System Administration Guide: Basic Administration

Part No: 821–1451–01
November 2010

Copyright © 1998, 2010, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related software documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are “commercial computer software” or “commercial technical data” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Copyright © 1998, 2010, Oracle et/ou ses affiliés. Tous droits réservés.

Ce logiciel et la documentation qui l’accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d’utilisation et de divulgation. Sauf disposition de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, breveter, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d’interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu’elles soient exemptes d’erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l’accompagne, est concédé sous licence au Gouvernement des Etats-Unis, ou à toute entité qui délivre la licence de ce logiciel ou l’utilise pour le compte du Gouvernement des Etats-Unis, la notice suivante s’applique :

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d’applications de gestion des informations. Ce logiciel ou matériel n’est pas conçu ni n’est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer des dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d’applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l’utilisation de ce logiciel ou matériel pour ce type d’applications.

Oracle et Java sont des marques déposées d’Oracle Corporation et/ou de ses affiliés.Tout autre nom mentionné peut correspondre à des marques appartenant à d’autres propriétaires qu’Oracle.

AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d’Advanced Micro Devices. Intel et Intel Xeon sont des marques ou des marques déposées d’Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. UNIX est une marque déposée concédé sous license par X/Open Company, Ltd.

110425@25097

Contents

Preface

System Administration Guide: Basic Administration is part of a documentation set that includes a significant part of the Oracle Solaris system administration information. This guide contains information for both SPARC based and x86 based systems.

This book assumes you have completed the following tasks:

- Installed the Oracle Solaris 11 Express software
- Set up all the networking software that you plan to use

For the Oracle Solaris 11 Express release, new features that might be interesting to system administrators are covered in sections called *What's New in ... ?* in the appropriate chapters.

Note – This Oracle Solaris release supports systems that use the SPARC and x86 families of processor architectures. The supported systems appear in the Oracle Solaris OS: Hardware Compatibility Lists (`http://www.sun.com/bigadmin/hcl`). This document cites any implementation differences between the platform types.

In this document these x86 related terms mean the following:

- "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
- "x64" relates specifically to 64-bit x86 compatible CPUs.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

For supported systems, see the *Oracle Solaris OS: Hardware Compatibility Lists*.

Who Should Use This Book

This book is intended for anyone responsible for administering one or more systems running the Oracle Solaris 11 Express release. To use this book, you should have 1–2 years of UNIX system administration experience. Attending UNIX system administration training courses might be helpful.

How the System Administration Guides Are Organized

Here is a list of the topics that are covered by the System Administration Guides.

Book Title	Topics
System Administration Guide: Basic Administration	User accounts and groups, shutting down and booting a system, and managing services
System Administration Guide: Advanced Administration	Terminals and modems, system resources, system processes, and troubleshooting Oracle Solaris software problems
System Administration Guide: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data
System Administration Guide: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, IP filter, Mobile IP, and IPQoS
System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)	DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP
System Administration Guide: Network Interfaces and Network Virtualization	Networking stack, NIC driver property configuration, NWAM configuration, manual network interface configuration, administration of VLANs and link aggregations, IP network multipathing (IPMP), WiFi wireless networking configuration, virtual NICs (VNICs), and network resource management
System Administration Guide: Network Services	Web cache servers, time-related services, network file systems (NFS and autofs), mail, SLP, and PPP
System Administration Guide: Printing	Printing topics and tasks, using services, tools, protocols, and technologies to set up and administer printing services and printers
System Administration Guide: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Oracle Solaris Cryptographic Framework, privileges, RBAC, SASL, and Oracle Solaris Secure Shell
System Administration Guide: Oracle Solaris Zones, Oracle Solaris 10 Containers, and Resource Management	Resource management features, which enable you to control how applications use available system resources; Oracle Solaris Zones software partitioning technology, which virtualizes operating system services to create an isolated environment for running applications; and Oracle Solaris 10 Containers, which host Oracle Solaris 10 environments running on the Oracle Solaris 11 Express kernel

Book Title	Topics
Oracle Solaris SMB and Windows Interoperability Administration Guide	Oracle Solaris SMB service, which enables you to configure an Oracle Solaris system to make SMB shares available to SMB clients; Oracle Solaris SMB client, which enables you to access SMB shares; and native identity mapping service, which enables you to map user and group identities between Oracle Solaris systems and Windows systems
Oracle Solaris Trusted Extensions Configuration and Administration	System installation, configuration, and administration that is specific to the Oracle Solaris' Trusted Extensions feature
Oracle Solaris ZFS Administration Guide	ZFS storage pool and file system creation and management, snapshots, clones, backups, using access control lists (ACLs) to protect ZFS files, using ZFS on an Oracle Solaris system with zones installed, emulated volumes, and troubleshooting and data recovery

Related Third-Party Web Site References

Note – Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

See the following web sites for additional resources:

- Documentation (`http://docs.sun.com`)
- Support (`http://www.oracle.com/us/support/systems/index.html`)
- Training (`http://education.oracle.com`) – Click the Sun link in the left navigation bar.

Oracle Software Resources

Oracle Technology Network (`http://www.oracle.com/technetwork/index.html`) offers a range of resources related to Oracle software:

- Discuss technical problems and solutions on the Discussion Forums (`http://forums.oracle.com`).
- Get hands-on step-by-step tutorials with Oracle By Example (`http://www.oracle.com/technetwork/tutorials/index.html`).
- Download Sample Code (`http://www.oracle.com/technology/sample_code/index.html`).

