Trusted Solaris 2.5 Man Pages: 7TSOL Devices

Sun Microsystems Federal, Inc.
A Sun Microsystems, Inc. Business
901 San Antonio Road, MS USJC01-201
Palo Alto, CA 94303
U.S.A.

Copyright 1997 Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

Sun, Sun Microsystems, the Sun logo, SunSoft, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, and NFS are trademarks, or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS : Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

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.

Copyright 1997 Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, Californie 94043-1100 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, SunSoft, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, et NFS sont des marques de fabrique ou des marques déposées, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

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.

Portions © AT&T 1983-1990 and reproduced with permission from AT&T.

Preface

In the Trusted Solaris Reference Manual, each collection of information on a particluar topic is called a man page, even though a man *page* may actually consist of *many pages* of text.

A man page is intended to answer concisely the question “What does it do?”. The man pages are not intended to be a tutorial. Depending what you are trying to do, refer to the other Trusted Solaris user, developer, and administrator manuals for when and why to use a command or other features described in the man pages.

ACCESSING MAN PAGES

The man pages that make up the reference manual may be accessed in three ways.

Note: The following discussion of man page viewing options uses the term **package**, which is a unit of software that is typically delivered on Sun’s product CDs. Installing the documentation packages is optional, because they are not required for operations. Each customer’s administrators decides whether or not the documentation packages are installed and made available.

The first means of accessing the man pages is through the use of the **man**(1) command. When the contents of the man page package, SUNWman, are available on the local system, anyone with a login account, plus a terminal emulator (such as **cmdtool**(1), **shelltool**(1), or **dtterm**(1)) and the **man**(1) command in one of the account’s execution profiles can view a man page on-line. (For more about Trusted Solaris execution profiles and user accounts, see the Trusted Solaris user and administrator

documentation.) To view a man page, enter the **man** command followed by the name of the man page. For example, to view the **ls**(1) man page that describes the command used to print out a directory's contents, a user enters the command: **man**ls.

The second way to read man pages is in the printed Trusted Solaris Reference Manual. The reference manual is in the Trusted Solaris documentation set, and it may be ordered in hardcopy form from Sun by using part number: 805-8005-10.

The third means of reading the man pages is by viewing them in AnswerBook format. When the Trusted Solaris AnswerBook package, SUNWtab, is available on the local system, anyone with a login account and with the **answerbook**() command and a terminal emulator in an execution profile can display the Trusted Solaris reference manual and the other user documentation. For Trusted Solaris 2.5, the Trusted Solaris documentation AnswerBook is shipped on a separate documentation CD, but it may be bundled on the same CD with the Trusted Solaris software in future releases.

Trusted Solaris man pages are identified with a TSOL suffix in the section name. The TSOL suffix is used for man pages that are either new to Trusted Solaris or modified from the base man pages from the Solaris, CDE, or Solstice products that are bundled into Trusted Solaris. The man pages are organized alphabetically by section.

- Section 1TSOL describes new or modified user commands available with the Trusted Solaris operating system.

- Section 1BTSOL describes printer commands adapted for Trusted Solaris from the Berkeley Software Distribution (BSD) print subsystem, which are used chiefly for printing administration.

  Note: Use of the equivalent System V print commands is recommended (such as **lp**(1TSOL)instead of **lpr**(1BTSOL)) because although the BSD commands are included for compatability, they will be removed in future releases.

- Section 1MTSOL describes Trusted Solaris system maintenance and administration commands.

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

- 3*TSOL subsections describe functions found in various Trusted Solaris libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2TSOL.

Subsections include: 3CTSOL, 3NTSOL, 3RTSOL, 3TSOL, and 3X11TSOL.

- Section 4TSOL outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5TSOL contains miscellaneous documentation such as Trusted Solaris macros.
- 7*TSOL subsections describe various special files that refer to specific hardware peripherals and device drivers.

  Subsections include: 7DTSOL and 7TSOL.
- 9*TSOL subsections provide reference information for writing device drivers in the kernel operating system environment.

  Subsections include: 9FTSOL and 9TSOL.

Following is a generic list of headings on each man page. The man pages of each manual section include only the headings they need. 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.

AVAILABILITY

This section briefly states any limitations on the availabilty of the command. These limitations could be hardware or software specific.

A specification of a class of hardware platform, such as **x86** or **SPARC**, denotes that the command or interface is applicable for the hardware platform specified.

