detail of information. If *n* is omitted, the default level is 1. If *n* is specified and greater than 1, the level of detail is increased.

–**c** Specifies to check only *file* for errors. Errors in **use=** links are not detected.

file Contains one or more **terminfo** terminal descriptions in source format [see **terminfo**(4)]. Each description in the file describes the capabilities of a particular terminal. When a **use**=*entry*-*name* field is discovered in a terminal entry currently being compiled, **tic** reads in the binary from **/usr/share/lib/terminfo** to complete the entry. (Entries created from *file* will be used first. If the environment variable **TERMINFO** is set, that directory is searched instead of **/usr/share/lib/terminfo**.) **tic** duplicates the capabilities in *entry*-*name* for the current entry, with the exception of those capabilities that are explicitly defined in the current entry.

FILES

/usr/share/lib/terminfo/?/* compiled terminal description database

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

captoinfo(1M), **infocmp**(1M), **curses**(3X), **terminfo**(4), **attributes**(5)

NOTES

When an entry, for example, **entry_name_1**, contains a **use=***entry_name_2* field, any canceled capabilities in *entry_name_2* must also appear in **entry_name_1** before **use=** for these capabilities to be canceled in **entry_name_1**.

NAME

ttyadm – format and output port monitor-specific information

SYNOPSIS

/usr/sbin/ttyadm [**–b**] [**–c**] [**–h**] [**–I**] [**–r** *count*] [**–i** *msg*] [**–m** *modules*] [**–p** *prompt*]
[**–t** *timeout*] [**–S y | n**] [**–T** *termtype*] **–d** *device* **–l** *ttylabel* **–s** *service*
/usr/sbin/ttyadm –V

DESCRIPTION

The **ttyadm** command is an administrative command that formats **ttymon**(1M)-SPECIFIC information and writes it to standard output. The Service Access Facility (SAF) requires each port monitor to provide such a command. Note that the port monitor administrative file is updated by the Service Access Controller's administrative commands, **sacadm**(1M) and **pmadm**(1M). **ttyadm** provides a means of presenting formatted port monitor-specific (**ttymon**-specific) data to these commands.

OPTIONS

–b Set the ‘‘bi-directional port’’ flag. When this flag is set, the line can be used in both directions. **ttymon** will allow users to connect to the service associated with the port, but if the port is free, **uucico**(1M), **cu**(1C), or **ct**(1C) can use it for dialing out.

–c Set the connect-on-carrier flag for the port. If the **–c** flag is set, **ttymon** will invoke the port's associated service immediately when a connect indication is received (that is, no prompt is printed and no baud-rate searching is done).

–h Set the hangup flag for the port. If the **–h** flag is not set, **ttymon** will force a hangup on the line by setting the speed to **0** before setting the speed to the default or specified value.

–I Initialize only. If the **–I** option is used, **ttymon** will invoke the service only once. This can be used to configure a particular device without actually monitoring it, as with software carrier.

–d *device* *device* is the full pathname of the device file for the TTY port.

–i *message* Specify the inactive (disabled) response message. This message will be sent to the TTY port if the port is disabled or the **ttymon** monitoring the port is disabled.

–l *ttylabel* Specify which *ttylabel* in the **/etc/ttydefs** file to use as the starting point when searching for the proper baud rate.

–m *modules* Specify a list of pushable STREAMS modules. The modules will be pushed in the order in which they are specified before the service is invoked. *modules* must be a comma-separated list of modules, with no white space included. Any modules currently on the stream will be popped before these modules are pushed.

–p *prompt* Specify the prompt message, for example, ‘‘**login:**’’.

–r *count* When the **–r** option is invoked, **ttymon** will wait until it receives data from the port before it displays a prompt. If *count* is **0**, **ttymon** will wait until it receives any character. If *count* is greater than **0**, **ttymon** will wait until *count* newlines have been received.

–s *service* *service* is the full pathname of the service to be invoked when a connection request is received. If arguments are required, the command and its arguments must be enclosed in double quotes (" ").

–t *timeout* Specify that **ttymon** should close a port if the open on the port succeeds, and no input data is received in *timeout* seconds.

–S y | n Set the software carrier value. **y** will turn software carrier on. **n** will turn software carrier off.

–T *termtype* Set the terminal type. The **TERM** environment variable will be set to *termtype*.

–V Display the version number of the current **/usr/lib/saf/ttymon** command.

OUTPUT

If successful, **ttyadm** will generate the requested information, write it to standard output, and exit with a status of **0**. If **ttyadm** is invoked with an invalid number of arguments or invalid arguments, or if an incomplete option is specified, an error message will be written to standard error and **ttymon** will exit with a non-zero status.

FILES

/etc/ttydefs

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ct(1C), **cu**(1C), **pmadm**(1M), **sacadm**(1M), **ttymon**(1M), **uucico**(1M), **attributes**(5)

System Administration Guide

NAME

ttymon – port monitor for terminal ports

SYNOPSIS

/usr/lib/saf/ttymon
/usr/lib/saf/ttymon **−g** [**−d** *device*] [**−h**] [**−t** *timeout*] [**−l** *ttylabel*] [**−p** *prompt*]
[**−m** *modules*] [**−T** *termtype*]

DESCRIPTION

ttymon is a STREAMS-based TTY port monitor. Its function is to monitor ports, to set terminal modes, baud rates, and line disciplines for the ports, and to connect users or applications to services associated with the ports. Normally, **ttymon** is configured to run under the Service Access Controller, **sac**(1M), as part of the Service Access Facility (SAF). It is configured using the **sacadm**(1M) command. Each instance of **ttymon** can monitor multiple ports. The ports monitored by an instance of **ttymon** are specified in the port monitor's administrative file. The administrative file is configured using the **pmadm**(1M) and **ttyadm**(1M) commands. When an instance of **ttymon** is invoked by the **sac** command, it starts to monitor its ports. For each port, **ttymon** first initializes the line disciplines, if they are specified, and the speed and terminal settings. For ports with entries in **/etc/logindevperm**, device owner, group and permissions are set. (See **logindevperm**(4).) The values used for initialization are taken from the appropriate entry in the TTY settings file. This file is maintained by the **sttydefs**(1M) command. Default line disciplines on ports are usually set up by the **autopush**(1M) command of the Autopush Facility.

ttymon then writes the prompt and waits for user input. If the user indicates that the speed is inappropriate by pressing the BREAK key, **ttymon** tries the next speed and writes the prompt again. When valid input is received, **ttymon** interprets the per-service configuration file for the port, if one exists, creates a **utmp** entry if required (see **utmp**(4)), establishes the service environment, and then invokes the service associated with the port. Valid input consists of a string of at least one non-newline character, terminated by a carriage return. After the service terminates, **ttymon** cleans up the **utmp** entry, if one exists, and returns the port to its initial state.

If *autobaud* is enabled for a port, **ttymon** will try to determine the baud rate on the port automatically. Users must enter a carriage return before **ttymon** can recognize the baud rate and print the prompt. Currently, the baud rates that can be determined by *autobaud* are **110**, **1200**, **2400**, **4800**, and **9600**.

If a port is configured as a bidirectional port, **ttymon** will allow users to connect to a service, and, if the port is free, will allow **uucico**(1M), **cu**(1C), or **ct**(1C) to use it for dialing out. If a port is bidirectional, **ttymon** will wait to read a character before it prints a prompt.

If the *connect-on-carrier* flag is set for a port, **ttymon** will immediately invoke the port's associated service when a connection request is received. The prompt message will not be sent.

If a port is disabled, **ttymon** will not start any service on that port. If a disabled message is specified, **ttymon** will send out the disabled message when a connection request is received. If **ttymon** is disabled, all ports under that instance of **ttymon** will also be disabled.

SERVICE INVOCATION

The service **ttymon** invokes for a port is specified in the **ttymon** administrative file. **ttymon** will scan the character string giving the service to be invoked for this port, looking for a **%d** or a **%%** two-character sequence. If **%d** is found, **ttymon** will modify the service command to be executed by replacing those two characters by the full path name of this port (the device name). If **%%** is found, they will be replaced by a single **%**.

When the service is invoked, file descriptor **0**, **1**, and **2** are opened to the port device for reading and writing. The service is invoked with the user ID, group ID and current home directory set to that of the user name under which the service was registered with **ttymon**. Two environment variables, **HOME** and **TTYPROMPT,** are added to the service's environment by **ttymon**. **HOME** is set to the home directory of the user name under which the service is invoked. **TTYPROMPT** is set to the prompt string configured for the service on the port. This is provided so that a service invoked by **ttymon** has a means of determining if a prompt was actually issued by **ttymon** and, if so, what that prompt actually was.

See **ttyadm**(1M) for options that can be set for ports monitored by **ttymon** under the Service Access Controller.

SECURITY

ttymon uses **pam**(3) for session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **ttymon**. Here is a partial **pam.conf** file with entries for **ttymon** using the UNIX session management module.

```
ttymon   session   required   /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **ttymon** service, then the entries for the "other" service will be used.

OPTIONS

–g A special invocation of **ttymon** is provided with the **–g** option. This form of the command should only be called by applications that need to set the correct baud rate and terminal settings on a port and then connect to **login** service, but that cannot be pre-configured under the SAC. The following combinations of options can be used with **–g**:

–d *device* *device* is the full path name of the port to which **ttymon** is to attach. If this option is not specified, file descriptor **0** must be set up by the invoking process to a TTY port.

–h If the -h flag is not set, **ttymon** will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed.

–l *ttylabel* *ttylabel* is a link to a speed and TTY definition in the **ttydefs** file. This definition tells **ttymon** at what speed to run initially, what the initial TTY settings are, and what speed to try next if the user indicates that the speed is inappropriate by pressing the BREAK key. The default speed is 9600 baud.

–m *modules* When initializing the port, **ttymon** will pop all modules on the port, and then push *modules* in the order specified. *modules* is a comma-separated list of pushable modules. Default modules on the ports are usually set up by the Autopush Facility.

–p *prompt* Allows the user to specify a prompt string. The default prompt is **Login:** .

–t *timeout* Specifies that **ttymon** should exit if no one types anything in *timeout* seconds after the prompt is sent.

–T *termtype* Sets the **TERM** environment variable to *termtype*.

ENVIRONMENT

If any of the **LC_**∗ variables (**LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC,** and **LC_MONETARY**) (see **environ**(5)) are not set in the environment, the operational behavior of **ttymon** for each corresponding locale category is determined by the value of the **LANG** environment variable. If **LC_ALL** is set, its contents are used to override both the **LANG** and the other **LC_**∗ variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how **ttymon** behaves.

LC_CTYPE

Determines how **ttymon** handles characters. When **LC_CTYPE** is set to a valid value, **ttymon** can display and handle text and filenames containing valid characters for that locale. **ttymon** can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. **ttymon** can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

FILES

/etc/logindevperm

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ct(1C), **cu**(1C), **autopush**(1M), **pmadm**(1M), **sac**(1M), **sacadm**(1M), **sttydefs**(1M), **ttyadm**(1M), **uucico**(1M), **pam**(3), **logindevperm**(4), **pam.conf**(4), **utmp**(4), **attributes**(5), **environ**(5), **pam_unix**(5)

System Administration Guide

NOTES

If a port is monitored by more than one **ttymon**, it is possible for the **ttymon**s to send out prompt messages in such a way that they compete for input.

NAME

tunefs – tune up an existing file system

SYNOPSIS

tunefs [**–a** *maxcontig*] [**–d** *rotdelay*] [**–e** *maxbpg*] [**–m** *minfree*]
[**–o** [**space** | **time**]] *special* | *filesystem*

DESCRIPTION

tunefs is designed to change the dynamic parameters of a file system which affect the layout policies. The file system must be unmounted before using **tunefs**. When using **tunefs** with **filesystem**, **filesystem** must be in **/etc/vfstab**. The parameters which are to be changed are indicated by the options given below.

Generally one should optimize for time unless the file system is over 90% full.

OPTIONS

–a *maxcontig* Specify the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see **–d** below). The default value is **1**, since most device drivers require an interrupt per disk transfer. Device drivers that can chain several buffers together in a single transfer should set this to the maximum chain length.

–d *rotdelay* Specify the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

–e *maxbpg* Indicate the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to approximately one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

–m *minfree* Specify the percentage of space held back from normal users; the minimum free space threshold. The default value used is 10%. This value can be set to **0**, however up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. Note: If the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

–o [**space** | **time**]
Change optimization strategy for the file system.

space: conserve space
time: attempt to organize file layout to minimize access time

USAGE

See **largefile**(5) for the description of the behavior of **tunefs** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mkfs(1M), **fork**(2), **terminfo**(4), **attributes**(5), **largefile**(5)

NAME

uadmin – administrative control

SYNOPSIS

/sbin/uadmin *cmd fcn*

DESCRIPTION

The **uadmin** command provides control for basic administrative functions. This command is tightly coupled to the System Administration procedures and is not intended for general use. It may be invoked only by the super-user.

Both *cmd* (command) and *fcn* (function) are converted to integers and passed to the **uadmin** system call.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

uadmin(2), **attributes**(5)

NAME

ufsdump – incremental file system dump

SYNOPSIS

/usr/sbin/ufsdump [*options*] [*arguments*] *files_to_dump*

DESCRIPTION

ufsdump backs up all files specified by *files_to_dump* (normally either a whole file system or files within a file system changed after a certain date) to magnetic tape, diskette, or disk file. When running **ufsdump**, the file system must be inactive; otherwise, the output of **ufsdump** may be inconsistent and restoring files correctly may be impossible. A file system is inactive when it is unmouned or the system is in single user mode. A file system is not considered inactive if one tree of the file system is quiescent while another tree has files or directories being modified.

options is a single string of one-letter **ufsdump** options.

arguments may be multiple strings whose association with the options is determined by order. That is, the first argument goes with the first option that takes an argument; the second argument goes with the second option that takes an argument, and so on.

files_to_dump is required and must be the last argument on the command line. See **OPERANDS** for more information.

With most devices **ufsdump** can automatically detect the end-of-media. Consequently, the **d**, **s**, and **t** options are not necessary for multi-volume dumps, unless **ufsdump** does not understand the way the device detects the end-of-media, or the files are to be restored on a system with an older version of the **restore** command.

OPTIONS

The following options are supported:

0–9 The “dump level.” All files specified by *files_to_dump* that have been modified since the last **ufsdump** at a lower dump level are copied to the *dump_file* destination (normally a magnetic tape device). For instance, if a “level **2**” dump was done on Monday, followed by a “level **4**” dump on Tuesday, a subsequent “level **3**” dump on Wednesday would contain all files modified or added since the “level **2**” (Monday) backup. A “level **0**” dump copies the entire file system to the *dump_file.*

a *archive_file* Archive file. Archive a dump table-of-contents in the specified *archive_file* to be used by **ufsrestore**(1M) to determine whether a file is in the dump file that is being restored.

b *factor* Blocking factor. Specify the blocking factor for tape writes. The default is 20 blocks per write for tapes of density less than 6250BPI (bytes-per-inch). The default blocking factor for tapes of density 6250BPI and greater is 64. The default blocking factor for cartridge tapes (**c** option) is 126. The highest blocking factor available with most tape drives is 126. Note: the blocking factor is specified in terms of 512-byte blocks, for compatibility with **tar**(1).

c Cartridge. Set the defaults for cartridge instead of the standard half-inch reel. This sets the density to 1000BPI and the blocking factor to 126. Since **ufsdump** can automatically detect the end-of-media, only the blocking parameter normally has an effect. When cartridge tapes are used, and this option is *not* specified, **ufsdump** will slightly miscompute the size of the tape. If the **b**, **d**, **s** or **t** options are specified with this option, their values will override the defaults set by this option.

d *bpi* Tape density. Not normally required, as **ufsdump** can detect end-of-media. This parameter can be used to keep a running tab on the amount of tape used per reel. The default density is 6250BPI except when the **c** option is used for cartridge tape, in which case it is assumed to be 1000BPI per track. Typical values for tape devices are:

1/2" tape	6250 BPI
1/4" cartridge	1000 BPI The tape densities and other options are documented in the **st**(7D) man page.

D Diskette. Dump to diskette.

f *dump_file* Dump file. Use *dump_file* as the file to dump to, instead of **/dev/rmt/0**. If *dump_file* is specified as –, dump to standard output.

If the name of the file is of the form *machine***:***device*, the dump is done from the specified machine over the network using **rmt**(1M). Since **ufsdump** is normally run by root, the name of the local machine must appear in the **/.rhosts** file of the remote machine. If the file is specified as *user***@***machine***:***device*, **ufsdump** will attempt to execute as the specified user on the remote machine. The specified user must have a **.rhosts** file on the remote machine that allows the user invoking the command from the local machine to access the remote machine.

l Autoload. When the end-of-tape is reached before the dump is complete, take the drive offline and wait up to two minutes for the tape drive to be ready again. This gives autoloading (stackloader) tape drives a chance to load a new tape. If the drive is ready within two minutes, continue. If it is not, prompt for another tape and wait.

n Notify all operators in the **sys** group that **ufsdump** requires attention by sending messages to their terminals, in a manner similar to that used by the **wall**(1M) command. Otherwise, such messages are sent only to the terminals (such as the console) on which the user running **ufsdump** is logged in.

o Offline. Take the drive offline when the dump is complete or the end-of-media is reached and rewind the tape, or eject the diskette. In the case of some autoloading 8mm drives, the tape is removed from the drive automatically. This prevents another process which rushes in to use the drive, from inadvertently overwriting the media.

s *size*

Specify the *size* of the volume being dumped to. Not normally required, as **ufsdump** can detect end-of-media. When the specified size is reached, **ufsdump** waits for you to change the volume. **ufsdump** interprets the specified size as the length in feet for tapes and cartridges, and as the number of 1024-byte blocks for diskettes. The values should be a little smaller than the actual physical size of the media (for example, 425 for a 450-foot cartridge). Typical values for tape devices depend on the **c** option, for cartridge devices, and the **D** option for diskettes:

1/2" tape	2300 feet
60-Mbyte 1/4" cartridge	425 feet
150-Mbyte 1/4" cartridge	700 feet
diskette	1422 blocks (Corresponds to a 1.44-Mbyte diskette, with one cylinder reserved for bad block information.)

S

Size estimate. Determine the amount of space that is needed to perform the dump without actually doing it, and display the estimated number of bytes it will take. This is useful with incremental dumps to determine how many volumes of media will be needed.

t *tracks*

Specify the number of tracks for a cartridge tape. Not normally required, as **ufsdump** can detect end-of-media. The default is 9 tracks. The **t** option is not compatible with the **D** option. Values for Sun-supported tape devices are:

60-Mbyte 1/4" cartridge	9 tracks
150-Mbyte 1/4" cartridge	18 tracks

u

Update the dump record. Add an entry to the file **/etc/dumpdates,** for each file system successfully dumped that includes the file system name, date, and dump level.

v

Verify. After each tape or diskette is written, verify the contents of the media against the source file system. If any discrepancies occur, prompt for new media, then repeat the dump/verification process. The file system *must* be unmounted. This option cannot be used to verify a dump to standard output.

w

Warning. List the file systems that have not been backed up within a day. This information is gleaned from the files **/etc/dumpdates** and **/etc/vfstab**. When the **w** option is used, all other options are ignored. After reporting, **ufsdump** exits immediately.

W

Warning with highlight. Similar to the **w** option, except that the **W** option includes all file systems that appear in **/etc/dumpdates**, along with information about their most recent dump dates and levels. File systems that have not been backed up within a day are highlighted.

OPERANDS

The following operand is supported:

files_to_dump Specifies the files to dump. Usually it identifies a whole file system by its raw device name (for example, **/dev/rdsk/c0t3d0s6**). Incremental dumps (levels **1** to **9**) of files changed after a certain date only apply to a whole file system. Alternatively, *files_to_dump* can identify individual files or directories. All files or directories are dumped, which is equivalent to a level **0** dump; however, **/etc/dumpdates** is not updated, even with the **u** option specified. In all cases, the files must be contained in the same file system, and the file system must be local to the system where **ufsdump** is being run.

files_to_dump is required and must be the last argument on the command line.

If no *options* are given, the default is **9uf /dev/rmt/0** *files_to_dump*.

USAGE

See **largefile**(5) for the description of the behavior of **ufsdump** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

To make a full dump of a root file system on **c0t3d0**, on a 150-MByte cartridge tape unit **0**, use:

```
example# ufsdump 0cfu /dev/rmt/0 /dev/rdsk/c0t3d0s0
```

To make and verify an incremental dump at level **5** of the **usr** partition of **c0t3d0**, on a 1/2" reel tape unit **1,** use:

```
example# ufsdump 5fuv /dev/rmt/1 /dev/rdsk/c0t3d0s6
```

EXIT STATUS

While running, **ufsdump** emits many verbose messages. **ufsdump** returns the following exit values:

0	Normal exit.
1	Startup errors encountered.
3	Abort – no checkpoint attempted.

FILES

/dev/rmt/0	default unit to dump to
/etc/dumpdates	dump date record
/etc/group	to find group **sys**
/etc/hosts	to gain access to remote system with drive
/etc/vfstab	list of file systems

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cpio(1), **tar**(1), **dd**(1M), **devnm**(1M), **prtvtoc**(1M), **rmt**(1M), **shutdown**(1M), **ufsrestore**(1M), **volcopy**(1M), **wall**(1M), **attributes**(5), **largefile**(5), **st**(7D)

NOTES

Read Errors

Fewer than 32 read errors on the file system are ignored.

Process Per Reel

Because each reel requires a new process, parent processes for reels that are already written hang around until the entire tape is written.

Operator Intervention

ufsdump requires operator intervention on these conditions: end of volume, end of dump, volume write error, volume open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **n** option, **ufsdump** interacts with the operator on **ufsdump**'s control terminal at times when **ufsdump** can no longer proceed, or if something is grossly wrong. All questions **ufsdump** poses *must* be answered by typing **yes** or **no**, as appropriate.

Since backing up a disk can involve a lot of time and effort, **ufsdump** checkpoints at the start of each volume. If writing that volume fails for some reason, **ufsdump** will, with operator permission, restart itself from the checkpoint after a defective volume has been replaced.

Suggested Dump Schedule

It is vital to perform full, “level **0**”, dumps at regular intervals. When performing a full dump, bring the machine down to single-user mode using **shutdown**(1M). While preparing for a full dump, it is a good idea to clean the tape drive and heads. Incremental dumps should be performed with the system running in single-user mode.

Incremental dumps allow for convenient backup and recovery of active files on a more frequent basis, with a minimum of media and time. However, there are some tradeoffs. First, the interval between backups should be kept to a minimum (once a day at least). To guard against data loss as a result of a media failure (a rare, but possible occurrence), capture active files on (at least) two sets of dump volumes. Another consideration is the desire to keep unnecessary duplication of files to a minimum to save both operator time and media storage. A third consideration is the ease with which a particular backed-up version of a file can be located and restored. The following four-week schedule offers a reasonable tradeoff between these goals.

	Sun	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Week 1:	**Full**	**5**	**5**	**5**	**5**	**3**
Week 2:		**5**	**5**	**5**	**5**	**3**
Week 3:		**5**	**5**	**5**	**5**	**3**
Week 4:		**5**	**5**	**5**	**5**	**3**

Although the Tuesday through Friday incrementals contain “extra copies” of files from Monday, this scheme assures that any file modified during the week can be recovered from the previous day's incremental dump.

Process Priority of ufsdump

ufsdump uses multiple processes to allow it to read from the disk and write to the media concurrently. Due to the way it synchronizes between these processes, any attempt to run dump with a **nice** (process priority) of '–5' or better will likely make **ufsdump** run *slower* instead of faster.

Overlapping Partitions

Most disks contain one or more overlapping slices because slice 2 covers the entire disk. The other slices are of various sizes and usually do not overlap. For example, a common configuration places **root** on slice 0, **swap** on slice 1, **/opt** on slice 5 and **/usr** on slice 6.

It should be emphasized that **ufsdump** dumps one **ufs** file system at a time. Given the above scenario where slice 0 and slice 2 have the same starting offset, executing **ufsdump** on slice 2 with the intent of dumping the entire disk would instead dump only the **root** file system on slice 0. To dump the entire disk, the user must dump the file systems on each slice separately.

BUGS

The **/etc/vfstab** file does not allow the desired frequency of backup for file systems to be specified (as **/etc/fstab** did). Consequently, the **w** and **W** options assume file systems should be backed up daily, which limits the usefulness of these options.

NAME

ufsrestore – incremental file system restore

SYNOPSIS

/usr/lib/fs/ufs/ufsrestore i | **r** | **R** | **t** | **x** [**abcdfhmsvy**] [*archive_file*] [*factor*]
[*dumpfile*][*n*] [*filename* . . .]

DESCRIPTION

The **ufsrestore** utility restores files from backup media created with the **ufsdump** command. **ufsrestores**'s actions are controlled by the *key* argument. The *key* is exactly one *function letter* (**i**, **r**, **R** , **t**, or **x**) and zero or more *function modifier*s (letters). The *key* string contains no SPACE characters. Function modifier arguments are listed on the command line in the same order as their corresponding function modifiers appear in the *key* string.

OPTIONS

Function Letters

One (and only one) of the following function letters is required:

i Interactive. After reading in the directory information from the media, **ufsrestore** invokes an interactive interface that allows you to browse through the dump file's directory hierarchy and select individual files to be extracted. See **Interactive Commands**, below, for a description of available commands.

r Recursive. Restore the entire contents of the media into the current directory (which should be the top-level of the file system). To completely restore a file system, use this function letter to restore the level **0** dump, and again for each incremental dump. Although, this function letter is intended for a complete restore onto a clear file system, if the file system contains files not on the media, they are preserved.

R Resume restoring. **ufsrestore** requests a particular volume of a multivolume set from which to resume a full restore (see the **r** function letter above). This allows **ufsrestore** to start from a checkpoint when it is interrupted in the middle of a full restore.

t Table of contents. List each *filename* that appears on the media. If no *filename* argument is given, the root directory is listed. This results in a list of all files on the media, unless the **h** function modifier is in effect. The table of contents is taken from the media or from the specified archive file, when the **a** function modifier is used. This function modifier is mutually exclusive with the **x** and **r** function letters.

x Extract the named files from the media. If a named file matches a directory whose contents were written onto the media, and the **h** modifier is not in effect, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). Existing files are overwritten and a warning is given. If no *filename* argument is given, the root directory is extracted. This results in the entire tape being extracted unless the **h** modifier is in effect.

Function Modifiers

a *archive_file* Read the table of contents from *archive_file* instead of the media. This function modifier can be used in combination with the **t**, **i**, or **x** function letters, making it possible to check whether files are on the media without having to mount the media. When used with the **x** and interactive (**i**) function letters, it prompts for the volume containing the file(s) before extracting them.

b *factor* Blocking factor. Specify the blocking factor for tape reads. For variable length SCSI tape devices, unless the data was written with the default blocking factor, a blocking factor at least as great as that used to write the tape must be used; otherwise, an error will be generated. Note that a tape block is 512 bytes. Refer to the man page for your specific tape driver for the maximum blocking factor.

c Convert the contents of the media in 4.1BSD format to the new **ufs** file system format.

d Debug. Turn on debugging output.

f *dump_file* Use *dump_file* instead of **/dev/rmt/0** as the file to restore from. Typically *dump_file* specifies a tape or diskette drive. If *dump_file* is specified as '–', **ufsrestore** reads from the standard input. This allows **ufsdump**(1M) and **ufsrestore** to be used in a pipeline to copy a file system:

```
example# ufsdump 0f – /dev/rdsk/c0t0d0s7 | (cd /home;ufsrestore xf –)
```

If the name of the file is of the form *machine*:*device*, the restore is done from the specified machine over the network using **rmt**(1M). Since **ufsrestore** is normally run by root, the name of the local machine must appear in the **/.rhosts** file of the remote machine. If the file is specified as *user@machine*:*device*, **ufsrestore** will attempt to execute as the specified user on the remote machine. The specified user must have a **.rhosts** file on the remote machine that allows the user invoking the command from the local machine to access the remote machine.

h Extract or list the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

m Extract by inode numbers rather than by filename to avoid regenerating complete pathnames. Regardless of where the files are located in the dump hierarchy, they are restored into the current directory and renamed with their inode number. This is useful if only a few files are being extracted.

s *n* Skip to the *n*'th file when there are multiple dump files on the same tape. For example, the command:

```
example# ufsrestore xfs /dev/rmt/0hn 5
```

would position you at the fifth file on the tape.

v Verbose. **ufsrestore** displays the name and inode number of each file it restores, preceded by its file type.

y Do not ask whether to abort the restore in the event of tape errors. **ufsrestore** tries to skip over the bad tape block(s) and continue as best it can.

Interactive Commands

ufsrestore enters interactive mode when invoked with the **i** function letters. Interactive commands are reminiscent of the shell. For those commands that accept an argument, the default is the current directory. The interactive options are:

add [*filename*] Add the named file or directory to the list of files to extract. If a directory is specified, add that directory and its files (recursively) to the extraction list (unless the **h** modifier is in effect).

cd *directory* Change to *directory* (within the dump file).

delete [*filename*] Delete the current directory, or the named file or directory from the list of files to extract. If a directory is specified, delete that directory and all its descendents from the extraction list (unless the **h** modifier is in effect). The most expedient way to extract a majority of files from a directory is to add that directory to the extraction list, and then delete specific files to omit.

extract Extract all files on the extraction list from the dump media. **ufsrestore** asks which volume the user wishes to mount. The fastest way to extract a small number of files is to start with the last volume and work toward the first.

help Display a summary of the available commands.

ls [*directory*] List files in *directory* or the current directory, represented by a '**.**' (period). Directories are appended with a '/' (slash). Entries marked for extraction are prefixed with a '*' (asterisk). If the verbose option is in effect, inode numbers are also listed.

pwd Print the full pathname of the current working directory.

quit **ufsrestore** exits immediately, even if the extraction list is not empty.

setmodes Prompts: **set owner/mode for** '**.**' (period). Type **y** for yes to set the mode (permissions, owner, times) of the current directory '**.**' (period) into which files are being restored equal to the mode of the root directory of the file system from which they were dumped. Normally, this is what you want when restoring a whole file system, or restoring individual files into the same locations from which they were dumped. Type **n** for no, to leave the mode of the current directory unchanged. Normally, this is what you want when restoring part of a dump to a directory other than the one from which the files were dumped.

verbose Toggle the status of the **v** modifier. While **v** is in effect, the **ls** command lists the inode numbers of all entries, and **ufsrestore** displays information about each file as it is extracted.

what Display the dump header on the media.

OPERANDS

The following operands are supported.

filename Specifies the pathname of files (or directories) to be restored to disk. Unless the **h** function modifier is also used, a directory name refers to the files it contains, and (recursively) its subdirectories and the files they contain. *filename* is associated with either the **x** or **t** function letters, and must come last.

USAGE

See **largefile**(5) for the description of the behavior of **ufsrestore** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred. Verbose messages are displayed.

FILES

/dev/rmt/0	the default tape drive
/tmp/rstdir∗	file containing directories on the tape
/tmp/rstmode∗	owner, mode, and timestamps for directories
./restoresymtable	information passed between incremental restores

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mkfs(1M), **mount**(1M), **rmt**(1M), **ufsdump**(1M), **attributes**(5), **largefile**(5)

DIAGNOSTICS

ufsrestore complains about bad option characters.

Read errors result in complaints. If **y** has been specified, or the user responds **y**, **ufsrestore** will attempt to continue.

If the dump extends over more than one tape, **ufsrestore** asks the user to change tapes. If the **x** or **i** function letter has been specified, **ufsrestore** also asks which volume the user wishes to mount.

There are numerous consistency checks that can be listed by **ufsrestore**. Most checks are self-explanatory or can “never happen”. Common errors are given below.

Converting to new file system format

A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

filename: **not found on tape**

The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, or from using a dump tape created on an active file system.

expected next file *inumber*, **got** *inumber*

A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

Incremental tape too low

When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

Incremental tape too high

When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or one that has too high an incremental level has been loaded.

media read error: invalid argument

Blocking factor specified for read is smaller than the blocking factor used to write data.

Tape read error while restoring *filename*
Tape read error while skipping over inode inumber
Tape read error while trying to resynchronize
A tape read error has occurred

If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

resync ufsrestore, skipped *num*

After a tape read error, **ufsrestore** may have to resynchronize itself. This message lists the number of blocks that were skipped over.

NOTES

ufsrestore can get confused when doing incremental restores from dump tapes that were made on active file systems.

A “level **0**” dump must be done after a full restore. Because **ufsrestore** runs in user mode, it has no control over inode allocation. This means that **ufsrestore** repositions the files, although it does not change their contents. Thus, a full dump must be done to get a new set of directories reflecting the new file positions, so that later incremental dumps will be correct.

NAME

unshare – make local resource unavailable for mounting by remote systems

SYNOPSIS

unshare [–**F** *FSType*] [–**o** *specific_options*] [*pathname* | *resourcename*]

DESCRIPTION

The **unshare** command makes a shared local resource unavailable as file system type *FSType*. If the option –**F** *FSType* is omitted, then the first file system type listed in file **/etc/dfs/fstypes** will be used as the default. *Specific_options*, as well as the semantics of *resourcename*, are specific to particular distributed file systems.

OPTIONS

–**F** *FSType* Specify the file system type.

–**o** *specific_options* Specify options specific to the file system provided by the –**F** option.

FILES

/etc/dfs/fstypes
/etc/dfs/sharetab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

share(1M), **shareall**(1M), **attributes**(5)

NOTES

If *pathname* or *resourcename* is not found in the shared information, an error message will be sent to standard error.

NAME

unshare_nfs – make local NFS filesystems unavailable for mounting by remote systems

SYNOPSIS

unshare [**–F nfs**] *pathname*

DESCRIPTION

The **unshare** command makes local filesystems unavailable for mounting by remote systems. The shared filesystem must correspond to a line with NFS as the *FSType* in the file **/etc/dfs/sharetab**.

OPTIONS

–F This option may be omitted if NFS is the first file system type listed in the file **/etc/dfs/fstypes**.

FILES

/etc/dfs/fstypes
/etc/dfs/sharetab

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

share(1M), **attributes**(5)

NOTES

If the filesystem being unshared is a symbolic link to a valid pathname, the canonical path (the path which the symbolic link follows) will be unshared.

For example, if **/export/foo** is a symbolic link to **/export/bar** (**/export/foo -> /export/bar**), the following **unshare** command will result in **/export/bar** as the unshared pathname (and not **/export/foo**).

```
example# unshare -F nfs /export/foo
```

NAME

useradd – administer a new user login on the system

SYNOPSIS

useradd [−**c** *comment*] [−**d** *dir*] [−**e** *expire*] [−**f** *inactive*] [−**g** *group*]
[−**G** *group* [, *group* . . .]] [−**m** [−**k** *skel_dir*]] [−**u** *uid* [−**o**]] [−**s** *shell*] *login*

useradd −**D** [−**b** *base_dir*] [−**e** *expire*] [−**f** *inactive*] [−**g** *group*]

DESCRIPTION

useradd adds a new user entry to the **/etc/passwd** and **/etc/shadow** files. It also creates supplementary group memberships for the user (−**G** option) and creates the home directory (−**m** option) for the user if requested. The new login remains locked until the **passwd**(1) command is executed.

Specifying **useradd** −**D** with the −**g**, −**b**, −**f**, or −**e** options (or any combination of these) sets the default values for the respective fields. See the −**D** option below. Subsequent **useradd** commands without the −**D** option use these arguments.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options may exceed this limit.

The login field (*login*) is a string no more than eight bytes consisting of characters from the set of alphabetic characters, numeric characters, period (**.**), underscore (**_**), and hypen (**-**). The first character should be alphabetic and the field should contain at least one lower case alphabetic character. A warning message will be written if these restrictions are not met. A future Solaris release may refuse to accept login fields that do not meet these requirements. The login field must contain at least one character and must not contain a colon (**:**) or a newline (**n**).

OPTIONS

−**c** *comment* Any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's **/etc/passwd** entry.

−**d** *dir* The home directory of the new user. It defaults to *base_dir* / *login*, where *base_dir* is the base directory for new login home directories and *login* is the new login name.

−**e** *expire* Specify the expiration date for a login. After this date, no user will be able to access this login. *expire* is a date entered in any format you like (except a Julian date). If the date format that you choose includes spaces, it must be quoted. For example, you may enter **10/6/90** or "**October 6, 1990**". A null value (" ") defeats the status of the expired date. This option is useful for creating temporary logins.

−**f** *inactive* The maximum number of days allowed between uses of a login ID before that login ID is declared invalid. Normal values are positive integers. A value of **0** defeats the status.

−**g** *group* An existing group's integer ID or character-string name. Without the −**D** option, it defines the new user's primary group membership and defaults to the default group. You can reset this default value by invoking **useradd** −**D** −**g** *group*.

–**G** *group* An existing group's integer ID or character-string name. It defines the new user's supplementary group membership. Duplicates between *group* with the –**g** and –**G** options are ignored. No more than **NGROUPS_MAX** groups may be specified.

–**k** *skel_dir* A directory that contains skeleton information (such as **.profile**) that can be copied into a new user's home directory. This directory must already exist. The system provides the **/etc/skel** directory that can be used for this purpose.

–**m** Create the new user's home directory if it does not already exist. If the directory already exists, it must have read, write, and execute permissions by *group*, where *group* is the user's primary group.

–**s** *shell* Full pathname of the program used as the user's shell on login. It defaults to an empty field causing the system to use **/bin/sh** as the default. The value of *shell* must be a valid executable file.

–**u** *uid* The UID of the new user. This UID must be a non-negative decimal integer below **MAXUID** as defined in **<sys/param.h>**. The UID defaults to the next available (unique) number above the highest number currently assigned. For example, if UIDs 100, 105, and 200 are assigned, the next default UID number will be 201. (UIDs from 0-99 are reserved by SunOS for future applications.)

–**o** This option allows a UID to be duplicated (non-unique).

–**D** Display the default values for *group*, *base_dir*, *skel_dir*, *shell*, *inactive*, and *expire*. When used with the –**g**, –**b**, –**f**, or –**e** options, the –**D** option sets the default values for the specified fields. The default values are:

group	**other** (GID of 1)
base_dir	**/home**
skel_dir	**/etc/skel**
shell	**/bin/sh**
inactive	**0**
expire	Null (unset).

–**b** *base_dir* The default base directory for the system if –**d** *dir* is not specified. *base_dir* is concatenated with the user's login to define the home directory. If the –**m** option is not used, *base_dir* must exist.

FILES

/etc/passwd
/etc/shadow
/etc/group
/etc/skel
/usr/include/limits.h

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

passwd(1), **users**(1B), **groupadd**(1M), **groupdel**(1M), **groupmod**(1M), **grpck**(1M), **logins**(1M), **pwck**(1M), **userdel**(1M), **usermod**(1M), **passwd**(4), **attributes**(5)

DIAGNOSTICS

In case of an error, **useradd** prints an error message and exits with a non-zero status.

UX: useradd: ERROR: *login* **is already in use. Choose another.**
The *login* specified is already in use.

UX: useradd: ERROR: uid *uid* **is already in use. Choose another.**
The *uid* specified with the −**u** option is not unique.

UX: useradd: ERROR: group *group* **does not exist. Choose another.**
The *group* specified with the −**g** option is already in use.

UX: useradd: WARNING: uid *uid* **is reserved.**
The *uid* specified with the −**u** option is in the range of reserved UIDs (from 0-99).

UX: useradd: ERROR: uid *uid* **is too big. Choose another.**
The *uid* specified with the −**u** option exceeds **MAXUID** as defined in **<sys/param.h>**.

UX: useradd: ERROR: Cannot update system files - login cannot be created.
The **/etc/passwd** or **/etc/shadow** files do not exist.

NOTES

useradd only adds a user definition to the local system. If a network nameservice such as NIS or NIS+ is being used to supplement the local **/etc/passwd** file with additional entries, **useradd** cannot change information supplied by the network nameservice. However **useradd** will verify the uniqueness of the user name and user id and the existence of any group names specified against the external nameservice.

NAME

userdel – delete a user’s login from the system

SYNOPSIS

userdel [**–r**] *login*

DESCRIPTION

The **userdel** command deletes a user’s login from the system and makes the appropriate login-related changes to the system file and file system.

OPTIONS

–r Remove the user’s home directory from the system. This directory must exist. The files and directories under the home directory will no longer be accessible following successful execution of the command.

login A string of printable characters that specify an existing login on the system. It may not contain a colon (:) or a newline (\n).

FILES

/etc/passwd
/etc/shadow
/etc/group

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

passwd(1), **users**(1B), **groupadd**(1M), **groupdel**(1M), **groupmod**(1M), **logins**(1M), **useradd**(1M), **usermod**(1M), **passwd**(4), **attributes**(5)

DIAGNOSTICS

The **userdel** command exits with one of the following values:

0 Success.

2 Invalid command syntax. A usage message for the **userdel** command is displayed.

6 The login to be removed does not exist.

8 The login to be removed is in use.

10 Cannot update the **/etc/group** file but the login is removed from the **/etc/passwd** file.

12 Cannot remove or otherwise modify the home directory.

NOTES

userdel only deletes a user definition that is in the local **/etc/passwd** file. If a network name service such as **NIS** or **NIS+** is being used to supplement the local **/etc/passwd** file with additional entries, **userdel** cannot change information supplied by the network name service.

NAME

usermod – modify a user’s login information on the system

SYNOPSIS

usermod [**–u** *uid* [**–o**]] [**–g** *group*] [**–G** *group* [**,** *group* . . .]] [**–d** *dir* [**–m**]]
[**–s** *shell*] [**–c** *comment*] [**–l** *new_logname*] [**–f** *inactive*] [**–e** *expire*] *login*

DESCRIPTION

The **usermod** utility modifies a user’s login definition on the system. It changes the definition of the specified login and makes the appropriate login-related system file and file system changes.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options may exceed this limit.

login is a string of printable characters that specifies the existing login name of the user. It must exist and may not contain a colon (:) or a newline (\n).

OPTIONS

–u *uid* Specify a new UID for the user. It must be a non-negative decimal integer less than **MAXUID** as defined in **<param.h>**. Note that the UID associated with the user’s home directory is not modified with this option; a user will not have access to their home directory until the UID is manually reassigned using **chown**(1M).

–o This option allows the specified UID to be duplicated (non-unique).

–g *group* Specify an existing group’s integer ID or character-string name. It redefines the user’s primary group membership.

–G *group* Specify an existing group’s integer "ID" "," or character string name. It redefines the user’s supplementary group membership. Duplicates between *group* with the **–g** and **–G** options are ignored. No more than **NGROUPS_UMAX** groups may be specified as defined in **<param.h>**.

–d *dir* Specify the new home directory of the user. It defaults to *base_dir/login*, where *base_dir* is the base directory for new login home directories, and *login* is the new login.

–m Move the user’s home directory to the new directory specified with the **–d** option. If the directory already exists, it must have permissions read / write / execute by *group*, where *group* is the user’s primary group.

–s *shell* Specify the full pathname of the program that is used as the user’s shell on login. The value of *shell* must be a valid executable file.

–c *comment* Specify a comment string. *comment* can be any text string. It is generally a short description of the login, and is currently used as the field for the user’s full name. This information is stored in the user’s **/etc/passwd** entry.

–l *new_logname* Specify a string of printable characters that specifies the new login name for the user. It may not contain a colon (:) or a newline (\n).

–e *expire* Specify the future date on which a login can no longer be used; after this date, no user will be able to access this login. This option is useful for creating temporary logins. You may type the value of the argument *expire* (which is a date) in any format you like (except a Julian date). For example, you may enter **10/6/90** or **October 6, 1990**. A value of " " defeats the status of the expired date.

–f *inactive* Specify the maximum number of days allowed between uses of a login ID before that login ID is declared invalid. Normal values are positive integers. A value of **0** defeats the status.

EXIT CODES

In case of an error, **usermod** prints an error message and exits with one of the following values:

2 The command syntax was invalid. A usage message for the **usermod** command is displayed.

3 An invalid argument was provided to an option.

4 The *uid* given with the **–u** option is already in use.

5 The password files contain an error. **pwconv**(1M) can be used to correct possible errors. See **passwd**(4).

6 The login to be modified does not exist, the *group* does not exist, or the login shell does not exist.

8 The login to be modified is in use.

9 The *new_logname* is already in use.

10 Cannot update the **/etc/group** file. Other update requests will be implemented.

11 Insufficient space to move the home directory (**–m** option). Other update requests will be implemented.

12 Unable to complete the move of the home directory to the new home directory.

FILES

/etc/passwd	system password file
/etc/shadow	system file containing users' encrypted passwords and related information
/etc/group	system file containing group definitions
/etc/datemsk	system file of date formats

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

passwd(1), **users**(1B), **chown**(1M), **groupadd**(1M), **groupdel**(1M), **groupmod**(1M), **logins**(1M), **pwconv**(1M), **useradd**(1M), **userdel**(1M), **passwd**(4), **attributes**(5)

NOTES

The **usermod** utility modifies **passwd** definitions only in the local **/etc/passwd** and **/etc/shadow** files. If a network nameservice such as **NIS** or **NIS+** is being used to supplement the local files with additional entries, **usermod** cannot change information supplied by the network nameservice. However **usermod** will verify the uniqueness of user name and user ID against the external nameservice.

The **usermod** utility uses the **/etc/datemsk** file, available with SUNWaccr, for date formatting.

NAME

utmpd – utmp and utmpx monitoring daemon

SYNOPSIS

utmpd [**–debug**]

DESCRIPTION

The **utmpd** daemon monitors **/var/adm/utmp** and **/var/adm/utmpx** files. See **utmp**(4) and **utmpx**(4).

utmpd receives requests from **pututline**(3C) and **pututxline**(3C) by way of a named pipe. It maintains a table of processes and uses **poll**(2) on **/proc** files to detect process termination. When **utmpd** detects that a process has terminated, it checks that the process has removed its **utmp** entry from **/var/adm/utmp** and **/var/adm/utmpx**. If the process' **utmp** entry has not been removed, **utmpd** removes the entry. By periodically scanning the **/var/adm/utmp** and **/var/adm/utmpx** files, **utmpd** also monitors processes that are not in its table.

OPTIONS

–debug Run in debug mode, leaving the process connected to the controlling terminal. Write debugging information to standard output.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

FILES

/var/adm/utmp	file containing user and accounting information for commands such as **who**(1), **write**(1), and **login**(1)
/var/adm/utmpx	file containing an extended version of the information in **/var/adm/utmp**
/proc	directory containing files for processes whose **utmp** entries are being monitored

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

poll(2), **pututline**(3C), **pututxline**(3C), **proc**(4), **utmp**(4), **utmpx**(4), **attributes**(5)

NAME

uucheck – check the uucp directories and permissions file

SYNOPSIS

/usr/lib/uucp/uucheck [**–v**] [**–x***debug-level*]

DESCRIPTION

uucheck checks for the presence of the **uucp** system required files and directories. **uucheck** also does error checking of the **Permissions** file (**/etc/uucp/Permissions**).

uucheck is executed during package installation. Note that **uucheck** can only be used by the super-user or **uucp**.

OPTIONS

–v Give a detailed (verbose) explanation of how the **uucp** programs will interpret the **Permissions** file.

–x*debug-level* Produce debugging output on the standard output. *debug-level* is a number from **0** to **9**. Higher numbers give more detailed debugging information.

FILES

/etc/uucp/Devices
/etc/uucp/Limits
/etc/uucp/Permissions
/etc/uucp/Systems
/var/spool/locks/∗
/var/spool/uucp/∗
/var/spool/uucppublic/∗

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **uucico**(1M), **uusched**(1M), **attributes**(5)

BUGS

The program does not check file / directory modes or some errors in the **Permissions** file such as duplicate login or machine name.

NAME

uucico – file transport program for the uucp system

SYNOPSIS

/usr/lib/uucp/uucico [–**f**] [–**c***type*] [–**d***spool-directory*] [–**i***interface*] [–**r***role-number*] [–**s***system-name*] [–**x***debug-level*]

DESCRIPTION

uucico is the file transport program for **uucp** work file transfers.

OPTIONS

–**f** This option is used to "force execution" of **uucico** by ignoring the limit on the maximum number of uucicos defined in the **/etc/uucp/Limits** file.

–**c***type* The first field in the **Devices** file is the "Type" field. The –**c** option forces **uucico** to only use entries in the "Type" field that match the user specified *type*. The specified *type* is usually the name of a local area network.

–**d***spool-directory* This option specifies the directory *spool-directory* that contains the **uucp** work files to be transferred. The default spool directory is **/var/spool/uucp**.

–**i***interface* This option defines the *interface* used with **uucico**. The interface only affects slave mode. Known interfaces are UNIX (default), TLI (basic Transport Layer Interface), and TLIS (Transport Layer Interface with Streams modules, read / write).

–**r***role-number* The *role-number* **1** is used for master mode. *role-number* **0** is used for slave mode (default). When **uucico** is started by a program or **cron**, *role-number* **1** should be used for master mode.

–**s***system-name* The –**s** option defines the remote system (*system-name*) that **uucico** will try to contact. It is required when the role is master; *system-name* must be defined in the **Systems** file.

–**x***debug-level* Both **uux** and **uucp** queue jobs that will be transferred by **uucico**. These jobs are normally started by the **uusched** scheduler, for debugging purposes, and can be started manually. For example, the shell **Uutry** starts **uucico** with debugging turned on. The *debug-level* is a number between **0** and **9**. Higher numbers give more detailed debugging information.

FILES

/etc/uucp/Devconfig
/etc/uucp/Devices
/etc/uucp/Limits
/etc/uucp/Permissions
/etc/uucp/Sysfiles
/etc/uucp/Systems
/var/spool/locks/∗
/var/spool/uucp/∗
/var/spool/uucppublic/∗

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **Uutry**(1M), **cron**(1M), **uusched**(1M), **attributes**(5)

NAME

uucleanup – uucp spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [–**C***time*] [–**D***time*] [–**m***string*] [–**o***time*] [–**s***system*] [–**W***time*] [–**x***debug-level*] [–**X***time*]

DESCRIPTION

uucleanup will scan the spool directories for old files and take appropriate action to remove them in a useful way:

- Inform the requester of send / receive requests for systems that can not be reached.
- Return undeliverable mail to the sender.
- Deliver **rnews** files addressed to the local system.
- Remove all other files.

In addition, there is a provision to warn users of requests that have been waiting for a given number of days (default 1 day). Note: **uucleanup** will process as if all option *times* were specified to the default values unless *time* is specifically set.

This program is typically started by the shell **uudemon.cleanup**, which should be started by **cron**(1M).

OPTIONS

–**C***time* Remove any **C.** files greater or equal to *time* days old and send appropriate information to the requester (default 7 days).

–**D***time* Remove any **D.** files greater or equal to *time* days old, make an attempt to deliver mail messages, and execute **rnews** when appropriate (default 7 days).

–**m***string* Include *string* in the warning message generated by the –**W** option. The default line is "See your local administrator to locate the problem".

–**o***time* Delete other files whose age is more than *time* days (default 2 days).

–**s***system* Execute for *system* spool directory only.

–**W***time* Any **C.** files equal to *time* days old will cause a mail message to be sent to the requester warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (–**m** option) (default 1 day).

–**x***debug-level* Produce debugging output on standard ouput. *debug-level* is a single digit between 0 and 9; higher numbers give more detailed debugging information. (This option may not be available on all systems.)

–**X***time* Any **X.** files greater or equal to *time* days old will be removed. The **D.** files are probably not present (if they were, the **X.** could get executed). But if there are **D.** files, they will be taken care of by D. processing (default 2 days).

FILES

/usr/lib/uucp	directory with commands used by **uucleanup** internally
/var/spool/uucp	spool directory

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

SEE ALSO

uucp(1C), **uux**(1C), **cron**(1M), **attributes**(5)

NAME

uusched – the scheduler for the uucp file transport program

SYNOPSIS

/usr/lib/uucp/uusched [–**u***debug-level*] [–**x***debug-level*]

DESCRIPTION

uusched is the **uucp**(1C) file transport scheduler. It is usually started by the daemon *uudemon.hour* that is started by **cron**(1M) from an entry in user **uucp**'s crontab file:

```
11,41 * * * * /etc/uucp/uucp/uudemon.hour
```

OPTIONS

The options are for debugging purposes only. *debug-level* is a number between 0 and 9. Higher numbers give more detailed debugging information:

–**u***debug-level* The –**u***debug-level* option is passed to **uucico**(1M) as –**x***debug-level.*

–**x***debug-level* Outputs debugging messages from **uusched**.

FILES

/etc/uucp/Devices
/etc/uucp/Permissions
/etc/uucp/Systems
/var/spool/locks/*
/var/spool/uucp/*
/var/spool/uucppublic/*

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **cron**(1M), **uucico**(1M), **attributes**(5)

NAME

Uutry, uutry – try to contact remote system with debugging on

SYNOPSIS

/usr/lib/uucp/Uutry [**–r**] [**–c***type*] [**–x***debug-level*] *system-name*

DESCRIPTION

Uutry is a shell script that is used to invoke **uucico**(1M) to call a remote site. Debugging is initially turned on and is set to the default value of **5**. The debugging output is put in file **/tmp/***system-name.*

OPTIONS

–r This option overrides the retry time that is set in file **/var/uucp/.Status/***system-name*.

–c*type* The first field in the **Devices** file is the "Type" field. The **–c** option forces **uucico** to only use entries in the "Type" field that match the user specified *type*. The specified *type* is usually the name of a local area network.

–x*debug-level* *debug-level* is a number from **0** to **9**. Higher numbers give more detailed debugging information.

FILES

/etc/uucp/Devices
/etc/uucp/Limits
/etc/uucp/Permissions
/etc/uucp/Systems
/tmp/*system-name*
/var/spool/locks/*
/var/spool/uucp/*
/var/spool/uucppublic/*

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

SEE ALSO

uucp(1C), **uux**(1C), **uucico**(1M), **attributes**(5)

NAME

uuxqt – execute remote command requests

SYNOPSIS

/usr/lib/uucp/uuxqt [–**s***system*] [–**x***debug-level*]

DESCRIPTION

uuxqt is the program that executes remote job requests from remote systems generated by the use of the **uux** command. (**mail** uses **uux** for remote mail requests). **uuxqt** searches the spool directories looking for execution requests. For each request, **uuxqt** checks to see if all the required data files are available, accessible, and the requested commands are permitted for the requesting system. The **Permissions** file is used to validate file accessibility and command execution permission.

There are two environment variables that are set before the **uuxqt** command is executed:

- **UU_MACHINE** is the machine that sent the job (the previous one).
- **UU_USER** is the user that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

OPTIONS

–**s***system* Specifies the remote *system* name.

–**x***debug-level* *debug-level* is a number from **0** to **9**. Higher numbers give more detailed debugging information.

FILES

/etc/uucp/Limits
/etc/uucp/Permissions
/var/spool/locks/∗
/var/spool/uucp/∗

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

SEE ALSO

mail(1), **uucp**(1C), **uustat**(1C), **uux**(1C), **uucico**(1M), **attributes**(5)

NAME

vmstat – report virtual memory statistics

SYNOPSIS

vmstat [**–cisS**] [*disks*] [*interval* [*count*]]

DESCRIPTION

vmstat delves into the system and reports certain statistics kept about process, virtual memory, disk, trap and CPU activity. Note: **vmstat** statistics are only supported for certain devices.

Without options, **vmstat** displays a one-line summary of the virtual memory activity since the system was booted. If *interval* is specified, **vmstat** summarizes activity over the last *interval* seconds, repeating forever. If a *count* is given, the statistics are repeated *count* times. Note: *interval* and *count* do not apply to the –**i** and –**s** options. If *disks* are specified, they are given priority when **vmstat** chooses which disks to display (only four fit on a line). Common disk names are id, sd, xd, or xy, followed by a number, for example, sd2, xd0, and so forth. For more general system statistics, use **sar**(1), **iostat**(1M), or **sar**(1M).

See *Solaris 1.x to 2.x Transition Guide* for device naming conventions for disks.

OPTIONS

–c Report cache flushing statistics. By default, report the total number of each kind of cache flushed since boot time. The types are: user, context, region, segment, page, and partial-page.

–i Report the number of interrupts per device.

–s Display the total number of various system events since boot.

–S Report on swapping rather than paging activity. This option will change two fields in **vmstat**'s ''paging'' display: rather than the ''re'' and ''mf'' fields, **vmstat** will report ''si'' (swap-ins) and ''so'' (swap-outs).

EXAMPLES

The following command displays a summary of what the system is doing every five seconds.

```
example% vmstat 5
procs    memory            page               disk          faults         cpu
r b w    swap free  re  mf  pi  po  fr  de  sr  s0  s1  s2  s3   in   sy   cs  us  sy  id
0 0 0   11456 4120   1  41  19   1   3   0   2   0   4   0   0   48  112  130   4  14  82
0 0 1   10132 4280   0   4  44   0   0   0   0   0  23   0   0  211  230  144   3  35  62
0 0 1   10132 4616   0   0  20   0   0   0   0   0  19   0   0  150  172  146   3  33  64
0 0 1   10132 5292   0   0   9   0   0   0   0   0  21   0   0  165  105  130   1  21  78
1 1 1   10132 5496   0   0   5   0   0   0   0   0  23   0   0  183   92  134   1  20  79
1 0 1   10132 5564   0   0  25   0   0   0   0   0  18   0   0  131  231  116   4  34  62
1 0 1   10124 5412   0   0  37   0   0   0   0   0  22   0   0  166  179  118   1  33  67
1 0 1   10124 5236   0   0  24   0   0   0   0   0  14   0   0  109  243  113   4  56  39
^C
example%
```

The fields of **vmstat**'s display are:

procs Report the number of processes in each of the three following states:

r in run queue
b blocked for resources (I / O, paging, and so forth)
w runnable but swapped

memory Report on usage of virtual and real memory.

swap amount of swap space currently available (Kbytes)
free size of the free list (Kbytes)

page Report information about page faults and paging activity. The information on each of the following activities is given in units per second.

re page reclaims — but see the –**S** option for how this field is modified.
mf minor faults — but see the –**S** option for how this field is modified.
pi kilobytes paged in
po kilobytes paged out
fr kilobytes freed
de anticipated short-term memory shortfall (Kbytes)
sr pages scanned by clock algorithm

disk Report the number of disk operations per second. There are slots for up to four disks, labeled with a single letter and number. The letter indicates the type of disk (s = SCSI, i = IPI, and so forth); the number is the logical unit number.

faults Report the trap / interrupt rates (per second).

in (non clock) device interrupts
sy system calls
cs CPU context switches

cpu Give a breakdown of percentage usage of CPU time. On MP systems, this is an average across all processors.

us user time
sy system time
id idle time

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

sar(1), **iostat**(1M), **sar**(1M), **attributes**(5)

Solaris 1.x to 2.x Transition Guide
System Administration Guide

NAME

volcopy – make an image copy of file system

SYNOPSIS

volcopy [**–F** *FSType*] [**–V**] [*generic_options*] [**–o** *FSType-specific_options*] *operands*

DESCRIPTION

volcopy makes a literal copy of the file system.

OPTIONS

The following options are supported:

–F *FSType*
Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from **/etc/vfstab** by matching the *operands* with an entry in the table. Otherwise, the default file system type specified in **/etc/default/fs** will be used.

–V
Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from **/etc/vfstab**. This option should be used to verify and validate the command line.

generic_options
Options that are commonly supported by most *FSType*-specific command modules. The following options are available:

–a Require the operator to respond “yes” or “no” instead of simply waiting ten seconds before the copy is made.

–s (Default) Invoke the DEL if wrong verification sequence.

–o *FSType-specific_options*
Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

OPERANDS

The following operand is supported:

operands
generally include the device and volume names and are file system specific. A detailed description of the *operands* can be found on the *FSType*-specific man pages of **volcopy**.

EXIT STATUS

The following exit values are returned:

0	Successful file system copy
1	An error has occurred.

FILES

/etc/vfstab list of default parameters for each file system

/etc/default/fs default local file system type. Default values can be set for the following flags in **/etc/default/fs**. For example: **LOCAL=ufs.**

LOCAL: The default partition for a command if no *FSType* is specified.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

labelit(1M), **vfstab**(4), **attributes**(5)
Manual pages for the *FSType*-specific modules of **volcopy**.

NOTES

This command may not be supported for all *FSTypes*.

NAME

volcopy_ufs – make an image copy of a ufs file system

SYNOPSIS

volcopy [**–F ufs**] [*generic_options*] *fsname srcdevice volname1 destdevice volname2*

DESCRIPTION

volcopy makes a literal copy of the **ufs** file system using a blocksize matched to the device.

OPTIONS

The following option is supported:

generic_options options supported by the generic **volcopy** command. See **volcopy**(1M).

OPERANDS

The following operands are supported:

fsname represents the mount point (for example, **root**, **u1**, etc.) of the file system being copied.

srcdevice or *destdevice* the disk partition specified using the raw device (for example, **/dev/rdsk/cld0s8**, **/dev/rdsk/cld1s8**, etc.).

srcdevice and *volname1* the device and physical volume from which the copy of the file system is being extracted.

destdevice and *volname2* the target device and physical volume.

fsname and *volname* are limited to six or fewer characters and recorded in the superblock. *volname* may be '–' to use the existing volume name.

EXIT STATUS

The following exit values are returned:

0 Successful file system copy.
non-zero An error has occurred.

FILES

/var/adm/filesave.log a record of file systems / volumes copied

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

cpio(1), **dd**(1M), **labelit**(1M), **volcopy**(1M), **fs_ufs**(4), **attributes**(5)

NOTES

volcopy does not support copying to tape devices. Use **dd**(1M) for copying to and from tape devices.

NAME

vold – Volume Management daemon to manage CD-ROM and floppy devices

SYNOPSIS

/usr/sbin/vold [**–n**] [**–t**] [**–v**] [**–f** *config-file*] [**–l** *log-file*] [**–d** *root-dir*]
[**–L** *debug-level*]

DESCRIPTION

The Volume Management daemon, **vold**, creates and maintains a file system image rooted at *root-dir* that contains symbolic names for floppies and CD-ROMs. The default *root-dir* is set to **/vol** if no directory is specified by the **–d** option.

vold reads the **/etc/vold.conf** configuration file upon startup. If the configuration file is modified later, **vold** must be told to reread the **/etc/vold.conf** file. Do this by entering