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

General Conventions

Be aware of the following conventions used in this book.

- When following steps or using examples, be sure to type double-quotes ("), left single-quotes (‘), and right single-quotes (`'`) exactly as shown.
- The key referred to as Return is labeled Enter on some keyboards.
- The root path usually includes the `/sbin`, `/usr/sbin`, `/usr/bin`, and `/etc` directories, so the steps in this book show the commands in these directories without absolute path names. Steps that use commands in other, less common, directories show the absolute paths in the examples.

◆ ◆ ◆ CHAPTER 1

Managing User Accounts and Groups (Overview)

This is a list of the information that is in this chapter:

What Are User Accounts and Groups?

One basic system administration task is to set up a user account for each user at a site. A typical user account includes the information a user needs to log in and use a system, without having the system's root password. User account components are described in "User Account Components" on page 17.

When you set up a user account, you can add the user to a predefined groups of users. A typical use of groups is to set up group permissions on a file and directory, which allows access only to those users who are part of that group.

For example, you might have a directory containing confidential files that only a few users should be able to access. You could set up a group called `topsecret` that includes the users that are working on the `topsecret` project. In addition, you could set up the `topsecret` files with read permission for the `topsecret` group. That way, only the users in the `topsecret` group would be able to read the files.

A special type of user account, called a *role*, gives selected users special privileges. For more information, see "Role-Based Access Control (Overview)" in *System Administration Guide: Security Services*.

User Account Components

The following sections describe the specific components of a user account.

User (Login) Names

User names, also called *login names*, let users access their own systems and remote systems that have the appropriate access privileges. You must choose a user name for each user account that you create.

Consider establishing a standard way of assigning user names so that they are easier for you to track. Also, names should be easy for users to remember. A simple scheme when selecting a user name is to use the first name initial and first seven letters of the user's last name. For example, Ziggy Ignatz becomes `zignatz`. If this scheme results in duplicate names, you can use the first initial, middle initial, and the first six characters of the user's last name. For example, Ziggy Top Ignatz becomes `ztignatz`.

If this scheme still results in duplicate names, consider using the following scheme to create a user name:

- The first initial, middle initial, first five characters of the user's last name
- The number 1, or 2, or 3, and so on, until you have a unique name

---

Note – Each new user name must be distinct from any mail aliases that are known to the system or to a NIS domain. Otherwise, mail might be delivered to the alias rather than to the actual user.

For detailed guidelines on setting up user (login) names, see "Guidelines for Assigning User Names, User IDs, and Group IDs" on page 24.

User ID Numbers

Associated with each user name is a user identification number (UID). The UID number identifies the user name to any system on which the user attempts to log in. And, the UID number is used by systems to identify the owners of files and directories. If you create user accounts for a single individual on a number of different systems, always use the same user name and ID number. In that way, the user can easily move files between systems without ownership problems.

UID numbers must be a whole number that is less than or equal to 2147483647. UID numbers are required for both regular user accounts and special system accounts. The following table lists the UID numbers that are reserved for user accounts and system accounts.

TABLE 1–1 Reserved UID Numbers

UID Numbers	User or Login Accounts	Description
0 – 99	root, `daemon`, `bin`, `sys`, and so on	Reserved for use by the operating system
100 – 2147483647	Regular users	General purpose accounts

TABLE 1–1 Reserved UID Numbers *(Continued)*

UID Numbers	User or Login Accounts	Description
60001 and 65534	`nobody` and `nobody4`	Anonymous users
60002	`noaccess`	Non trusted users

Do not assign UIDs 0 through 99. These UIDs are reserved for allocation by Oracle Solaris. By definition, `root` always has UID 0, `daemon` has UID 1, and pseudo-user `bin` has UID 2. In addition, you should give `uucp` logins and pseudo user logins, such as `who`, `tty`, and `ttytype`, low UIDs so that they fall at the beginning of the `passwd` file.

For additional guidelines on setting up UIDs, see "Guidelines for Assigning User Names, User IDs, and Group IDs" on page 24.

As with user (login) names, you should adopt a scheme for assigning unique UID numbers. Some companies assign unique employee numbers. Then, administrators add a number to the employee number to create a unique UID number for each employee.

To minimize security risks, you should avoid reusing the UIDs from deleted accounts. If you must reuse a UID, "wipe the slate clean" so that the new user is not affected by attributes set for a former user. For example, a former user might have been denied access to a printer by being included in a printer deny list. However, that attribute might be inappropriate for the new user.

Using Large User IDs and Group IDs

UIDs and group IDs (GIDs) can be assigned up to the maximum value of a signed integer, or 2147483647.

However, UIDs and GIDs over 60000 do not have full functionality and are incompatible with many Oracle Solaris features. So, avoid using UIDs or GIDs over 60000.

The following table describes interoperability issues with Oracle Solaris products and previous releases.

TABLE 1–2 Interoperability Issues for UIDs or GIDs Over 60000

Category	Product or Command	Issue
NFS interoperability	SunOS 4.0 NFS software and compatible releases	NFS server and client code truncates large UIDs and GIDs to 16 bits. This situation can create security problems if systems running SunOS 4.0 and compatible releases are used in an environment where large UIDs and GIDs are being used. Systems running SunOS 4.0 and compatible releases require a patch to avoid this problem.

TABLE 1–2 Interoperability Issues for UIDs or GIDs Over 60000 *(Continued)*

Category	Product or Command	Issue
Name service interoperability	NIS name service and file-based name service	Users with UIDs greater than 60000 can log in or use the `su` command on systems running the Solaris 2.5 (and compatible releases). However, their UIDs and GIDs will be set to 60001 (`nobody`).