In Section 1TSOL and Section 1MTSOL, **AVAILABILITY** indicates which package contains the command being described on the manual page. In order to use the command, the specified package must have been installed with the operating system. If the package was not installed, see **pkgadd**(1) for information on how to upgrade.

MT-LEVEL

This section lists the **MT-LEVEL** of the library functions described in the Section 3 manual pages. The **MT-LEVEL** defines the libraries' ability to support threads. See **Intro**(3TSOL) for more information.

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 7TSOL 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 in an administrative role,

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.

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.

SUMMARY OF TRUSTED SOLARIS CHANGES

On base man pages that have Trusted Solaris modifications, this section summarizes the changes in a single easy-to-find place on the man page.

NAME

Intro, intro – introduction to special files

DESCRIPTION

This section describes various device and network interfaces available on the system. The types of interfaces described include character and block devices, STREAMS modules, network protocols, file systems, and ioctl requests for driver subsystems and classes.

This section contains the following major collections:

(7D) The system provides drivers for a variety of hardware devices, such as disk, magnetic tapes, serial communication lines, mice, and frame buffers, as well as virtual devices such as pseudo-terminals and windows.

This section describes special files that refer to specific hardware peripherals and device drivers. STREAMS device drivers are also described. Characteristics of both the hardware device and the corresponding device driver are discussed where applicable.

An application accesses a device through that device's special file. This section specifies the device special file to be used to access the device as well as application programming interface (API) information relevant to the use of the device driver.

All device special files are located under the **/devices** directory. The **/devices** directory hierarchy attempts to mirror the hierarchy of system busses, controllers, and devices configured on the system. Logical device names for special files in **/devices** are located under the **/dev** directory. Although not every special file under **/devices** will have a corresponding logical entry under **/dev**, whenever possible, an application should reference a device using the logical name for the device. Logical device names are listed in the **FILES** section of the page for the device in question.

This section also describes driver configuration where applicable. Many device drivers have a driver configuration file of the form *driver_name***.conf** associated with them (see **driver.conf**(4)). The configuration information stored in the driver configuration file is used to configure the driver and the device. Driver configuration files are located in **/kernel/drv** and **/usr/kernel/drv**. Driver configuration files for platform dependent drivers are located in **/platform/‘uname -i‘/kernel/drv** where **‘uname -i‘** is the output of the **uname**(1) command with the **-i** option.

Some driver configuration files may contain user configurable properties. Changes in a driver's configuration file will not take effect until the system is rebooted or the driver has been removed and re-added (see **rem_drv**(1M) and **add_drv**(1M)).

(7FS) This section describes the programmatic interface for several file systems supported by SunOS.

(7I) This section describes ioctl requests which apply to a class of drivers or subsystems. For example, ioctl requests which apply to most tape devices are

discussed in **mtio**(7I). Ioctl requests relevant to only a specific device are described on the man page for that device. The page for the device in question should still be examined for exceptions to the ioctls listed in section 7I.

(7ITSOL)
This section describes Trusted Solaris new and modified ioctl requests.

(7M) This section describes STREAMS modules. Note that STREAMS drivers are discussed in section 7D. **streamio**(7ITSOL) **streamio**(7I) contains a list of ioctl requests used to manipulate STREAMS modules and interface with the STREAMS framework. Ioctl requests specific to a STREAMS module will be discussed on the man page for that module.

(7P) This section describes various network protocols available in SunOS.

SunOS supports both socket-based and STREAMS-based network communications. The Internet protocol family, described in **inet**(7P), is the primary protocol family supported by SunOS, although the system can support a number of others. The raw interface provides low-level services, such as packet fragmentation and reassembly, routing, addressing, and basic transport for socket-based implementations. Facilities for communicating using an Internet-family protocol are generally accessed by specifying the **AF_INET** address family when binding a socket; see **socket**(3N) for details.

Major protocols in the Internet family include:

- The Internet Protocol (IP) itself, which supports the universal datagram format, as described in **ip**(7P). This is the default protocol for **SOCK_RAW** type sockets within the **AF_INET** domain.
- The Transmission Control Protocol (TCP); see **tcp**(7P). This is the default protocol for **SOCK_STREAM** type sockets.
- The User Datagram Protocol (UDP); see **udp**(4P). This is the default protocol for **SOCK_DGRAM** type sockets.
- The Address Resolution Protocol (ARP); see **arp**(7P).
- The Internet Control Message Protocol (ICMP); see **icmp**(7P).