```
example# kill -HUP vold_pid
```

OPTIONS

The following options are supported:

–n Never writeback. Volume Management updates media labels with unique information if labels are not unique. This flag keeps Volume Management from changing your media. The default setting is FALSE.

–t Dump NFS trace information to the log file. The default setting is FALSE.

–v Provide lots of status information to the log file. The default setting is FALSE (do not provide status info to log file).

–d *root-dir* Specify an alternate root directory. The default location is **/vol**. Setting this will also cause other Volume Management utilities to use this as the default root directory.

–f *config-file* Specify an alternate configuration file. The default file is **/etc/vold.conf**.

–l *log-file* Specify an alternate log file. The default log file is **/var/adm/vold.log**.

–L *debug-level* Change the level (verbosity) of debug messages sent to the log file. The range is **0** to **99** where **0** is nothing and **99** is everything. The default level is **0**.

ENVIRONMENT

vold sets the following environment variables to aid programs which are called when events such as **insert**, **notify**, and **eject** occur:

VOLUME_ACTION	Event that caused this program to be executed.
VOLUME_PATH	Pathname of the matched *regex* from the **vold.conf** file.
VOLUME_DEVICE	Device (in **/vol/dev**) that applies to the media.
VOLUME_NAME	Name of the volume in question.
VOLUME_USER	User ID of the user causing the event to occur.
VOLUME_SYMNAME	Symbolic name of a device containing the volume.
VOLUME_MEDIATYPE	Name of the type of media (CD-ROM or floppy)

FILES

/etc/vold.conf Volume Management daemon configuration file. Directs the Volume Management daemon to control certain devices, and causes events to occur when specific criteria are met.

/usr/lib/vold/*.so.1 Shared objects called by Volume Management daemon when certain actions occur.

/var/adm/vold.log the default log file location (see the **–l** option for a description).

/vol the default Volume Management root directory.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWvolu

SEE ALSO

volcancel(1), **volcheck**(1), **volmissing**(1) **rmmount**(1M), **rmmount.conf**(4), **vold.conf**(4), **attributes**(5), **volfs**(7FS)

System Administration Guide

NAME

wall – write to all users

SYNOPSIS

/usr/sbin/wall [**–a**] [**–g** *grpname*] [*filename*]

DESCRIPTION

wall reads its standard input until an end-of-file. It then sends this message to all currently logged-in users preceded by:

Broadcast Message from . . .

If *filename* is given, then the message is read in from that file. Normally, pseudo-terminals that do not correspond to rlogin sessions are ignored. Thus, when using a window system, the message appears only on the console window. However, **–a** will send the message even to such pseudo-terminals.

It is used to warn all users, typically prior to shutting down the system.

The sender must be super-user to override any protections the users may have invoked (see **mesg**(1)).

wall runs **setgid( )** (see **setuid**(2)) to the group ID **tty**, in order to have write permissions on other user's terminals.

wall will detect non-printable characters before sending them to the user's terminal. Control characters will appear as a " **^** " followed by the appropriate ASCII character; characters with the high-order bit set will appear in "meta" notation. For example, '**\003**' is displayed as '**^C**' and '**\372**' as '**M–z**'.

OPTIONS

–a   broadcast message to the console and pseudo-terminals.

–g *grpname*   broadcast to a specified group only.

ENVIRONMENT

If the **LC_**∗ variables (**LC_CTYPE, LC_TIME, LC_COLLATE, LC_NUMERIC,** and **LC_MONETARY**) (see **environ**(5)) are not set in the environment, the operational behavior of **wall** for each corresponding locale category is determined by the value of the **LANG** environment variable. If **LC_ALL** is set, its contents are used to override both the **LANG** and the other **LC_**∗ variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how **wall** behaves.

FILES

/dev/tty∗

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

mesg(1), **write**(1), **setuid**(2), **attributes**(5), **environ**(5)

NOTES

''Cannot send to ...'' when the open on a user's tty file fails.

NAME

whodo – who is doing what

SYNOPSIS

/usr/sbin/whodo [**–h**] [**–l**] [*user*]

DESCRIPTION

whodo produces formatted and dated output from information in the **/var/adm/utmp**, **/tmp/ps_data**, and **/proc/pid** files.

The display is headed by the date, time, and machine name. For each user logged in, device name, user-ID and login time is shown, followed by a list of active processes associated with the user-ID. The list includes the device name, process-ID, CPU minutes and seconds used, and process name.

If *user* is specified, output is restricted to all sessions pertaining to that user.

OPTIONS

–h Suppress the heading.

–l Produce a long form of output. The fields displayed are: the user's login name, the name of the tty the user is on, the time of day the user logged in (in *hours*:*minutes*), the idle time — that is, the time since the user last typed anything (in *hours*:*minutes*), the CPU time used by all processes and their children on that terminal (in *minutes*:*seconds*), the CPU time used by the currently active processes (in *minutes*:*seconds*), and the name and arguments of the current process.

EXAMPLES

The command:

```
example% whodo
```

produces a display like this:

```
Tue Mar 12 15:48:03 1985
bailey

tty09    mcn      8:51
   tty09   28158    0:29 sh

tty52    bdr     15:23
   tty52   21688    0:05 sh
   tty52   22788    0:01 whodo
   tty52   22017    0:03 vi
   tty52   22549    0:01 sh

xt162    lee     10:20
   tty08    6748   0:01 layers
   xt162    6751   0:01 sh
   xt163    6761   0:05 sh
   tty08    6536   0:05 sh
```

ENVIRONMENT

If any of the **LC_**∗ variables (**LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC,** and **LC_MONETARY**) (see **environ**(5)) are not set in the environment, the operational behavior of **tar**(1) for each corresponding locale category is determined by the value of the **LANG** environment variable. If **LC_ALL** is set, its contents are used to override both the **LANG** and the other **LC_**∗ variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how **tar** behaves.

LC_CTYPE
Determines how **tar** handles characters. When **LC_CTYPE** is set to a valid value, **tar** can display and handle text and filenames containing valid characters for that locale. **tar** can display and handle Extended Unix code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. **tar** can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES
Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

LC_TIME
Determines how **tar** handles date and time formats. In the "C" locale, date and time handling follow the U.S. rules.

FILES

/etc/passwd
/tmp/ps_data
/var/adm/utmp
/proc/pid

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ps(1), **tar**(1), **who**(1), **attributes**(5), **environ**(5)

DIAGNOSTICS

If the PROC driver is not installed or configured or if **/proc** is not mounted, a message to that effect is issued and **whodo** will fail.

The exit status is zero on success, non-zero on failure.

NAME

xntpd – Network Time Protocol daemon

SYNOPSIS

/usr/lib/inet/xntpd [–**bdm**] [– *conffile*] [– *authdelay*] [– *driftfile*] [– *keyfile*] [– *logfile*]
[– *pidfile*] [– *broadcastdelay*] [– *statsdir*] [– *trustedkey*] [– *variable*] [– *variable*]

DESCRIPTION

xntpd is a daemon which sets and maintains a Unix system time–of–day in agreement with Internet standard time servers. **xntpd** is a complete implementation of the Network Time Protocol (NTP) version 3 standard, as defined by *RFC 1305*. It also retains compatibility with version 1 and 2 servers as defined by *RFC 1059* and *RFC 1119*, respectively. The computations done in the protocol and clock adjustment code are carried out with high precision and with attention to the details which might introduce systematic bias into the computations. This is done to try to maintain an accuracy suitable for synchronizing with even the most precise external time source.

Ordinarily, **xntpd** reads its configuration from a configuration file at startup time. The default configuration file name is **/etc/inet/ntp.conf**, although this may be overridden from the command line. It is also possible to specify a working, although limited, **xntpd** configuration entirely on the command line, obviating the need for a configuration file. This may be particularly appropriate when **xntpd** is to be configured as a broadcast or multicast client, with all peers being determined by listening to broadcasts at run time. Through the use of the **ntpq**(1M) program, various internal **xntpd** variables can be displayed and configuration options altered while the daemon is running.

The daemon can operate in any of several modes, including symmetric active / passive, client / server and broadcast / multicast. A broadcast / multicast client can automatically discover remote servers, compute one-way delay correction factors and configure itself automatically. This makes it possible to deploy a fleet of workstations without specifying a configuration file or configuration details specific to its environment.

OPTIONS

The following command line arguments are understood by **xntpd** (see **Configuration Commands** for a more complete functional description):

–**a** Run in “authenticate” mode

–**b** Listen for broadcast NTP and sync to this if available

–**c** Specify an alternate configuration file

–**d** Specify debugging mode. This flag may occur multiple times, with each occurrence indicating greater detail of display

–**e** Specify the time (in seconds) it takes to compute the NTP encryption field on this computer

–**f** *driftfile* Specify the location of the drift file

–**k** Specify the location of the file which contains the NTP authentication keys

–**l** *logfile* Specify a log file instead of logging to syslog

–**m** Listen for multicast messages and synchronize to them if available (requires multicast kernel)

−p	Specify the name of the file to record the daemon's process id
−r	Ordinarily, the daemon automatically compensates for the network delay between the broadcast / multicast server and the client; if the calibration procedure fails, use the specified default delay (in seconds)
−s	Specify the directory to be used for creating statistics files
−t *trustedkey*	Add a key number to the trusted key list
−v	Add a system variable
−V	Add a system variable listed by default

USAGE

xntpd's configuration file format is similar to other Unix configuration files. Comments begin with a `#´ character and extend to the end of the line. Blank lines are ignored. Configuration commands consist of an initial keyword followed by a list of arguments, separated by whitespace. Some arguments may be optional. These commands may not be continued over multiple lines. Arguments may be host names, host addresses written in dotted−decimal, integers, floating point numbers (when specifying times in seconds) and text strings.

Configuration Commands

In the following descriptions, optional arguments are delimited by `[ ]´, while alternatives are separated by ` | ´.

The first three commands specify various time servers to be used and / or time services to be provided.

peer *host_address* [**key** #] [**version** #] [**prefer**]

Specifies that the local server is to operate in "symmetric active" mode with the remote server *host_address* named in the command. In this mode, the local server can be synchronized to the remote server. In addition, the remote server can be synchronized by the local server. This is useful in a network of servers where, depending on various failure scenarios, either the local or remote server host may be the better source of time.

The **peer** command, and the **server** and **broadcast** commands that follow, can take the following arguments:

Option	**Description**
key	Indicates that all packets sent to the address are to include authentication fields, encrypted using the specified key number. The range of this number is that of an unsigned 32 bit integer. By default, an encryption field is not included.

version Specifies the version number to be used for outgoing NTP packets. Versions 1, 2, and 3 are the choices; version 3 is the default.

prefer Marks the host as a preferred host. This host will be preferred for synchronization over other comparable hosts.

server *host_address* [**key** #] [**version** #] [**prefer**] [**mode** #] **server**

Specifies that the local server is to operate in "client" mode with the remote server named in the command. In this mode the local server can be synchronized to the remote server, but the remote server can never be synchronized to the local server.

broadcast *host_address* [**key** #] [**version** #] [**ttl** #]

Specifies that the local server is to operate in "broadcast" mode where the local server sends periodic broadcast messages to a client population at the broadcast / multicast address named in the command. Ordinarily, this specification applies only to the local server operating as a transmitter. For operation as a broadcast client, see **broadcastclient** or **multicastclient** commands elsewhere in this document. In broadcast mode the *host_address* is usually the broadcast address on a local network or a multicast address assigned to NTP. The IANA has assigned the network, 224.0.1.1 to NTP. This is presently the only network that should be used.

The following option is used only with the broadcast mode:

Option	**Description**
ttl	Specifies the time-to-live (TTL) to use on multicast packets. Selection of the proper value, which defaults to 127, is something of a black art and must be coordinated with the network administrator(s).

broadcastclient

Directs the local server to listen for broadcast messages on the local network, in order to discover other servers on the same subnet. Upon hearing a broadcast message for the first time, the local server measures the nominal network delay using a brief client / server exchange with the remote server. Then the server enters the "broadcastclient" mode, in which it listens for and synchronizes to succeeding broadcast messages. In order to avoid accidental or malicious disruption in this mode, both the local and remote servers must operate using authentication, with the same trusted key and key identifier.

multicastclient [*IP address* . . .]

Used in the same way as the **broadcastclient** command, but operates using IP multicasting. Support for this command requires the use of authentication. If one or more IP addresses are given, the server joins the respective multicast group(s). If none are given, the IP address assigned to NTP (224.0.1.1) is assumed.

driftfile *filename*

Specifies the name of the file used to record the frequency offset of the local clock oscillator. If the file exists, it is read at startup in order to set the initial frequency offset. Then the file is updated once per hour with the current offset computed by the daemon. If the file does not exist or this command is not given, the initial frequency offset is assumed to be zero. In this case, it may take some hours for the frequency to stabilize and the residual timing errors to subside. The file contains a single floating point value equal to the offset in parts-per-million (ppm). The file is updated by first writing the current drift value into a temporary file and then using **rename**(2) to replace the old version. This implies that **xntpd** must have write permission for the directory the drift file is located in, and that file system links, symbolic or otherwise, should probably be avoided.

enable auth | bclient | pll | monitor | stats [. . .]
disable auth | bclient | pll | monitor | stats [. . .]

Provides a way to enable or disable various server options. To do so, execute a two word command, where the first word is **enable** or **disable** and the second is the flag. Flags not mentioned are unaffected. Flags that can be changed are described below, along with their default values.

Flag	**Default**	**Description**
auth	**disable**	Causes the server to synchronize with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier.
bclient	**disable**	Causes the server to listen for a message from a broadcast or multicast server. After this occurs, an association is automatically instantiated for that server. default for this flag is disable (off).
pll	**enable**	Enables the server to adjust its local clock. If not set, the local clock free-runs at its intrinsic time and frequency offset. This flag is useful in case the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients.
monitor	**disable**	Enables the monitoring facility (see elsewhere).
stats	**enable**	Enables statistics facility filegen (see **Monitoring Commands** below).

Authentication Commands

keys *filename*

Specifies the name of a file which contains the encryption keys and key identifiers used by **xntpd** when operating in authenticated mode. The format of this file is described later in this document.

trustedkey *#* [*#* . . .]

Specifies the encryption key identifiers which are trusted for the purposes of authenticating peers suitable for synchronization. The authentication procedures require that both the local and remote servers share the same key and key identifier, defined to be used for this purpose. However, different keys can be used with different servers. The arguments are 32 bit unsigned integers. Note, however, that key **0** is fixed and globally known. If meaningful authentication is to be performed, the 0 key should not be trusted.

controlkey *#*

Specifies the key identifier to use with the **ntpq**(1M) program, which is useful to diagnose and repair problems that affect **xntpd** operation. The operation of the **ntpq** program and **xntpd** conform to those specified in *RFC 1305*. Requests from a remote **ntpq** program which affect the state of the local server must be authenticated. This requires that both the remote program and local server share a common key and key identifier. The argument to this command is a 32 bit unsigned integer. If no **controlkey** command is included in the configuration file, or if the keys don't match. These requests are ignored.

authdelay *seconds*

Indicates the amount of time it takes to encrypt an NTP authentication field on the local computer. This value is used to correct transmit timestamps when the authentication is used on outgoing packets. The value usually lies somewhere in the range 0.0001 seconds to 0.003 seconds, though it is very dependent on the CPU speed of the host computer.

Access Control Commands

restrict *address* [**mask** *numeric_mask*] [*flag*] [. . .]

xntpd implements a general purpose address–and–mask based restriction list. The list is sorted by IP address and mask, and the list is searched in this order for matches, with the last match found defining the restriction flags associated with the incoming packets. The source address of incoming packets is used for the match, with the 32 bit address being logically and'ed with the mask associated with the restriction entry and then compared with the entry's address (which has also been and'ed with the mask) to look for a match. The "mask" argument defaults to 255.255.255.255, meaning that the "address" is treated as the address of an individual host. A default entry (address 0.0.0.0, mask 0.0.0.0) is always included and, given the sort algorithm, is always the first entry in the list. Note that, while "address" is normally given in dotted–quad format, the text string "default", with no mask option, may be used to indicate the default entry.

In the current implementation, flags always restrict access, i.e., an entry with no flags indicates that free access to the server is to be given. The flags are not orthogonal, in that more restrictive flags often make less restrictive ones redundant. The flags can generally be classed into two categories, those which restrict time service and those which restrict informational queries and attempts to do run time reconfiguration of the server.

One or more of the following flags may be specified:

ignore Ignore all packets from hosts which match this entry. If this flag is specified neither queries nor time server polls will be responded to.

noquery Ignore all NTP mode 7 packets (i.e., information queries and configuration requests) from the source. Time service is not affected.

nomodify Ignore all NTP mode 7 packets which attempt to modify the state of the server (i.e., run time reconfiguration). Queries which return information are permitted.

notrap Decline to provide mode 6 control message trap service to matching hosts. The trap service is a subsystem of the mode 6 control message protocol which is intended for use by remote event logging programs.

lowpriotrap Declare traps set by matching hosts to be low priority. The number of traps a server can maintain is limited. The current limit is 3. Traps are usually assigned on a first come, first served basis, with later trap requestors being denied service. This flag modifies the assignment algorithm by allowing low priority traps to be overridden by later requests for normal priority traps.

noserve Ignore NTP packets whose mode is other than 7. In effect, time service is denied, though queries may still be permitted.

nopeer Provide stateless time service to polling hosts, but do not allocate peer memory resources to these hosts even if they otherwise might be considered useful as future synchronization partners.

notrust Treat these hosts normally in other respects, but never use them as synchronization sources.

limited These hosts are subject to a limitation on number of clients from the same net that will be accepted. Net in this context refers to the IP notion of net (class A, class B, class C, etc.). Only the first *client_limit* hosts that have shown up at the server and that have been active during the last *client_limit_period* seconds are accepted. Requests from other clients from the same net are rejected. Only time request packets are taken into account. “Private”, “control”, and “broadcast” packets are not subject to client limitation and therefore do not contribute to client count. A history of clients is kept using the monitoring capability of **xntpd**. Thus, monitoring is active as long as there is a restriction entry with the **limited** flag. The default value for *client_limit* is 3. The default value for *client_limit_period* is 3600 seconds. Currently both variables are not runtime configurable.

ntpport This is actually a match algorithm modifier, rather than a restriction flag. Its presence causes the restriction entry to be matched only if the source port in the packet is the standard NTP UDP port (123). Both **ntpport** and **non–ntpport** may be specified. The **ntpport** is considered more specific and is sorted later in the list.

Default restriction list entries, with the flags, **ignore**, **ntpport**, for each of the local host’s interface addresses are inserted into the table at startup to prevent the server from attempting to synchronize to its own time. A default entry is also always present, though if it is otherwise unconfigured no flags are associated with the default entry (i.e., everything besides your own NTP server is unrestricted).

The restriction facility was added to allow the current access policies of the time servers running on the NSFnet backbone to be implemented with **xntpd** as well. This facility may be useful for keeping unwanted or broken remote time servers from affecting your own. However, it should not be considered an alternative to the standard NTP authentication facility.

clientlimit *limit*

Sets *client_limit* to *limit*; allows configuration of client limitation policy. This variable defines the number of clients from the same network that are allowed to use the server.

clientperiod period

Sets *client_limit_period*; allows configuration of client limitation policy. This variable specifies the number of seconds after which a client is considered inactive and thus no longer is counted for client limit restriction.

Monitoring Commands

statsdir */directory path/*

Indicates the full path of a directory where statistics files should be created (see below). This keyword allows the (otherwise constant) filegen filename prefix to be modified for file generation sets used for handling statistics logs (see **filegen** statement below).

statistics *name . . .*

Enables writing of statistics records. Currently, three kinds of statistics are supported. Each type is described below by giving its *name*, a sample line of data, and an explanation of each field:

loopstats enables recording of loop filter statistics information. Each update of the local clock outputs a line of the following form to the file generation set named "loopstats":