The following table describes UID and GUI limitations.

TABLE 1–3 Large UID and GID Limitation Summary

UID or GID	Limitations
60003 or greater	Users who log in to systems running Solaris 2.5 (and compatible releases) and the NIS or files name service get a UID and GID of `nobody`.
65535 or greater	■ Systems running Solaris 2.5 (and compatible releases) with the NFS version 2 software truncate UIDs to 16 bits, creating possible security problems. ■ Users who use the `cpio` command with the default archive format to copy a file see an error message for each file. And, the UIDs and GIDs are set to `nobody` in the archive. ■ x86 based systems: Users that run SVR3-compatible applications will probably see `EOVERFLOW` return codes from system calls. ■ x86 based systems: If users attempt to create a file or directory on a mounted System V file system, the System V file system returns an `EOVERFLOW` error.
100000 or greater	The `ps -l` command displays a maximum five-digit UID. So, the printed column won't be aligned when it includes a UID or GID larger than 99999.
262144 or greater	Users who use the `cpio` command with the `-H odc` format or the `pax -x cpio` command to copy files see an error message returned for each file. And, the UIDs and GIDs are set to `nobody` in the archive.
1000000 or greater	Users who use the `ar` command have their UIDs and GIDs set to `nobody` in the archive.
2097152 or greater	Users who use the `tar` command, the `cpio -H ustar` command, or the `pax -x tar` command have their UIDs and GIDs set to `nobody`.

UNIX Groups

A *group* is a collection of users who can share files and other system resources. For example, users who working on the same project could be formed into a group. A group is traditionally known as a UNIX group.

Each group must have a name, a group identification (GID) number, and a list of user names that belong to the group. A GID number identifies the group internally to the system.

The two types of groups that a user can belong to are as follows:

- **Primary group –** Specifies a group that the operating system assigns to files that are created by the user. Each user must belong to a primary group.
- **Secondary groups –** Specifies one or more groups to which a user also belongs. Users can belong to up to 15 secondary groups.

For detailed guidelines on setting up group names, see "Guidelines for Assigning User Names, User IDs, and Group IDs" on page 24.

Sometimes, a user's secondary group is not important. For example, ownership of files reflect the primary group, not any secondary groups. Other applications, however, might rely on a user's secondary group memberships. For example, a user has to be a member of the `sysadmin` group (group 14) to use the Admintool software in previous Solaris releases. However, it doesn't matter if group 14 is his or her current primary group.

The `groups` command lists the groups that a user belongs to. A user can have only one primary group at a time. However, a user can temporarily change the user's primary group, with the `newgrp` command, to any other group in which the user is a member.

When adding a user account, you must assign a primary group for a user or accept the default group, `staff` (group 10). The primary group should already exist. If the primary group does not exist, specify the group by a GID number. User names are not added to primary groups. If user names were added to primary groups, the list might become too long. Before you can assign users to a new secondary group, you must create the group and assign it a GID number.

Groups can be local to a system or managed through a name service. To simplify group administration, you should use a name service such as NIS or a directory service such as LDAP. These services enable you to centrally manage group memberships.

User Passwords

You can specify a password for a user when you add the user. Or, you can force the user to specify a password when the user first logs in.

User passwords must comply with the following syntax:

- Password length must at least match the value identified by the `PASSLENGTH` variable in the `/etc/default/passwd` file. By default, `PASSLENGTH` is set to 6.
- The first 6 characters of the password must contain at least two alphabetic characters and have at least one numeric or special character.

Although user names are publicly known, passwords must be kept secret and known only to users. Each user account should be assigned a password. The password can be a combination of six to eight letters, numbers, or special characters.

To make your computer systems more secure, users should change their passwords periodically. For a high level of security, you should require users to change their passwords every six weeks. Once every three months is adequate for lower levels of security. System administration logins (such as `root` and `sys`) should be changed monthly, or whenever a person who knows the root password leaves the company or is reassigned.

Many breaches of computer security involve guessing a legitimate user's password. You should make sure that users avoid using proper nouns, names, login names, and other passwords that a person might guess just by knowing something about the user.

Good choices for passwords include the following:

- Phrases (`beammeup`).
- Nonsense words made up of the first letters of every word in a phrase. For example, `swotrb` for `SomeWhere Over The RainBow`.
- Words with numbers or symbols substituted for letters. For example, `sn00py` for snoopy.

Do not use these choices for passwords:

- Your name (spelled forwards, backwards, or jumbled)
- Names of family members or pets
- Car license numbers
- Telephone numbers
- Social Security numbers
- Employee numbers
- Words related to a hobby or interest
- Seasonal themes, such as Santa in December
- Any word in the dictionary

For task-related information, see "How to Add a User" on page 45.

Home Directories

The home directory is the portion of a file system allocated to a user for storing private files. The amount of space you allocate for a home directory depends on the kinds of files the user creates, their size, and the number of files that are created.

A home directory can be located either on the user's local system or on a remote file server. In either case, by convention the home directory should be created as `/export/home/`*username*. For a large site, you should store home directories on a server. Use a separate file system for each `/export/home`*n* directory to facilitate backing up and restoring home directories. For example, `/export/home1`, `/export/home2`.