SEE ALSO

add_drv(1M), **rem_drv**(1M), **intro**(2), **ioctl**(2), **socket**(3N), **driver.conf**(4), **arp**(7P), **icmp**(7P), **inet**(7P), **ip**(7P), **mtio**(7I), **st**(7D), **streamio**(7ITSOL), **tcp**(7P), **udp**(7P)

Solaris 1.x to 2.x Transition Guide

TCP/IP and Data Communications Administration Guide

STREAMS Programming Guide

Writing Device Drivers

Trusted Solaris Developer's Guide

Name	Description
sad(7DTSOL)	STREAMS Administrative Driver
wscons(7DTSOL)	workstation console

NAME

sad – STREAMS Administrative Driver

SYNOPSIS

```
#include <sys/types.h>
#include <sys/conf.h>
#include <sys/sad.h>
#include <sys/stropts.h>

int ioctl ( int fildes, int command, int arg);
```

DESCRIPTION

The STREAMS Administrative Driver provides an interface for applications to perform administrative operations on STREAMS modules and drivers. The interface is provided through **ioctl**(2) commands. Privileged operations may access the **sad** driver using **/dev/sad/admin**. The requesting process must have **PRIV_SYS_DEVICES** privilege in its effective set. Unprivileged operations may access the **sad** driver using **/dev/sad/user**.

fildes is an open file descriptor that refers to the **sad** driver. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a *command*-specific data structure.

COMMAND FUNCTIONS

The autopush facility (see **autopush**(1M)) allows one to configure a list of modules to be automatically pushed on a stream when a driver is first opened. Autopush is controlled by the following commands:

SAD_SAP Allows the administrator to configure the given device's autopush information. *arg* points to a **strapush** structure, which contains the following members:

```
uint   sap_cmd;
long   sap_major;
long   sap_minor;
long   sap_lastminor;
long   sap_npush;
uint   sap_list [MAXAPUSH] [FMNAMESZ + 1];
```

The **sap_cmd** field indicates the type of configuration being done. It may take on one of the following values:

SAP_ONE	Configure one minor device of a driver.
SAP_RANGE	Configure a range of minor devices of a driver.
SAP_ALL	Configure all minor devices of a driver.
SAP_CLEAR	Undo configuration information for a driver.

The **sap_major** field is the major device number of the device to be configured. The **sap_minor** field is the minor device number of the device to be configured. The **sap_lastminor** field is used only with the **SAP_RANGE** command, which configures a range of minor devices between **sap_minor** and **sap_lastminor**, inclusive. The minor fields have no meaning for the **SAP_ALL** command. The **sap_npush** field indicates the number of modules to be automatically pushed when the device is opened. It must be less than or equal to **MAXAPUSH** , defined in **sad.h**. It must also be less than or equal to **NSTRPUSH**, the maximum number of modules that can be pushed on a stream, defined in the kernel master file. The field **sap_list** is an array of NULL-terminated module names to be pushed in the order in which they appear in the list.

When using the **SAP_CLEAR** command, the user sets only **sap_major** and **sap_minor**. This will undo the configuration information for any of the other commands. If a previous entry was configured as **SAP_ALL**, **sap_minor** should be set to zero. If a previous entry was configured as **SAP_RANGE** , **sap_minor** should be set to the lowest minor device number in the range configured.

On failure, **errno** is set to the following value:

EPERM the requesting process does not have **PRIV_SYS_DEVICES** privilege in its effective set.

EFAULT *arg* points outside the allocated address space.

EINVAL The major device number is invalid, the number of modules is invalid, or the list of module names is invalid.

ENOSTR The major device number does not represent a STREAMS driver.

EEXIST The major-minor device pair is already configured.

ERANGE The command is **SAP_RANGE** and **sap_lastminor** is not greater than **sap_minor**, or the command is **SAP_CLEAR** and **sap_minor** is not equal to the first minor in the range.

ENODEV The command is **SAP_CLEAR** and the device is not configured for autopush.

ENOSR An internal autopush data structure cannot be allocated.

SAD_GAP Allows any user to query the **sad** driver to get the autopush configuration information for a given device. *arg* points to a **strapush** structure as described in the previous command.