```
48773  10847.650  0.0001307  17.3478  2
```

Field No.	Description
1	The date (Modified Julian day)
2	The time (seconds and fraction past UTC midnight)
3	Time offset in seconds
4	Frequency offset in parts-per-million
5	Time constant of the clock-discipline algorithm at each update of the clock

peerstats enables recording of peer statistics information. This includes statistics records of all peers of a NTP server and of the 1-pps signal, where present and configured. Each valid update appends a line similar to the one below, to the current element of a file generation set named “peerstats” :

```
48773  10847.650  127.127.4.1  9714  -0.001605  0.00000  0.00142
```

Field No.	**Description**
1	The date (Modified Julian Day)
2	The time (seconds and fraction past UTC midnight)
3	The peer address in dotted-quad notation
4	peer status. The status field is encoded in hex in the format described in Appendix A of the NTP specification, *RFC 1305*.
5	Offset in seconds
6	Delay in seconds
7	Dispersion in seconds

clockstats

enables recording of clock driver statistics information. Each update received from a clock driver outputs a line of the following form to the file generation set named “clockstats” :

```
49213  525.624  127.127.4.1  93  226  00:08:29.606  D
```

Field No.	**Description**
1	The date (Modified Julian Day)
2	The time (seconds and fraction past UTC midnight)
3	The clock address in dotted-quad notation
4	The last timecode received from the clock in decoded ASCII format, where meaningful

In some clock drivers a good deal of additional information can be gathered and displayed as well.

Statistic files are managed using file generation sets (see **filegen** below). The information obtained by enabling statistics recording allows analysis of temporal properties of a **xntpd** server. It is usually only useful to primary servers or maybe main campus servers.

filegen *name* [**file** *filename*] [**type** *typename*] [**flag** *flagval*] [**link** | **nolink**]
[enable **|** disable **]**

Configures setting of generation file set *name*. Generation file sets provide a means for handling files that are continuously growing during the lifetime of a server. Server statistics are a typical example for such files. Generation file sets provide access to a set of files used to store the actual data. At any time at most one element of the set is being written to. The **type** given specifies when and how data will be directed to a new element of the set. This way, information stored in elements of a file set that are currently unused are available for administrational operations without the risk of disturbing the operation of **xntpd**. (Most important: they can be removed to free space for new data produced.)

Filenames of set members are built from three elements:

prefix This is a constant filename path. It is not subject to modifications via the **filegen** statement. It is defined by the server, usually specified as a compile time constant. It may, however, be configurable for individual file generation sets via other commands. For example, the prefix used with “loopstats” and “peerstats” filegens can be configured using the **statsdir** statement explained above.

filename This string is directly concatenated to the *prefix* mentioned above (no intervening `/´ (slash)). This can be modified using the **file** argument to the **filegen** statement. No `..´ elements are allowed in this component to prevent filenames referring to parts outside the filesystem hierarchy denoted by *prefix*.

suffix This part is reflects individual elements of a file set. It is generated according to the *type* of a file set as explained below.

A file generation set is characterized by its type. The following types are supported:

none The file set is actually a single plain file.

pid One element of file set is used per incarnation of a **xntpd** server. This type does not perform any changes to file set members during runtime. However it provides an easy way of separating files belonging to different **xntpd** server incarnations. The set member filename is built by appending a `.´ (dot) to concatenated *prefix* and *filename* strings, and appending the decimal representation of the process id of the **xntpd** server process.

day One file generation set element is created per day. The term *day* is based on UTC. A day is defined as the period between 00:00 and 24:00 UTC. The file set member suffix consists of a `.´ (dot) and a day specification in the form, *YYYYMMDD*. *YYYY* is a 4 digit year number (e.g., 1992). *MM* is a two digit month number. *DD* is a two digit day number. Thus, all information written at December 10th, 1992 would end up in a file named, *PrefixFilename*.19921210.

week Any file set member contains data related to a certain week of a year. The term *week* is defined by computing “day of year” modulo 7. Elements of such a file generation set are distinguished by appending the following suffix to the file set filename base: a dot, a four digit year number, the letter `W´, and a two digit week number. For example, information from January, 5th 1992 would end up in a file with suffix “.1992W1”.

month One generation file set element is generated per month. The file name suffix consists of a dot, a four digit year number, and a two digit month.

year One generation file elment is generated per year. The filename suffix consists of a dot and a 4 digit year number.

age This type of file generation sets changes to a new element of the file set every 24 hours of server operation. The filename suffix consists of a dot, the letter `a´, and an eight digit number. This number is taken to be the number of seconds the server is running at the start of the corresponding 24 hour period.

Information is only written to a file generation set when this set is **enabled**. Output is prevented by specifying, **disabled**.

It is convenient to be able to access the current element of a file generation set by a fixed name. This feature is enabled by specifying **link** and disabled using **nolink**. If **link** is specified, a hard link from the current file set element to a file without suffix is created. When there is already a file with this name and the number of links of this file is one, it is renamed appending a dot, the letter, `C´, and the pid of the **xntpd** server process. When the number of links is greater than one, the file is unlinked. This allows the current file to be accessed by a constant name.

Miscellaneous Commands

precision #
Specifies the nominal precision of the local clock. The value of, # is an integer approximately equal to the base 2 logarithm of the local timekeeping precision in seconds. Normally, the daemon determines the precision automatically at startup. So this command is necessary only in special cases when the precision cannot be determined automatically.

broadcastdelay *seconds*
The broadcast and multicast modes require a special calibration to determine the network delay between the local and remote servers. Ordinarily, this is done automatically by the initial protocol exchanges between the local and remote servers. In some cases, the calibration procedure may fail due to, for example, network or server access controls. This command specifies the default delay to be used under these circumstances. Typically (for Ethernet), a number between 0.003 and 0.007 is appropriate for *seconds*. When this command is not used, the default is 0.004 seconds.

trap *host_address* [**port** *port_number*] [**interface** *interface_address*]
Configures a trap receiver at the given *host_address* and *port_number* for sending messages with the specified local *interface_address.* If the port number is unspecified, a value of 18447 is used. If the interface address is not specified, the message is sent with the source address of the local interface the message is sent through. On a multi-homed host, the interface used may change with routing changes.

The trap receiver generally logs event messages and other information from the server in a log file. While such monitor programs may also request their own trap dynamically, configuring a trap receiver ensures that no messages are lost when the server is started.

setvar *variable* [*default*]

This command adds an additional system variable. Variables like this can be used to distribute additional information such as the access policy. If the variable of the form, *variable_name=value* is followed by the *default* keyword, the variable will be listed as one of the default system variables (see the **ntpq**(1M) command). Additional variables serve informational purposes only. They can be listed; but they are not related to the protocol. The known protocol variables always override any variables defined via the **setvar** mechanism.

Three special variables contain the names of all variable of the same group. *sys_var_list* holds the names of all system variables. *peer_var_list* holds the names of all peer variables. And *clock_var_list* hold the names of the reference clock variables.

monitor [**yes | no**]
authenticate [**yes | no**]

These commands have been superseded by the **enable** and **disable** commands. They are listed here for historical purposes.

logconfig *configkeyword*

Controls the amount of output written to syslog or the logfile. By default all output is turned on. *configkeyword* is formed by concatenating the message class with the event class. It is permissible to use the prefix, **all**, instead of a message class. A message class may also be followed by the keyword, **all**, meaning to enable / disable all of the respective message class. All *configkeywords* can be prefixed with the symbols, \`=´, \`+´ and \`−´ . Here, \`=´ sets the syslogmask, \`+´ adds messages, and \`−´ removes messages. Syslog messages can be controlled in four classes: *sys*, *peer*, *clock*, *sync*. Within these classes four types of messages can be controlled. Each is described below, along with its **configkeyword**:

Configkeyword	**Message type**
info	Informational messages control configuration information.
events	Event messages control logging of events (reachability, synchronization, alarm conditions).
statistics	Statistical messages control statistical output.
status	Status messages describe mainly the synchronization status.

Examples:

A minimal log configuration might look like this:

```
logconfig =syncstatus +sysevents
```

A configuration like this lists, just the synchronization state of **xntp** and the major system events. For a simple reference server, the following minimum message configuration could be useful:

```
logconfig =syncall +clockall
```

This configuration lists all clock information and synchronization information. All other events and messages about peers, system events and so on, is suppressed.

Authentication Key File Format

The NTP standard specifies an extension to allow verification of the authenticity of received NTP packets, and to provide an indication of authenticity in outgoing packets. This is implemented in **xntpd** using the DES or MD5 algorithms to compute a digital signature, or message-digest. The specification allows any one of possibly 4 billion keys, numbered with 32 bit key identifiers, to be used to authenticate an association. The servers involved in an association must agree on the key and key identifier used to authenticate their data. However they must each learn the key and key identifier independently. In the case of DES, the keys are 56 bits long with, depending on type, a parity check on each byte. In the case of MD5, the keys are 64 bits (8 bytes). **xntpd** reads its keys from a file specified using the –**k** command line option or the **keys** statement in the configuration file. While key number 0 is fixed by the NTP standard (as 56 zero bits) and may not be changed, one or more of the keys numbered 1 through 15 may be arbitrarily set in the keys file.

The key file uses the same comment conventions as the configuration file. Key entries use a fixed format of the form, *keyno type key*. Here, *keyno* is a positive integer, *type* is a single character which defines the format the key is given in, and *key* is the key itself.

The *key* may be given in one of several different formats, controlled by the *type* character. The different key types, and corresponding formats, are described below:

Key: S
Format: A 64 bit hexadecimal number in DES format

In this format, the high order 7 bits of each octet are used to form the 56 bit key while the low order bit of each octet is given a value such that odd parity is maintained for the octet. Leading zeroes must be specified (i.e., the key must be exactly 16 hex digits long) and odd parity must be maintained. Hence a zero key, in standard format, would be given as: 0101010101010101.

Key: N
Format: A 64 bit hexadecimal number in NTP format

This format is the same as the DES format except the bits in each octet have been rotated one bit right so that the parity bit is now the high order bit of the octet. Leading zeroes must be specified and odd parity must be maintained. A zero key in NTP format would be specified as: 8080808080808080.

Key: A
Format: A 1–to–8 character ASCII string

A key is formed from this by using the lower order 7 bits of the ASCII representation of each character in the string. Zeroes are added on the right when necessary to form a full width 56 bit key.

Key: S
Format: A 1–to–8 character ASCII string, using the MD5 authentication scheme.

Note that both the keys and the authentication schemes (DES or MD5) must be identical between a set of peers sharing the same key number.

Primary Clock Support

xntpd has been built to be compatible with all supported types of reference clocks. A reference clock is generally (though not always) a radio timecode receiver which is synchronized to a source of standard time such as the services offered by the NRC in Canada and NIST in the U.S. The interface between the computer and the timecode receiver is device dependent and will vary, but it is often a serial port.

For the purposes of configuration, **xntpd** treats reference clocks in a manner analogous to normal NTP peers as much as possible. Reference clocks are referred to by address, much as a normal peer is. However, an invalid IP address is used to distinguish them from normal peers. Reference clock addresses are of the form *127.127.t.u* where *t* is an integer denoting the clock type and *u* indicates the type–specific unit number. Reference clocks are configured using a **server** statement in the configuration file where the *host_address* is the clock address. The **key**, **version** and **ttl** options are not used for reference clock support. Some reference clocks require a **mode** option to further specify their operation. The **prefer** option can be useful to persuade the server to cherish a reference clock with somewhat more enthusiasm than other reference clocks or peers. Clock addresses may generally be used anywhere in the configuration file that a normal IP address can be used. For example, they can be used in **restrict** statements, although such use would normally be considered strange.

Reference clock support provides the **fudge** command, which can be used to configure reference clocks in special ways. The generic format that applies to this command is,

fudge *127.127.t.u* [**time1** *secs*] [**time2** *secs*] [**stratum** *int*] [**refid** *int*] [**flag1** *0* | *1*]
[**flag2** *0* | *1*] [**flag3** *0* | *1*] [**flag4** *0* | *1*]

with options described as follows:

time1
time2 Are specified in fixed point seconds and used in some clock drivers as calibration constants. By convention, and unless indicated otherwise, **time1** is used as a calibration constant to adjust the nominal time offset of a particular clock to agree with an external standard, such as a precision PPS signal. The specified offset is in addition to the propagation delay provided by other means, such as internal DIP switches.

stratum Is a number in the range zero to 15 and is used to assign a nonstandard operating stratum to the clock.

refid Is an ASCII string in the range one to four characters and is used to assign a nonstandard reference identifier to the clock.

flag1
flag2
flag3
flag4 Are binary flags used for customizing the clock driver. The interpretation of these values, and whether they are used at all, is a function of the needs of the particular clock driver. However, by convention, and unless indicated otherwise, *flag3* is used to attach the ppsclock streams module to the configured driver, while *flag4* is used to enable recording verbose monitoring data to the clockstats file configured with the **filegen** command. Further information on the ppsclock streams module is in the **README** file in the **./kernel** directory in the current xntp3 program distribution. Further information on this feature is available in the **./scripts/stats** directory in the same distribution.

Ordinarily, the stratum of a reference clock is zero, by default. Since the **xntpd** daemon adds one to the stratum of each peer, a primary server ordinarily displays stratum one. In order to provide engineered backups, it is often useful to specify the reference clock stratum as greater than zero. The **stratum** option is used for this purpose. Also, in cases involving both a reference clock and a 1-pps discipline signal, it is useful to specify the reference clock identifier as other than the default, depending on the driver. The *refid* option is used for this purpose. Except where noted, these options apply to all clock drivers.

xntpd on Unix machines currently supports several different types of clock hardware. It also supports a special pseudo–clock used for backup or when no other clock source is available. In the case of most of the clock drivers, support for a 1-pps precision timing signal is available as described in the **README** file in the **./doc directory of the xntp3** program distribution. The clock drivers, and the addresses used to configure them, are described in the file, **README.refclocks**, in the doc directory of the current program distribution.

Variables

Most variables used by the NTP protocol can be examined with **ntpq** (mode 6 messages). Currently very few variables can be modified via mode 6 messages. These variables are either created with the **setvar** directive or the leap warning variables. The leap warning bits that can be set in the *leapwarning* variable (up to one month ahead). Both, the *leapwarning* and in the *leapindication* variable, have a slightly different encoding than the usual **leap** bits interpretation:

00 The daemon passes the leap bits of its synchronization source (usual mode of operation).
01/10 A leap second is added / deleted (operator forced leap second).
11 Leap information from the synchronization source is ignored (thus LEAP_NOWARNING is passed on).

FILES

/etc/inet/ntp.conf the default name of the configuration file

/etc/inet/ntp.drift the conventional name of the drift file

/etc/inet/ntp.keys the conventional name of the key file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

ntpdate(1M), **ntpq**(1M), **rename**(2), **attributes**(5)

NOTES

While not huge, **xntpd** it has gotten larger than might be desirable for an elevated−priority daemon running on a workstation. This is particularly true, since many of the fancy features which consume the space were designed with a busy primary server in mind, rather than a high stratum workstation.

NAME

ypbind – NIS binder process

SYNOPSIS

/usr/lib/netsvc/yp/ypbind [**–broadcast** | **–ypset** | **–ypsetme**]

DESCRIPTION

NIS provides a simple network lookup service consisting of databases and processes. The databases are stored at the machine that runs an NIS server process. The programmatic interface to NIS is described in **ypclnt**(3N). Administrative tools are described in **ypinit**(1M), **ypwhich**(1), and **ypset**(1M). Tools to see the contents of NIS maps are described in **ypcat**(1), and **ypmatch**(1).

ypbind is a daemon process that is activated at system startup time from the startup script **/etc/init.d/rpc**. By default, it is invoked as **ypbind –broadcast**. **ypbind** runs on all client machines that are set up to use NIS, see **sysidtool**(1M).

The function of **ypbind** is to remember information that lets all NIS client processes on a node communicate with some NIS server process. **ypbind** must run on every machine which has NIS client processes. The NIS server may or may not be running on the same node, but must be running somewhere on the network. The NIS server is **not supported** in SunOS releases anymore, but **ypbind** can communicate with an NIS server, **ypserv,** on an earlier SunOS release or an NIS+ server in "YP-compatibility mode", see **rpc.nisd**(1M). Refer to the NOTES section in **ypfiles**(4) for implications of being served by such an NIS+ server.

The information **ypbind** remembers is called a *binding* — the association of a domain name with a NIS server.

The process of binding is driven by client requests. As a request for an unbound domain comes in, if started with the **–broadcast** option, the **ypbind** process broadcasts on the net trying to find an NIS server, either a **ypserv** process serving the domain or an **rpc.nisd** process in "YP-compatibility mode" serving NIS+ directory with name the same as (case sensitive) the domain in the client request. Since the binding is established by broadcasting, there must be at least one NIS server on the net.

If started without the **–broadcast** option, **ypbind** process steps through the list of NIS servers that was created by **ypinit –c** for the requested domain. There must be an NIS server process on at least one of the hosts in the NIS servers file. All the hosts in the NIS servers file **must be listed** in /etc/hosts along with their IP addresses. Once a domain is bound by **ypbind**, that same binding is given to every client process on the node. The **ypbind** process on the local node or a remote node may be queried for the binding of a particular domain by using the **ypwhich**(1) command.

If **ypbind** is unable to speak to the NIS server process it is bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain will wait until the requested domain is bound. In general, a bound domain is marked as unbound when the node running the NIS server crashes or gets overloaded. In such a case, **ypbind** will try to bind to another NIS server using the process described above.

ypbind also accepts requests to set its binding for a particular domain. The request is usually generated by the **ypset**(1M) command. In order for **ypset** to work, **ypbind** must have been invoked with flags −**ypset** or −**ypsetme**.

OPTIONS

−**broadcast** Send a broadcast datagram using UDP / IP that requests the information needed to bind to a specific NIS server. This option is analogous to ypbind with no options in earlier Sun releases and is recommended for ease of use.

−**ypset** Allow users from any remote machine to change the binding via the **ypset** command. By default, no one can change the binding. This option is really insecure.

−**ypsetme** Only allow **root** on the local machine to change the binding to a desired server via the **ypset** command. **ypbind** can verify the caller is indeed a **root** user by accepting such requests only on the loopback transport. By default, no external process can change the binding.

FILES

/var/yp/binding/*ypdomain***/ypservers**
/etc/hosts

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

ypcat(1), **ypmatch**(1), **ypwhich**(1), **ifconfig**(1M), **rpc.nisd**(1M), **ypinit**(1M), **ypset**(1M), **ypclnt**(3N), **hosts**(4), **ypfiles**(4), **attributes**(5)

NOTES

ypbind supports multiple domains. The **ypbind** process can maintain bindings to several domains and their servers, the default domain is the one specified by the **domainname**(1M) command at startup time.

The −**broadcast** option works only on the UDP transport. It is insecure since it trusts "any" machine on the net that responds to the broadcast request and poses itself as an NIS server.

NAME

ypinit – set up NIS client

SYNOPSIS

/usr/sbin/ypinit [–**c**] [–**m**] [–**s** *master_server*]

DESCRIPTION

ypinit can be used to set up an NIS client system. You must be the super-user to run this command. This script need not be used at all if **ypbind**(1M) is started with the –**broadcast** option (it is invoked with this option from the start up script **/etc/init.d/rpc**).

Normally, **ypinit** is run only once after installing the system. It may be run whenever a new NIS server is added to the network or an existing one is decommissioned.

ypinit prompts for a list of NIS servers to bind the client to; this list should be ordered from the closest to the furthest server. Each of these NIS servers *must* be listed in **/etc/hosts** along with its IP address. **ypinit** stores the list in file **/var/yp/binding/***domain***/ypservers**. This file is used by **ypbind** when run without the –**broadcast** option.

OPTIONS

–**c** Set up a ypclient system.

–**m** Build a master ypserver data base.

–**s** *master_server* Slave data base. *master_server* must be the same master configured in the YP maps and returned by the **ypwhich -m** command.

FILES

/etc/hosts
/var/yp/binding/*domain***/ypservers**

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

ypbind(1M), **sysinfo**(2), **hosts**(4), **attributes**(5)

BUGS

ypinit sets up the list of NIS servers only for the current domain on the system when it is run, that is, the domain returned by the **SI_SRPC_DOMAIN** command to **sysinfo**(2). Care should be taken to ensure that this is the same as the desired domain for NIS client processes.

NAME

ypmake – rebuild NIS database

SYNOPSIS

cd /var/yp ; make [*map*]

DESCRIPTION

The file called **Makefile** in **/var/yp** is used by **make**(1) to build the Network Information Service (NIS) database. With no arguments, **make** creates **dbm** databases for any NIS maps that are out-of-date, and then executes **yppush**(1M) to notify slave databases that there has been a change.

If you supply a *map* on the command line, **make** will update that map only. Typing **make passwd** will create and **yppush** the password database (assuming it is out of date). Likewise, **make hosts** and **make networks** will create and **yppush** the host and network files, **/etc/hosts** and **/etc/networks**.

There are three special variables used by **make**: DIR, which gives the directory of the source files; NOPUSH, which when non-null inhibits doing a **yppush** of the new database files; and DOM, used to construct a domain other than the master's default domain. The default for DIR is **/etc**, and the default for NOPUSH is the null string.

Refer to **ypfiles**(4) and **ypserv**(1M) for an overview of the NIS service.

FILES

/var/yp	Directory containing NIS configuration files.
/etc/hosts	System hosts file.
/etc/networks	System networks file.

SEE ALSO

make(1), **nis+**(1), **makedbm**(1M), **rpc.nisd**(1M), **ypbind**(1M), **yppush**(1M), **ypserv**(1M), **ypclnt**(3N), **ypfiles**(4)

NOTES

The NIS makefile is only used when running the **ypserv**(1M) server to provide NIS services. If these are being provided by the NIS+ server running in NIS compatibility mode, see **rpc.nisd**(1M); this makefile is not relevant. See **ypfiles**(4) for more details.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppoll – return current version of a NIS map at a NIS server host

SYNOPSIS

/usr/sbin/yppoll [**–d** *ypdomain*] [**–h** *host*] *mapname*

DESCRIPTION

The **yppoll** command asks a **ypserv**( ) process what the order number is, and which host is the master NIS server for the named map.

OPTIONS

–d *ypdomain* Use *ypdomain* instead of the default domain.

–h *host* Ask the **ypserv** process at *host* about the map parameters. If *host* is not specified, the NIS server for the local host is used. That is, the default host is the one returned by **ypwhich**(1).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

ypwhich(1), **ypfiles**(4), **attributes**(5)

NAME

yppush – force propagation of changed NIS map

SYNOPSIS

/usr/etc/yp/yppush [**–v**] [**–h** *host*] [**–d** *domain*] [**–p** *#parallel-xfrs*] *mapname*

DESCRIPTION

yppush copies a new version of a Network Information Service (NIS) map from the master NIS server to the slave NIS servers. It is normally run only on the master NIS server by the **Makefile** in **/var/yp** after the master databases are changed. It first constructs a list of NIS server hosts by reading the NIS **ypservers** map within the *domain*. Keys within the **ypservers** map are the ASCII names of the machines on which the NIS servers run.

A “transfer map” request is sent to the NIS server at each host, along with the information needed by the transfer agent (the program which actually moves the map) to call back the **yppush**. When the attempt has completed (successfully or not), and the transfer agent has sent **yppush** a status message, the results may be printed to stdout. Messages are also printed when a transfer is not possible; for instance when the request message is undeliverable, or when the timeout period on responses has expired.

Refer to **ypfiles**(4) and **ypserv**(1M) for an overview of the NIS service.

OPTIONS

–d *domain* Specify a *domain*.

–h *host* Propagate only to the named *host*.

–p *#parallel-xfrs* Allow the specified number of map transfers to occur in parallel.

–v Verbose. This prints messages when each server is called, and for each response. If this flag is omitted, only error messages are printed.

FILES

/var/yp Directory where NIS configuration files reside.

/var/yp/*domain***/ypservers.{***dir***,** *pag***}** Map containing list of NIS servers to bind to when running in server mode.

SEE ALSO

ypserv(1M), **ypxfr**(1M), **ypfiles**(4)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

BUGS

In the current implementation (version 2 NIS protocol), the transfer agent is **ypxfr**(1M), which is started by the **ypserv** program. If **yppush** detects that it is speaking to a version 1 NIS protocol server, it uses the older protocol, sending a version 1 **YPPROC_GET** request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for version 1 servers. **yppush** prints a message saying that an “old-style” message has been sent. The system administrator should later check to see that the transfer has actually taken place.

NAME

ypserv, ypxfrd – NIS server and binder processes

SYNOPSIS

/usr/lib/netsvc/yp/ypserv [**–dv**]

/usr/lib/netsvc/yp/ypxfrd

DESCRIPTION

The Network Information Service (NIS) provides a simple network lookup service consisting of databases and processes. The databases are **ndbm** files (see **dbm_clearerr**(3)) in a directory tree rooted at **/var/yp**. These files are described in **ypfiles**(4). The processes are **/usr/lib/netsvc/yp/ypserv**, the NIS database lookup server, and **/usr/lib/netsvc/yp/ypbind**, the NIS binder. The programmatic interface to the NIS service is described in **ypclnt**(3N). Administrative tools are described in **yppoll**(1M), **yppush**(1M), **ypset**(1M), **ypxfr**(1M), and **ypwhich**(1M). Tools to see the contents of NIS maps are described in **ypcat**(1), and **ypmatch**(1). Database generation and maintenance tools are described in **ypinit**(1M), **ypmake**(1M), and **makedbm**(1M).

The **ypserv** utility is a daemon process typically activated at system startup time from **/etc/init.d/rpc**. Alternatively, NIS services can also be started using **ypstart**(1M) from the command-line as the root user. **ypserv** runs only on NIS server machines with a complete NIS database. All NIS services can be halted using the **ypstop**(1M) command.

The **ypxfrd** utility transfers entire NIS maps in an efficient manner. For systems that use this daemon, map transfers are 10 to 100 times faster, depending on the map. To use this daemon, **ypxfrd** should be run on the master server. See **/usr/lib/netsvc/yp/ypstart**. **ypxfr** attempts to use **ypxfrd** first, if that fails, it prints a warning and then uses the older transfer method.

The **ypserv** daemon's primary function is to look up information in its local database of NIS maps.

The operations performed by **ypserv** are defined for the implementor by the *YP Protocol Specification*, and for the programmer by the header file **rpcsvc/yp_prot.h**.

Communication to and from **ypserv** is by means of RPC calls. Lookup functions are described in **ypclnt**(3N), and are supplied as C-callable functions in the **libnsl**(4) library. There are four lookup functions, all of which are performed on a specified map within some NIS domain: **match**, **get_first**, **get_next**, and **get_all**. The **match** operation takes a key, and returns the associated value. The **get_first** operation returns the first key-value pair from the map, and **get_next** can be used to enumerate the remainder. **get_all** ships the entire map to the requester as the response to a single RPC request.

There are a number of special keys in the DBM files that can alter the way in which **ypserv** operates. The keys of interest are:

YP_INTERDOMAIN
The presence of this key causes **ypserv** to forward host lookups that cannot be satisfied by the DBM files to a DNS server.

YP_SECURE
This key causes **ypserv** to only answer questions coming from clients on reserved ports.

YP_MULTI_*hostname*
This is a special key in the form, **YP_MULTI_***hostname addr1,...,addrN*. A client looking for *hostname* has the “closest“ address returned.

Two other functions supply information about the map, rather than map entries: **get_order_number**, and **get_master_name**. In fact, both order number and master name exist in the map as key-value pairs, but the server will not return either through the normal lookup functions. If you examine the map with **makedbm**(1M), however, they are visible. Other functions are used within the NIS service subsystem itself, and are not of general interest to NIS clients. They include **do_you_serve_this_domain?**, **transfer_map**, and **reinitialize_internal_state**.

OPTIONS

ypserv

−d The NIS service should go to the DNS (Domain Name Service) for more host information. This requires the existence of a correct **/etc/resolv.conf** file pointing at a machine running **in.named**(1M). This option supersedes the non-existence of the **YP_INTERDOMAIN** flag in the hosts maps. In the absence of an **/etc/resolv.conf** file, **ypserv** complains, but ignores the −**d** option.

−v Operate in the verbose mode, printing diagnostic messages to stderr.

FILES

/var/yp/securenets Defines the hosts and networks which are granted access to information in the served domain; it is read at startup time by both **ypserv** and **ypxfrd**.

/etc/init.d/rpc Startup file that starts up basic RPC services, and NIS by calling **ypstart**(1M)**.**

If the **/var/yp/ypserv.log** file exists when **ypserv** starts up, log information will be written to it when error conditions arise.

The file **/var/yp/binding/***domainname***/ypservers** is used to list the NIS server hosts that **ypbind** will bind to.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

SEE ALSO

ypcat(1), **ypmatch**(1), **ypwhich**(1), **domainname**(1M), **in.named**(1M), **makedbm**(1M), **ypbind**(1M), **ypinit**(1M), **ypmake**(1M), **yppoll**(1M), **yppush**(1M), **ypset**(1M), **ypstart**(1M), **ypstop**(1M), **ypxfr**(1M), **dbm_clearerr**(3), **ypclnt**(3N), **securenets**(4), **ypfiles**(4), **attributes**(5)

Network Interfaces Programmer’s Guide
System Administration Guide

NOTES

ypserv supports multiple domains. The **ypserv** process determines the domains it serves by looking for directories of the same name in the directory **/var/yp**. It replies to all broadcasts requesting yp service for that domain.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypset – point ypbind at a particular server

SYNOPSIS

/usr/sbin/ypset [**–d** *ypdomain*] [**–h** *host*] *server*

DESCRIPTION

In order to run **ypset**, **ypbind** must be initiated with the **–ypset** or **–ypsetme** options. See **ypbind**(1M). **ypset** tells **ypbind** to get NIS services for the specified *ypdomain* from the **ypserv** process running on *server*. If *server* is down, or is not running **ypserv**, this may not be discovered until an NIS client process tries to get a binding for the domain. At this point, the binding set by **ypset** will be tested by **ypbind**. If the binding is invalid, **ypbind** will attempt to rebind for the same domain.

ypset is useful for binding a client node which is not on a broadcast net, or is on a broadcast net which is not running a NIS server host. It also is useful for debugging NIS client applications, for instance where a NIS map only exists at a single NIS server host.

In cases where several hosts on the local net are supplying NIS services, it is possible for **ypbind** to rebind to another host even while you attempt to find out if the **ypset** operation succeeded. For example, you can type:

```
example% ypset host1
example% ypwhich
host2
```

which can be confusing. This is a function of the NIS subsystem's attempt to load-balance among the available NIS servers, and occurs when **host1** does not respond to **ypbind** because it is not running **ypserv** (or is overloaded), and **host2**, running **ypserv**, gets the binding.

server indicates the NIS server to bind to, and must be specified as a name or an IP address. This will work only if the node has a current valid binding for the domain in question, and **ypbind** has been set to allow use of **ypset**. In most cases, *server* should be specified as an IP address.

ypset tries to bind over a connectionless transport. The NIS library call, **yp_all( )**, uses connection-oriented transport and derives the NIS server's address based on the connectionless address supplied by **ypset**.

Refer to **ypfiles**(4) for an overview of the NIS name service.

OPTIONS

–d *ypdomain* Use *ypdomain*, instead of the default domain.

–h *host* Set **ypbind**'s binding on *host,* instead of locally. *host* must be specified as a name.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

ypwhich(1), **ypfiles**(4), **attributes**(5)

NAME

ypstart, ypstop – Start and stop NIS services

SYNOPSIS

/usr/lib/netsvc/yp/ypstart

/usr/lib/netsvc/yp/ypstop

DESCRIPTION

The **ypstart** command is used to start the Network Information Service (NIS). Once the host has been configured using the **ypinit**(1M) command, **ypstart** automatically determines the NIS status of the machine and starts the appropriate daemons.

The **ypstop** command is used to stop the Network Information Service (NIS).

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

SEE ALSO

ypinit(1M), **attributes**(5)

System Administration Guide
Network Interfaces Programmer's Guide

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypxfr, ypxfr_1perday, ypxfr_1perhour, ypxfr_2perday – transfer NIS map from a NIS server to host

SYNOPSIS

/usr/lib/netsvc/yp/ypxfr [**–c**] [**–f**] [**–C** *tid prog server*] [**–d** *ypdomain*] [**–h** *host*] [**–s** *ypdomain*] *mapname*

DESCRIPTION

The **ypxfr** command moves a NIS map in the default domain for the local host to the local host by making use of normal NIS services. It creates a temporary map in the directory **/var/yp/***ypdomain* (this directory must already exist; *ypdomain* is the default domain for the local host), fills it by enumerating the map's entries, fetches the map parameters (master and order number), and loads them. It then deletes any old versions of the map and moves the temporary map to the real *name*.

If run interactively, **ypxfr** writes its output to the terminal. However, if it is started without a controlling terminal, and if the log file **/var/yp/ypxfr.log** exists, it appends all its output to that file. Since **ypxfr** is most often run from the privileged user's **crontab** file, or by **ypserv**, the log file can be used to retain a record of what was attempted, and what the results were.

For consistency between servers, **ypxfr** should be run periodically for every map in the NIS data base. Different maps change at different rates: a map may not change for months at a time, for instance, and may therefore be checked only once a day. Some maps may change several times per day. In such a case, you may want to check hourly for updates. A **crontab**(1) entry can be used to perform periodic updates automatically. Rather than having a separate **crontab** entry for each map, you can group commands to update several maps in a shell script. Examples (mnemonically named) are in **/usr/sbin/yp**: **ypxfr_1perday**, **ypxfr_2perday**, and **ypxfr_1perhour**. They can serve as reasonable first cuts.

Refer to **ypfiles**(4) for an overview of the NIS name service.

OPTIONS

–c Do not send a "Clear current map" request to the local **ypserv** process. Use this flag if **ypserv** is not running locally at the time you are running **ypxfr**. Otherwise, **ypxfr** complains that it cannot talk to the local **ypserv**, and the transfer fails.

–f Force the transfer to occur even if the version at the master is not more recent than the local version.

–C *tid prog server* This option is *only* for use by **ypserv**. When **ypserv** starts **ypxfr**, it specifies that **ypxfr** should call back a **yppush** process at the host *server*, registered as program number *prog*, and waiting for a response to transaction *tid*.

–d *ypdomain* Specify a domain other than the default domain.

–h *host* Get the map from *host*, regardless of what the map says the master is. If *host* is not specified, **ypxfr** asks the NIS service for the name of the master, and try to get the map from there. *host* must be a name.

–s *ypdomain* Specify a source domain from which to transfer a map that should be the same across domains.

FILES

/var/yp/ypxfr.log	log file
/usr/sbin/yp/ypxfr_1perday	script to run one transfer per day, for use with **cron**(1M)
/usr/sbin/yp/ypxfr_2perday	script to run two transfer per day, for use with **cron**(1M)
/usr/sbin/yp/ypxfr_1perhour	script for hourly transfers of volatile maps
/var/yp/*ypdomain*	NIS domain
/usr/spool/cron/crontabs/root	privileged user's **crontab** file

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

crontab(1), **cron**(1M), **yppush**(1M), **ypserv**(1M), **ypfiles**(4), **attributes**(5)

NAME

zdump – time zone dumper

SYNOPSIS

zdump [**–v**] [**–c** *cutoffyear*] [*zonename* . . .]

DESCRIPTION

The **zdump** command prints the current time for each timezone (*zonename*) listed on the command line. Specify *zonename* as the name of the timezone database file relative to **/usr/share/lib/zoneinfo**.

Specifying an invalid timezone (*zonename*) to **zdump** does not return an error, rather **zdump** uses GMT. This is consistent with the behavior of the library calls; **zdump** reflects the same behavior of the time routines in **libc**. See **ctime**(3C) and **mktime**(3C).

OPTIONS

The following options are supported:

–v Displays the entire contents of the timezone database file for *zonename*. Prints the time at the lowest possible time value, the time one day after the lowest possible time value, the times both one second before and exactly at each time at which the rules for computing local time change, the time at the highest possible time value, and the time at one day less than the highest possible time value. See **mktime**(3C) and **ctime**(3C) for information regarding time value (**time_t**). Each line of output ends with **isdst=1** if the given time is Daylight Saving Time or **isdst=0** otherwise.

–c *cutoffyear* Cuts off the verbose output near the start of the year *cutoffyear*.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred.

FILES

/usr/share/lib/zoneinfo standard zone information directory

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

zic(1M), **ctime**(3C), **mktime**(3C), **attributes**(5), **environ**(5)

NAME

zic – time zone compiler

SYNOPSIS

zic [**–s**] [**–v**] [**–l** *localtime*] [**–p** *posixrules*] [**–d** *directory*] [**–y** *yearistype*]
[*filename* . . .]

DESCRIPTION

zic reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is '–', the standard input is read.

Input lines are made up of fields. Fields are separated by any number of white space characters. Leading and trailing white space on input lines is ignored. A pound sign (**#**) indicates a comment and extends to the end of the line. White space characters and pound signs may be enclosed within double quotes (" ") if they are to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, or link lines.

Rule

A rule line has the form:

Rule	**NAME**	**FROM**	**TO**	**TYPE**	**IN**	**ON**	**AT**	**SAVE**	**LETTER/S**

For example:

Rule	**USA**	**1969**	**1973**	**–**	**Apr**	**lastSun**	**2:00**	**1:00**	**D**

The fields that make up a rule line are:

NAME Gives the (arbitrary) name of the set of rules this rule is part of.

FROM Gives the first year in which the rule applies. The word **minimum** (or an abbreviation) means the minimum year with a representable time value. The word **maximum** (or an abbreviation) means the maximum year with a representable time value.

TO Gives the final year in which the rule applies. In addition to **minimum** and **maximum** (as above), the word **only** (or an abbreviation) may be used to repeat the value of the **FROM** field.

TYPE Gives the type of year in which the rule applies. If **TYPE** is:

'–' The rule applies in all years between **FROM** and **TO** inclusive.
uspres The rule applies in U.S. Presidential election years.
nonpres The rule applies in years other than U.S. Presidential election years.
even The rule applies to even-numbered years.
odd The rule applies to odd-numbered years.

If **TYPE** is something else, then **zic** will attempt to execute the command

yearistype *year type*

to check the type of a year: an exit status of **0** means that the year is of the given type; an exit status of **1** means that the year is not of the given type. (Note: The **yearistype** command is not currently provided in the Solaris environment.)

IN Names the month in which the rule takes effect. Month names may be abbreviated.

ON Gives the day on which the rule takes effect. Recognized forms include:

5	the fifth day of the month
lastSun	the last Sunday in the month
lastMon	the last Monday in the month
Sun>=8	first Sunday on or after the eighth
Sun<=25	last Sunday on or before the 25th

Names of days of the week may be abbreviated or spelled out in full. Note: There can not be spaces within the **ON** field.

AT Gives the time of day at which the rule takes effect. Recognized forms include:

2	time in hours
2:00	time in hours and minutes
15:00	24-hour format time (for times after noon)
1:28:14	time in hours, minutes, and seconds

Any of these forms may be followed by the letter **w** if the given time is local "wall clock" time; **s** if the given time is local "standard" time; or **u** (or **g** or **z**) if the given time is universal time. In the absence of an indicator, wall clock time is assumed.

SAVE Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the **AT** field (without the **w** and **s** suffixes).

LETTER/S Gives the "variable part" (for example, the "S" or "D" in "EST" or "EDT" of time zone abbreviations to be used when this rule is in effect. If this field is '–', the variable part is null.

Zone A zone line has the form:

Zone	**NAME**	**GMTOFF**	**RULES/SAVE**	**FORMAT**	**[UNTIL]**

For example:

Zone	**Australia/SouthWest**	**9:30**	**–**	**CST**	**1992 Mar 15 12:00**
		8:30	**Aus**	**CST**	

The fields that make up a zone line are:

NAME The name of the time zone. This is the name used in creating the time conversion information file for the zone.

GMTOFF The amount of time to add to GMT to get standard time in this zone. This field has the same format as the **AT** and **SAVE** fields of rule lines; begin the field with a minus sign if time must be subtracted from GMT.

RULES/SAVE The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is '–' then standard time always applies in the time zone.

FORMAT The format for time zone abbreviations in this time zone. The pair of characters **%s** is used to show where the "variable part" of the time zone abbreviation goes.

UNTIL The time at which the GMT offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. The time of day has the same format as the **AT** field of rule lines. If this is specified, the time zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a "continuation" line; this has the same form as a zone line except that the string "Zone" and the name are omitted, as the continuation line will place information starting at the time specified as the **UNTIL** field in the previous line in the file used by the previous line. Continuation lines may contain an **UNTIL** field, just as zone lines do, indicating that the next line is a further continuation.

Link A link line has the form:

Link LINK-FROM LINK-TO

For example:

Link US/Eastern EST5EDT

The **LINK-FROM** field should appear as the **NAME** field in some zone line; the **LINK-TO** field is used as an alternate name for that zone.

Except for continuation lines, lines may appear in any order in the input.

OPTIONS

–d *directory* Create time conversion information files in the directory *directory* rather than in the standard directory **/usr/share/lib/zoneinfo**.

–l *localtime* Use the given time zone as local time *localtime*. **zic** will act as if the file contained a link line of the form:

Link *localtime* **localtime**

–p *posixrules* Use the rules of the given time zone *posixrules* when handling POSIX-format time zone environment variables. **zic** will act as if the input contained a link line of the form

Link *posixrules* **posixrules**

This option is not used by **ctime**(3C) and **mktime**(3C) in the Solaris environment.

−s Limit time values stored in output files to values that are the same whether they are taken to be signed or unsigned. You can use this option to generate SVVS-compatible files.

−v Complain if a year that appears in a data file is outside the range of years representable by system time values (**0:00:00 a.m. GMT**, **January 1**, **1970**, to **3:14:07 a.m. GMT**, **January 19**, **2038**).

−y *yearistype* Use the given command *yearistype* rather than **yearistype** when checking year types (see **Rules** under **DESCRIPTION**).

OPERANDS

filename A file containing input lines that specify the time conversion information files to be created. If a *filename* is '–', the standard input is read.

FILES

/usr/share/lib/zoneinfo standard directory used for created files

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

time(1), **ctime**(3C), **mktime**(3C), **attributes**(5)

NOTES

For areas with more than two types of local time, you may need to use local standard time in the **AT** field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

Index