Regardless of where their home directory is located, users usually access their home directories through a mount point named `/home/`*username*. When AutoFS is used to mount home directories, you are not permitted to create any directories under the `/home` mount point on any system. The system recognizes the special status of `/home` when AutoFS is active. For more information about automounting home directories, see "Task Overview for Autofs Administration" in *System Administration Guide: Network Services*.

To use the home directory anywhere on the network, you should always refer to the home directory as `$HOME`, not as `/export/home/`*username*. The latter is machine-specific. In addition, any symbolic links created in a user's home directory should use relative paths (for example, `../../../x/y/x`) so that the links are valid no matter where the home directory is mounted.

Name Services

If you are managing user accounts for a large site, you might want to consider using a name or directory service such as LDAP, or NIS . A name or directory service enables you to store user account information in a centralized manner instead of storing user account information in every system's `/etc` files. When you use a name or directory service for user accounts, users can move from system to system using the same user account without having site-wide user account information duplicated on every system. Using a name or directory service also promotes centralized and consistent user account information.

User's Work Environment

Besides having a home directory to create and store files, users need an environment that gives them access to the tools and resources they need to do their work. When a user logs in to a system, the user's work environment is determined by initialization files. These files are defined by the user's startup shell, which can vary, depending on the release.

A good strategy for managing the user's work environment is to provide customized user initialization files, such as`.bash_profile`, `.bash_login`, `.kshrc`, or `.profile`, in the user's home directory.

Note – Do not use system initialization files, such as `/etc/profile` or `/etc/.login`, to manage a user's work environment. These files reside locally on systems and are not centrally administered. For example, if AutoFS is used to mount the user's home directory from any system on the network, you would have to modify the system initialization files on each system to ensure a consistent environment whenever a user moved from system to system.

For detailed information about customizing user initialization files for users, see "Customizing a User's Work Environment" on page 30.

Another way to customize user accounts is through role-based access control (RBAC). See "Role-Based Access Control (Overview)" in *System Administration Guide: Security Services* for more information.

Guidelines for Assigning User Names, User IDs, and Group IDs

User names, UIDs, and GIDs should be unique within your organization, which could span multiple domains.

Keep the following guidelines in mind when creating user or role names, UIDs, and GIDs:

- **User names –** Should contain from two to eight letters and numerals. The first character should be a letter. At least one character should be a lowercase letter.

  Note – Even though user names can include a period (.), underscore (_), or hyphen (-), using these characters is not recommended because they can cause problems with some software products.

- **System accounts –** Do not use any of the user names, UIDs, or GIDs that are contained in the default `/etc/passwd` and `/etc/group` files. Do not use the UIDs and GIDs, 0-99. These numbers are reserved for allocation by Oracle Solaris and should not be used by anyone. Note that this restriction also applies to numbers not currently in use.

  For example, `gdm` is the reserved user name and group name for the GNOME Display Manager daemon and should not be used for another user. For a complete listing of the default `/etc/passwd` and `/etc/group` entries, see Table 1–4 and Table 1–5.

  The `nobody` and `nobody4` accounts should never be used for running processes. These two accounts are reserved for use by NFS. Use of these accounts for running processes could lead to unexpected security risks. Processes that need to run as a non-root user should use the `daemon` or `noaccess` accounts.
- **System account configuration** – The configuration of the default system accounts should never be changed. This includes changing the login shell of a system account that is currently locked. The only exception to this rule is the setting of a password and password aging parameters for the root account.

Where User Account and Group Information Is Stored

Depending on your site policy, user account and group information can be stored in your local system's `/etc` files or in a name or directory service as follows:

- The NIS name service information is stored in maps.
- The LDAP directory service information is stored in indexed database files.

Note – To avoid confusion, the location of the user account and group information is generically referred to as a *file* rather than as a *database*, *table*, or *map*.

Most user account information is stored in the `passwd` file. Password information is stored as follows:

- In the `passwd` file when you are using NIS
- In the `/etc/shadow` file when you are using `/etc` files
- In the `people` container when you are using LDAP

Password aging is available when you are using LDAP, but not NIS.

Group information is stored in the `group` file for NIS, and files. For LDAP, group information is stored in the `group` container.

Fields in the `passwd` File

The fields in the `passwd` file are separated by colons and contain the following information:

username:*password*:*uid*:*gid*:*comment*:*home-directory*:*login-shell*

For example:

```
kryten:x:101:100:Kryten Series 4000 Mechanoid:/export/home/kryten:/bin/csh
```

For a complete description of the fields in the `passwd` file, see the `passwd(1)` man page.

Default `passwd` File

The default `passwd` file contains entries for standard daemons. Daemons are processes that are usually started at boot time to perform some system-wide task, such as printing, network administration, or port monitoring.

```
root:x:0:0:Super-User:/root:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
dladm:x:15:3:Datalink Admin:/:
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/var/lib/gdm:
```

```
zfssnap:x:51:12:ZFS Automatic Snapshots Reserved UID:/:/usr/bin/pfsh
upnp:x:52:52:UPnP Server Reserved UID:/var/coherence:/bin/ksh
xvm:x:60:60:xVM User:/:
mysql:x:70:70:MySQL Reserved UID:/:
openldap:x:75:75:OpenLDAP User:/:
webservd:x:80:80:WebServer Reserved UID:/:
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
svctag:x:95:12:Service Tag UID:/:
unknown:x:96:96:Unknown Remote UID:/:
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:

root:x:0:0:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/:
webservd:x:80:80:WebServer Reserved UID:/:
postgres:x:90:90:PostgreSQL Reserved UID:/:/usr/bin/pfksh
unknown:x:96:96:Unknown Remote UID:/:
svctag:x:95:12:Service Tag UID:/:
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
```

TABLE 1–4 Default passwd File Entries

User Name	User ID	Description
`root`	0	Superuser account
`daemon`	1	Umbrella system daemon associated with routine system tasks
`bin`	2	Administrative daemon associated with running system binaries to perform some routine system task
`sys`	3	Administrative daemon associated with system logging or updating files in temporary directories
`adm`	4	Administrative daemon associated with system logging
`lp`	71	Line printer daemon
`uucp`	5	Daemon associated with uucp functions
`nuucp`	6	Another daemon associated with uucp functions
`dladm`	15	Account reserved for datalink administration.