The user should set the **sap_major** and **sap_minor** fields of the **strapush** structure to the major and minor device numbers, respectively, of the device in question. On return, the **strapush** structure will be filled in with the entire information used to configure the device. Unused entries in the module list will be zero-filled.

On failure, **errno** is set to one of the following values:

EFAULT *arg* points outside the allocated address space.

EINVAL The major device number is invalid.

ENOSTR The major device number does not represent a STREAMS driver.

ENODEV The device is not configured for autopush.

SAD_VML Allows any user to validate a list of modules (that is, to see if they are installed on the system). *arg* is a pointer to a **str_list** structure with the following members:

```
int                 sl_nmods;
struct str_mlist    *sl_modlist;
```

The **str_mlist** structure has the following member:

```
char                l_name[FMNAMESZ+1];
```

sl_nmods indicates the number of entries the user has allocated in the array and **sl_modlist** points to the array of module names. The return value is 0 if the list is valid, 1 if the list contains an invalid module name, or –1 on failure. On failure, **errno** is set to one of the following values:

EFAULT *arg* points outside the allocated address space.

EINVAL The **sl_nmods** field of the **str_list** structure is less than or equal to zero.

SUMMARY OF TRUSTED SOLARIS CHANGES

The **PRIV_SYS_DEVICES** privilege is required to perform the privileged operations through the administrative driver.

SEE ALSO

intro(2TSOL), **ioctl**(2), **open**(2TSOL)

STREAMS Programming Guide

DIAGNOSTICS

Unless otherwise specified, the return value from **ioctl** is **0** upon success and **–1** upon failure with **errno** set as indicated.

NAME

wscons – workstation console

SYNOPSIS

#include <sys/strredir.h>

ioctl(*fd***, SRIOCSREDIR,** *target***);**
ioctl(*fd***, SRIOCISREDIR,** *target***);**

DESCRIPTION

The "workstation console" is a device consisting of the combination of the workstation keyboard and frame buffer, acting in concert to emulate an ASCII terminal. It includes a redirection facility that allows I/O issued to the workstation console to be diverted to some other STREAMS device, so that, for example, window systems can arrange to redirect output that would otherwise appear directly on the frame buffer, corrupting its appearance.

Redirection

The redirection facility maintains a list of devices that have been named as redirection targets, through the **SRIOCSREDIR** ioctl described below. All entries but the most recent are inactive; when the currently active entry is closed, the most recent remaining entry becomes active. The active entry acts as a proxy for the device being redirected; it handles all **read**(2), **write**(2), **ioctl**(2), and **poll**(2) calls issued against the redirectee.

The following two ioctls control the redirection facility. In both cases, *fd* is a descriptor for the device being redirected (that is, the workstation console) and *target* is a descriptor for a STREAMS device.

SRIOCSREDIR Make *target* be the source and destination of I/O ostensibly directed to the device denoted by *fd*. The requesting process must have the **PRIV_SYS_C0NSOLE** privilege in its effective set for the operation to succeed.

SRIOCISREDIR Returns 1 if *target* names the device currently acting as proxy for the device denoted by *fd*, and 0 if it is not.

SPARC and PowerPC Edition: ANSI STANDARD TERMINAL EMULATION

On SPARC systems, the PROM monitor emulates an ANSI X3.64 terminal.

On PowerPC systems, the kernel display subsystem performs the same emulation.

Note: the VT100 also follows the ANSI X3.64 standard but both the Sun and the VT100 have nonstandard extensions to the ANSI X3.64 standard. The Sun terminal emulator and the VT100 are *not* compatible in any true sense.

The Sun console displays 34 lines of 80 ASCII characters per line, with scrolling, (*x*, *y*) cursor addressability, and a number of other control functions.

While the display size is usually 34 by 80, there are instances where it may be a different size.

- If the display device is not large enough to display 34 lines of text.
- On SPARC systems, if either screen-#rows or screen-#columns is set by the user to a value other than the default of 34 or 80 respectively. screen-#rows and screen-#columns are fields stored in NVRAM/EEPROM, see **eeprom**(1M).

The Sun console displays a cursor which marks the current line and character position on the screen. ASCII characters between 0x20 (space) and 0x7E (tilde) inclusive are printing characters — when one is written to the Sun console (and is not part of an escape sequence), it is displayed at the current cursor position and the cursor moves one position to the right on the current line.