TABLE 1–4 Default passwd File Entries *(Continued)*

User Name	User ID	Description
zfssnap	51	Account reserved for automatic snapshots.
upnp	52	Account reserved for UPnP server.
xvm	60	Reserved for xVM user.
openldap	75	Reserved for OpenLDAP user.
smmsp	25	Sendmail message submission program daemon
webservd	80	Account reserved for WebServer access
postgres	90	Account reserved for PostgresSQL access
svctag	95	Service Tag Registry access
gdm	50	GNOME Display Manager daemon
listen	37	Network listener daemon
nobody	60001	Account reserved for anonymous NFS access.
noaccess	60002	Assigned to a user or a process that needs access to a system through some application but without actually logging in
nobody4	65534	SunOS 4.0 or 4.1 version of the nobody user account
unknown	96	Account reserved for unmappable remote users in NFSv4 ACLs

Fields in the shadow File

The fields in the shadow file are separated by colons and contain the following information:

username:*password*:*lastchg*:*min*:*max*:*warn*:*inactive*:*expire*

Note – In the current Oracle Solaris release, the default password hashing algorithm has been changed to SHA256. The password hash for the user is similar to the following:

```
$5$cgQk2iUy$AhHtVGx5Qd0.W3NCKjikb8.KhOiA4DpxsW55sP0UnYD
```

For a complete description of the fields in the shadow file, see the shadow(4) and crypt(1) man pages.

Fields in the group File

The fields in the group file are separated by colons and contain the following information:

group-name:*group-password*:*gid*:*user-list*

For example:

```
bin::2:root,bin,daemon
```

For a complete description of the fields in the group file, see the group(4) man page.

Default group File

The default group file contains the following system groups that support some system-wide task, such as printing, network administration, or electronic mail. Many of these groups having corresponding entries in the passwd file.

```
root::0:
other::1:root
bin::2:root,daemon
sys::3:root,bin,adm
adm::4:root,daemon
uucp::5:root
mail::6:root
tty::7:root,adm
lp::8:root,adm
nuucp::9:root
staff::10:
daemon::12:root
sysadmin::14:
games::20:
smmsp::25:
gdm::50:
upnp::52:
xvm::60:
mysql::70:
openldap::75:
webservd::80:
postgres::90:
slocate::95:
unknown::96:
nobody::60001:
noaccess::60002:
nogroup::65534:
```

TABLE 1–5 Default group File Entries

Group Name	Group ID	Description
root	0	Superuser group

TABLE 1–5 Default `group` File Entries *(Continued)*

Group Name	Group ID	Description
`other`	`1`	Optional group
`bin`	`2`	Administrative group associated with running system binaries
`sys`	`3`	Administrative group associated with system logging or temporary directories
`adm`	`4`	Administrative group associated with system logging
`uucp`	`5`	Group associated with `uucp` functions
`mail`	`6`	Electronic mail group
`tty`	`7`	Group associated with `tty` devices
`lp`	`8`	Line printer group
`nuucp`	`9`	Group associated with `uucp` functions
`staff`	`10`	General administrative group.
`daemon`	`12`	Group associated with routine system tasks
`sysadmin`	`14`	Administrative group that is useful for system administrators
`smmsp`	`25`	Daemon for Sendmail message submission program
`gdm`	`50`	Group reserved for the GNOME Display Manager daemon
`webservd`	`80`	Group reserved for WebServer access
`postgres`	`90`	Group reserved for PostgresSQL access
`nobody`	`60001`	Group assigned for anonymous NFS access
`noaccess`	`60002`	Group assigned to a user or a process that needs access to a system through some application but without actually logging in
`nogroup`	`65534`	Group assigned to a user who is not a member of a known group
`unknown`	`96`	Group reserved for unmappable remote groups in NFSv4 ACLs

Tools for User Account and Group Account Management

Note – In this Oracle Solaris release, the Solaris Management Console and all its equivalent command-line tools are no longer supported.

The following table describes the available tools for user account and group management.

TABLE 1–6 Tools for User Account and Group Management

Tool Name	Description	For More Information
`useradd`, `groupadd`, `roleadd`;	The commands that are used to add users, groups, and roles.	"How to Add a User" on page 45 "How to Add a Group" on page 47 "How to Add a Role" on page 46
`usermod`, `groupmod`, `rolemod`	The commands that are used to modify users, groups, and roles.	*System Administration Guide: Security Services*
`userdel`, `groupdel`, `roledel`	The commands that are used to delete users, groups, and roles.	*System Administration Guide: Security Services*

Customizing a User's Work Environment

Part of setting up a user's home directory is providing user initialization files for the user's login shell. A *user initialization file* is a shell script that sets up a work environment for a user after the user logs in to a system. Basically, you can perform any task in a user initialization file that you can do in a shell script. However, a user initialization file's primary job is to define the characteristics of a user's work environment, such as a user's search path, environment variables, and windowing environment. Each login shell has its own user initialization file, or files, which are listed in the following table. Note that the default user initialization file for both the bash and `ksh93` shells is `/etc/skel/local.profile`.

TABLE 1–7 Bash and `ksh93` User Initialization Files

Shell	User Initialization File	Purpose
bash	`$HOME/.bash_profile`	Defines the user's environment at login
	`$HOME/.bash_login`	
	`$HOME/.profile`	
ksh93	`/etc/profile`	Defines the user's environment at login
	`$HOME/.profile`	
	`$ENV`	Defines user's environment at login in the file and is specified by the Korn shell's `ENV` environment variable

You can use these files as a starting point and then modify them to create a standard set of files that provide the work environment common to all users. You can also modify these files to provide the working environment for different types of users.

For step-by-step instructions on how to create sets of user initialization files for different types of users, see "How to Customize User Initialization Files" on page 43.

Using Site Initialization Files

The user initialization files can be customized by both the administrator and the user. This important task can be accomplished with centrally located and globally distributed user initialization files that are called, *site initialization files*. Site initialization files enable you to continually introduce new functionality to the user's work environment, while enabling the user to customize the user's initialization file.

When you reference a site initialization file in a user initialization file, all updates to the site initialization file are automatically reflected when the user logs in to the system or when a user starts a new shell. Site initialization files are designed for you to distribute site-wide changes to users' work environments that you did not anticipate when you added the users.

You can customize a site initialization file the same way that you customize a user initialization file. These files typically reside on a server, or set of servers, and appear as the first statement in a user initialization file. Also, each site initialization file must be the same type of shell script as the user initialization file that references it.

To reference a site initialization file in a bash or `ksh93` user initialization file, place a line at the beginning of the user initialization file similar to the following line:

```
. /net/machine-name/export/site-files/site-init-file
```

Avoiding Local System References

Do not add specific references to the local system in the user initialization file. The instructions in a user initialization file should be valid, regardless of which system the user logs into.

For example:

- To make a user's home directory available anywhere on the network, always refer to the home directory with the variable `$HOME`. For example, use `$HOME/bin` instead of `/export/home/`*username*`/bin`. The `$HOME` variable works when the user logs in to another system, and the home directories are auto-mounted.
- To access files on a local disk, use global path names, such as `/net/`*system-name/directory-name*. Any directory referenced by `/net/`*system-name* can be mounted automatically on any system on which the user logs in, assuming the system is running AutoFS.

Shell Features

The user account that is created when you install the Oracle Solaris release is assigned the GNU Bourne-Again Shell (bash) by default. The standard system shell, `bin/sh`, is now the Korn Shell

93 (ksh93). Both the bash and ksh93 shells feature command-line editing, which means you can edit commands before executing them. To change to a different shell, type the path of the shell that you want to use. To exit a shell, type `exit`.

The following table describes the shell options that are supported in this release.

TABLE 1–8 Basic Shell Features in the Oracle Solaris Release

Shell	Path	Comments
Bourne-Again Shell (bash)	`/usr/bin/bash`	Default shell for users that are created by an installer, as well as the `root` role
Korn Shell	`/usr/bin/ksh`	`ksh93` is the default shell in this Oracle Solaris release
C Shell and enhanced C Shell	`/usr/bin/csh` and `/usr/bin/tcsh`	C Shell and enhanced C Shell
POSIX-compliant Shell	`/usr/xpg4/bin/sh`	POSIX-compliant shell
Z Shell	`/usr/bin/zsh`	Z Shell

Note – The Z Shell (`zsh`) and the enhanced C Shell (`tsch`) are not installed on your system by default. To use either of these shells, you must first install the required software packages.

Bash and ksh93 Shell History

Both the `bash` and `ksh93` shells record a history of all of the commands that you run. This history is kept on a per user basis, which means history is persistent between login sessions and is representative of all your login sessions.

For example, if you are in a bash shell, to see the complete history of commands you have run, you would type:

```
$ history
1 ls
2 ls -a
3 pwd
4 whoami
.
.
.
```

To display a number of previous commands, include an integer in the command:

```
$ history 2
12 date
13 history
```

For more information, see the Bash and ksh93 Shell History man page.

Bash and `ksh93` Shell Environment Variables

The bash and `ksh93` shells store special variable information that is known to the shell as an*environment variable*. To view a complete list of the current environment variables for the bash shell, use the `declare` command as follows:

```
$ declare
BASH=/usr/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_LINEND=()
BASH_SOuRCE=()
BASH_VERSINFO=([0]=’’3’’ [1]=’’2’’ [2]=’’25’’ [3]=’’1’’
[4]=’’release’’ [5]’’
.
.
.
```

For the `ksh93` shell, use the `set` command, which is the bash shell's `declare` command equivalent:

```
$ set
  COLUMNS=80
  ENV=’$HOME/.kshrc’
  FCEDIT=/bin/ed
  HISTCMD=3
  HZ=’’
  IFS=$’ \t\n’
  KSH_VERSION=.sh.version
  LANG=C
  LINENO=1
  .
  .
  .
```

To print environment variables for either shell, use the `echo` or `printf` command. For example:

```
$ echo $SHELL
/bin/bash
$ printf ’’$PATH/n’’
/usr/bin
```

Note – Environment variables do not persist between sessions. To set up environment variables that remain consistent between logins, you must make the changes in the `.bashrc` file.