On SPARC systems, later PROM revisions have the full 8-bit ISO Latin-1 (ISO 8859-1) character set, not just ASCII. Earlier PROM revisions display characters in the range 0xA0 – 0xFE as spaces.

PowerPC systems have the full 8-bit ISO Latin-1 (ISO 8859-1) character set.

If the cursor is already at the right edge of the screen, it moves to the first character position on the next line. If the cursor is already at the right edge of the screen on the bottom line, the Line-feed function is performed (see CTRL-J below), which scrolls the screen up by one or more lines or wraps around, before moving the cursor to the first character position on the next line.

SPARC and PowerPC Edition: Control Sequence

The Sun console defines a number of control sequences which may occur in its input. When such a sequence is written to the Sun console, it is not displayed on the screen, but effects some control function as described below, for example, moves the cursor or sets a display mode.

Some of the control sequences consist of a single character. The notation

CTRL-*X*

for some character *X*, represents a control character.

Other ANSI control sequences are of the form

ESC [*params char*

Spaces are included only for readability; these characters must occur in the given sequence without the intervening spaces.

ESC represents the ASCII escape character (ESC, CTRL-[, 0x1B).

[The next character is a left square bracket '[' (0x5B).

params are a sequence of zero or more decimal numbers made up of digits between 0 and 9, separated by semicolons.

char represents a function character, which is different for each control sequence.

Some examples of syntactically valid escape sequences are (again, ESC represent the single ASCII character 'Escape'):

ESC[m	select graphic rendition with default parameter
ESC[7m	select graphic rendition with reverse image
ESC[33;54H	set cursor position
ESC[123;456;0;;3;B	move cursor down

Syntactically valid ANSI escape sequences which are not currently interpreted by the Sun console are ignored. Control characters which are not currently interpreted by the Sun console are also ignored.

Each control function requires a specified number of parameters, as noted below. If fewer parameters are supplied, the remaining parameters default to 1, except as noted in the descriptions below.

If more than the required number of parameters is supplied, only the last *n* are used, where *n* is the number required by that particular command character. Also, parameters which are omitted or set to zero are reset to the default value of 1 (except as noted below).

Consider, for example, the command character M which requires one parameter. ESC[;M and ESC[0M and ESC[M and ESC[23;15;32;1M are all equivalent to ESC[1M and provide a parameter value of 1. Note: ESC[;5M (interpreted as 'ESC[5M') is *not* equivalent to ESC[5;M (interpreted as 'ESC[5;1M') which is ultimately interpreted as 'ESC[1M').

In the syntax descriptions below, parameters are represented as '**#**' or '**#1;#2**'.

SPARC and PowerPC Edition: ANSI Control

The following paragraphs specify the ANSI control functions implemented by the Sun console. Each description gives:

- the control sequence syntax
- the hex equivalent of control characters where applicable
- the control function name and ANSI or Sun abbreviation (if any).
- description of parameters required, if any
- description of the control function
- for functions which set a mode, the initial setting of the mode. The initial settings can be restored with the SUNRESET escape sequence.

SPARC: Control Character Functions

CTRL-G (0x7) Bell (BEL)

The Sun Workstation Model 100 and 100U is not equipped with an audible bell. It 'rings the bell' by flashing the entire screen. The window system flashes the window. The screen will also be flashed on current models if the Sun keyboard is not the console input device.

CTRL-H (0x8) Backspace (BS)

The cursor moves one position to the left on the current line. If it is already at the left edge of the screen, nothing happens.

CTRL-I (0x9) Tab (TAB)

The cursor moves right on the current line to the next tab stop. The tab stops are fixed at every multiple of 8 columns. If the cursor is already at the right edge of the screen, nothing happens; otherwise the cursor moves right a minimum of one and a maximum of eight character positions.

CTRL-J (0xA) Line-feed (LF)

The cursor moves down one line, remaining at the same character position on the line. If the cursor is already at the bottom line, the screen either scrolls up or ''wraps around'' depending on the setting of an internal variable *S* (initially 1) which can be changed by the ESC[r control sequence. If *S* is greater than zero,

the entire screen (including the cursor) is scrolled up by *S* lines before executing the line-feed. The top *S* lines scroll off the screen and are lost.
S new blank lines scroll onto the bottom of the screen. After scrolling, the line-feed is executed by moving the cursor down one line.

If *S* is zero, 'wrap-around' mode is entered. 'ESC [1 r' exits back to scroll mode. If a line-feed occurs on the bottom line in wrap mode, the cursor goes to the same character position in the top line of the screen. When any line-feed occurs, the line that the cursor moves to is cleared. This means that no scrolling occurs. Wrap-around mode is not implemented in the window system.

On SPARC systems, the screen scrolls as fast as possible depending on how much data is backed up waiting to be printed. Whenever a scroll must take place and the console is in normal scroll mode ('ESC [1 r'), it scans the rest of the data awaiting printing to see how many line-feeds occur in it. This scan stops when any control character from the set {**VT**, **FF**, **SO**, **SI**, **DLE**, **DC1**, **DC2**, **DC3**, **DC4**, **NAK**, **SYN**, **ETB**, **CAN**, **EM**, **SUB**, **ESC**, **FS**, **GS**, **RS**, **US**} is found. At that point, the screen is scrolled by N lines ($N \geq 1$) and processing continues. The scanned text is still processed normally to fill in the newly created lines. This results in much faster scrolling with scrolling as long as no escape codes or other control characters are intermixed with the text.

See also the discussion of the 'Set scrolling' **(ESC[r)** control function below.

CTRL-K (0xB) Reverse Line-feed
The cursor moves up one line, remaining at the same character position on the line. If the cursor is already at the top line, nothing happens.

CTRL-L (0xC) Form-feed (FF)
The cursor is positioned to the Home position (upper-left corner) and the entire screen is cleared.

CTRL-M (0xD) Return (CR)
The cursor moves to the leftmost character position on the current line.

SPARC and PowerPC Edition: Escape Sequence

CTRL-[(0x1B) Escape (ESC)
This is the escape character. Escape initiates a multi-character control sequence.

ESC[#@ Insert Character (ICH)
Takes one parameter, # (default 1). Inserts # spaces at the current cursor position. The tail of the current line starting at the current cursor position inclusive is shifted to the right by # character positions to make room for the spaces. The rightmost # character positions shift off the line and are lost. The position of the cursor is unchanged.

ESC[#A Cursor Up (CUU)
Takes one parameter, # (default 1). Moves the cursor up # lines. If the cursor is fewer than # lines from the top of the screen, moves the cursor to the topmost line on the screen. The character position of the cursor on the line is unchanged.

ESC[#B Cursor Down (CUD)
Takes one parameter, # (default 1). Moves the cursor down # lines. If the cursor is fewer than # lines from the bottom of the screen, move the cursor to the last line on the screen. The character position of the cursor on the line is unchanged.

ESC[#C Cursor Forward (CUF)
Takes one parameter, # (default 1). Moves the cursor to the right by # character positions on the current line. If the cursor is fewer than # positions from the right edge of the screen, moves the cursor to the rightmost position on the current line.

ESC[#D Cursor Backward (CUB)
Takes one parameter, # (default 1). Moves the cursor to the left by # character positions on the current line. If the cursor is fewer than # positions from the left edge of the screen, moves the cursor to the leftmost position on the current line.

ESC[#E Cursor Next Line (CNL)
Takes one parameter, # (default 1). Positions the cursor at the leftmost character position on the #-th line below the current line. If the current line is less than # lines from the bottom of the screen, positions the cursor at the leftmost character position on the bottom line.

ESC[#1;#2f Horizontal And Vertical Position (HVP)
or
ESC[#1;#2H Cursor Position (CUP)
Takes two parameters, #1 and #2 (default 1, 1). Moves the cursor to the #2-th character position on the #1-th line. Character positions are numbered from 1 at the left edge of the screen; line positions are numbered from 1 at the top of the screen. Hence, if both parameters are omitted, the default action moves the cursor to the home position (upper left corner). If only one parameter is supplied, the cursor moves to column 1 of the specified line.

ESC[J Erase in Display (ED)
Takes no parameters. Erases from the current cursor position inclusive to the end of the screen. In other words, erases from the current cursor position inclusive to the end of the current line and all lines below the current line. The cursor position is unchanged.

ESC[K Erase in Line (EL)
Takes no parameters. Erases from the current cursor position inclusive to the end of the current line. The cursor position is unchanged.

ESC[#L Insert Line (IL)
Takes one parameter, # (default 1). Makes room for # new lines starting at the current line by scrolling down by # lines the portion of the screen from the current line inclusive to the bottom. The # new lines at the cursor are filled with spaces; the bottom # lines shift off the bottom of the screen and are lost. The

position of the cursor on the screen is unchanged.