A shell can have two types of variables:

Environment variables	Specifies variables that are exported to all processes that are spawned by the shell. The `export` command is used to export a variable. For example: `export VARIABLE=value` These settings can be displayed by using the `env` command. A subset of environment variables, such as `PATH`, affects the behavior of the shell itself.
Shell (local) variables	Specifies variables that affect only the current shell. In a user initialization file, you can customize a user's shell environment by changing the values of the predefined variables or by specifying additional variables.

The following table provides more details about the shell and environment variables that are available in the Oracle Solaris release.

TABLE 1–9 Shell and Environment Variable Descriptions

Variable	Description
`CDPATH`	Sets a variable used by the `cd` command. If the target directory of the `cd` command is specified as a relative path name, the `cd` command first looks for the target directory in the current directory (`.`). If the target is not found, the path names listed in the `CDPATH` variable are searched consecutively until the target directory is found and the directory change is completed. If the target directory is not found, the current working directory is left unmodified. For example, the `CDPATH` variable is set to `/home/jean`, and two directories exist under `/home/jean`, `bin`, and `rje`. If you are in the `/home/jean/bin` directory and type `cd rje`, you change directories to `/home/jean/rje`, even though you do not specify a full path.
`HOME`	Sets the path to the user's home directory.
`LANG`	Sets the locale.
`LOGNAME`	Defines the name of the user currently logged in. The default value of `LOGNAME` is set automatically by the login program to the user name specified in the `passwd` file. You should only need to refer to, not reset, this variable.
`MAIL`	Sets the path to the user's mailbox.
`MANPATH`	Sets the hierarchies of man pages that are available.

TABLE 1–9 Shell and Environment Variable Descriptions *(Continued)*

Variable	Description
PATH	Specifies, in order, the directories that the shell searches to find the program to run when the user types a command. If the directory is not in the search path, users must type the complete path name of a command. As part of the login process, the default PATH is automatically defined and set as specified in .profile. The order of the search path is important. When identical commands exist in different locations, the first command found with that name is used. For example, suppose that PATH is defined in the shell syntax as PATH=/bin:/usr/bin:/usr/sbin:$HOME/bin and a file named sample resides in both /usr/bin and /home/jean/bin. If the user types the command sample without specifying its full path name, the version found in /usr/bin is used.
PS1	Defines the shell prompt for the bash or ksh93 shell.
SHELL	Sets the default shell used by make, vi, and other tools.
TERMINFO	Names a directory where an alternate terminfo database is stored. Use the TERMINFO variable in either the /etc/profile or /etc/.login file. For more information, see the terminfo(4)man page. When the TERMINFO environment variable is set, the system first checks the TERMINFO path defined by the user. If the system does not find a definition for a terminal in the TERMINFO directory defined by the user, it searches the default directory, /usr/share/lib/terminfo, for a definition. If the system does not find a definition in either location, the terminal is identified as “dumb.”
TERM	Defines the terminal. This variable should be reset in either the /etc/profile or /etc/.login file. When the user invokes an editor, the system looks for a file with the same name that is defined in this environment variable. The system searches the directory referenced by TERMINFO to determine the terminal characteristics.
TZ	Sets the time zone. The time zone is used to display dates, for example, in the ls -l command. If TZ is not set in the user's environment, the system setting is used. Otherwise, Greenwich Mean Time is used.

Customizing the Bash Shell

To customize your bash shell, add the information to the .bashrc file that is located in your home directory. The initial user that is created when you install Oracle Solaris has a .bashrc file that sets the PATH, MANPATH, and command prompt. For more information, see the bash(1) man page.

About the `MANPATH` Environment Variable

The `MANPATH` environment variable is similar to the PATH variable. MANPATH specifies where the man command looks for reference manual pages. The `MANPATH` in the user that is created by an installer looks like the following:

```
$ echo $MANPATH
/usr/gnu/share/man:/usr/shar/man:/usr/X1/share/man
```

The `PATH` Variable

When the user executes a command by using the full path, the shell uses that path to find the command. However, when users specify only a command name, the shell searches the directories for the command in the order specified by the `PATH` variable. If the command is found in one of the directories, the shell executes the command.

A default path is set by the system. However, most users modify it to add other command directories. Many user problems related to setting up the environment and accessing the correct version of a command or a tool can be traced to incorrectly defined paths.

Setting Path Guidelines

Here are some guidelines for setting up efficient `PATH` variables:

- If security is not a concern, put the current working directory (.) first in the path. However, including the current working directory in the path poses a security risk that you might want to avoid, especially for superuser.
- Keep the search path as short as possible. The shell searches each directory in the path. If a command is not found, long searches can slow down system performance.
- The search path is read from left to right, so you should put directories for commonly used commands at the beginning of the path.
- Make sure that directories are not duplicated in the path.
- Avoid searching large directories, if possible. Put large directories at the end of the path.
- Put local directories before NFS mounted directories to lessen the chance of “hanging” when the NFS server does not respond. This strategy also reduces unnecessary network traffic.

Setting a User's Default Path

To set the user's default path in either a bash or `ksh93` user initialization file, you would add the following:

```
PATH=.:/usr/bin:/$HOME/bin:/net/glrr/files1/bin
export PATH
```

Locale Variables

The `LANG` and `LC` environment variables specify the locale-specific conversions and conventions for the shell. These conversions and conventions include time zones, collation orders, and formats of dates, time, currency, and numbers. In addition, you can use the `stty` command in a user initialization file to indicate whether the terminal session will support multibyte characters.