ESC[#M Delete Line (DL)

Takes one parameter, # (default 1). Deletes # lines beginning with the current line. The portion of the screen from the current line inclusive to the bottom is scrolled upward by # lines. The # new lines scrolling onto the bottom of the screen are filled with spaces; the # old lines beginning at the cursor line are deleted. The position of the cursor on the screen is unchanged.

ESC[#P Delete Character (DCH)

Takes one parameter, # (default 1). Deletes # characters starting with the current cursor position. Shifts to the left by # character positions the tail of the current line from the current cursor position inclusive to the end of the line. Blanks are shifted into the rightmost # character positions. The position of the cursor on the screen is unchanged.

ESC[#m Select Graphic Rendition (SGR)

Takes one parameter, # (default 0). Note: unlike most escape sequences, the parameter defaults to zero if omitted. Invokes the graphic rendition specified by the parameter. All following printing characters in the data stream are rendered according to the parameter until the next occurrence of this escape sequence in the data stream. Currently only two graphic renditions are defined:

0 Normal rendition.

7 Negative (reverse) image.

Negative image displays characters as white-on-black if the screen mode is currently black-on white, and vice-versa. Any non-zero value of # is currently equivalent to 7 and selects the negative image rendition.

ESC[p Black On White (SUNBOW)

Takes no parameters. Sets the screen mode to black-on-white. If the screen mode is already black-on-white, has no effect. In this mode spaces display as solid white, other characters as black-on-white. The cursor is a solid black block. Characters displayed in negative image rendition (see 'Select Graphic Rendition' above) is white-on-black in this mode. This is the initial setting of the screen mode on reset.

ESC[q White On Black (SUNWOB)

Takes no parameters. Sets the screen mode to white-on-black. If the screen mode is already white-on-black, has no effect. In this mode spaces display as solid black, other characters as white-on-black. The cursor is a solid white block. Characters displayed in negative image rendition (see 'Select Graphic Rendition' above) is black-on-white in this mode. The initial setting of the screen mode on reset is the alternative mode, black on white.

ESC[#r Set scrolling (SUNSCRL)

Takes one parameter, # (default 0). Sets to # an internal register which determines how many lines the screen scrolls up when a line-feed function is performed with the cursor on the bottom line. A parameter of 2 or 3 introduces a small amount of ''jump'' when a scroll occurs. A parameter of 34 clears the

screen rather than scrolling. The initial setting is 1 on reset.

A parameter of zero initiates ''wrap mode'' instead of scrolling. In wrap mode, if a linefeed occurs on the bottom line, the cursor goes to the same character position in the top line of the screen. When any linefeed occurs, the line that the cursor moves to is cleared. This means that no scrolling ever occurs. 'ESC [1 r' exits back to scroll mode.

For more information, see the description of the Line-feed (CTRL-J) control function above.

ESC[s Reset terminal emulator (SUNRESET)
Takes no parameters. Resets all modes to default, restores current font from PROM. Screen and cursor position are unchanged.

RETURN VALUES

When there are no errors, the redirection ioctls have return values as described above. Otherwise, they return **–1** and set **errno** to indicate the error.

If the *target* stream is in an error state, **errno** is set accordingly.

ERRORS

EPERM	An **SRIOCSREDIR** command is issued, and the requesting process does not have the **PRIV_SYS_CONSOLE** privilege in its effective set.
EBADF	*target* does not denote an open file.
ENOSTR	*target* does not denote a STREAMS device.
EINVAL	(x86 only) *fd* does not denote **/dev/console**.

SUMMARY OF TRUSTED SOLARIS CHANGES

To succeed, the **SRIOCSREDIR** command requires the **PRIV_SYS_CONSOLE** privilege. A new error code **EPERM** is added.

x86 FILES

/dev/systty	(x86 only)
/dev/syscon	(x86 only)
/dev/console	(x86 only) the device that must be opened for the **SRIOCSREDIR** and **SRIOCISREDIR** ioctls
/dev/wscons	the workstation console, accessed by way of the redirection facility

SEE ALSO

console(7D)

WARNINGS

The redirection ioctls block while there is I/O outstanding on the device instance being redirected. Thus, attempting to redirect the workstation console while there is a read outstanding on it will hang until the read completes.

Index