The `LANG` variable sets all possible conversions and conventions for the given locale. You can set various aspects of localization separately through these `LC` variables: `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_NUMERIC`, `LC_MONETARY`, and `LC_TIME`.

The following table describes some of the values for the `LANG` and `LC` environment variables.

TABLE 1–10 Values for `LANG` and `LC` Variables

Value	Locale
`de_DE.ISO8859-1`	`German`
`en_US.UTF-8`	`American English` (UTF-8)
`es_ES.ISO8859-1`	`Spanish`
`fr_FR.ISO8859-1`	`French`
`it_IT.ISO8859-1`	`Italian`
`ja_JP.eucJP`	`Japanese` (EUC)
`ko_KR.EUC`	`Korean` (EUC)
`sv_SE.ISO8859-1`	`Swedish`
`zh_CN.EUC`	`Simplified Chinese` (EUC)
`zh_TW.EUC`	`Traditional Chinese` (EUC)

For more information on supported locales, see the *International Language Environments Guide*.

EXAMPLE 1–1 Setting the Locale Using the `LANG` Variables

In a Bourne-shell or Korn-shell user initialization file, you would add the following:

```
LANG=de_DE.ISO8859-1; export LANG
```

Default File Permissions (`umask`)

When you create a file or directory, the default file permissions assigned to the file or directory are controlled by the *user mask*. The user mask is set by the `umask` command in a user initialization file. You can display the current value of the user mask by typing `umask` and pressing Return.

The user mask contains the following octal values:

- The first digit sets permissions for the user
- The second digit sets permissions for group
- The third digit sets permissions for other, also referred to as `world`

Note that if the first digit is zero, it is not displayed. For example, if the user mask is set to 022, 22 is displayed.

To determine the `umask` value that you want to set, subtract the value of the permissions you want from 666 (for a file) or 777 (for a directory). The remainder is the value to use with the `umask` command. For example, suppose you want to change the default mode for files to 644 (`rw-r--r--`). The difference between 666 and 644 is 022, which is the value you would use as an argument to the `umask` command.

You can also determine the `umask` value you want to set by using the following table. This table shows the file and directory permissions that are created for each of the octal values of `umask`.

TABLE 1–11 Permissions for `umask` Values

`umask` Octal Value	**File Permissions**	**Directory Permissions**
`0`	`rw-`	`rwx`
`1`	`rw-`	`rw-`
`2`	`r--`	`r-x`
`3`	`r--`	`r--`
`4`	`-w-`	`-wx`
`5`	`-w-`	`-w-`
`6`	`--x`	`--x`
`7`	`---` (none)	`---` (none)

The following line in a user initialization file sets the default file permissions to `rw-rw-rw-`.

```
umask 000
```

Customizing a User Initialization File

The following is an example of the `.profile` user initialization file. You can use this file to customize your own user initialization files. This example uses system names and paths that you will need to modify for your particular site.

EXAMPLE 1–2 The `.profile` File

```
(Line 1) PATH=$PATH:$HOME/bin:/usr/local/bin:/usr/ccs/bin:.
(Line 2) MAIL=/var/mail/$LOGNAME
(Line 3) NNTPSERVER=server1
(Line 4) MANPATH=/usr/share/man:/usr/local/man
(Line 5) PRINTER=printer1
(Line 6) umask 022
(Line 7) export PATH MAIL NNTPSERVER MANPATH PRINTER
```

1. Defines the user's shell search path.
2. Defines the path to the user's mail file.
3. Defines the user's Usenet news server.
4. Defines the user's search path for man pages.
5. Defines the user's default printer.
6. Sets the user's default file creation permissions.
7. Sets the listed environment variables.

CHAPTER 2

Managing User Accounts and Groups (Tasks)

This chapter describes how to set up and maintain user accounts and groups.

For background information about managing user accounts and groups, see Chapter 1, "Managing User Accounts and Groups (Overview)."

Setting Up and Administering User Accounts (Task Map)

Task	Description	For Instructions
Gather user information.	Use a standard form to gather user information to help you keep user information organized.	"Gathering User Information" on page 42
Customize user initialization files.	You can set up user initialization files, so that you can provide new users with consistent environments.	"How to Customize User Initialization Files" on page 43
Set up user account defaults.	Before adding a user, you must correctly set up the user account defaults.	"How to Set Account Defaults" on page 44
Create a user account.	Using the account defaults that you set up, create a local user by using the `useradd` command.	"How to Add a User" on page 45
Delete a user account.	You can delete a user account by using the `userdel` command.	"How to Delete a User" on page 45
Create a role to perform an administrative task.	Using the account defaults that you set up, create a local role, so that the user can perform a specific administrative command or task.	"How to Add a Role" on page 46

Task	Description	For Instructions
Create a group.	To create a new group, use the `groupadd` command.	"How to Add a Group" on page 47
Add security attributes to a user account.	After you set up a local user account, you can add the required security attributes.	"How to Change the RBAC Properties of a User" in *System Administration Guide: Security Services*
Share a user's home directory.	You must share the user's home directory, so that the directory can be remotely mounted from the user's system.	"How to Share a User's Home Directory" on page 48
Manually mount a user's home directory.	Typically, you do not need to manually mount a user's home directory. A home directory that is created as a ZFS file system is mounted automatically when it is created and also at boot time from the Service Management Facility (SMF) local file system service.	"Manually Mounting a User's Home Directory." on page 48

Setting Up User Accounts

Gathering User Information

You can create a form such as the following to gather information about users before adding their accounts.

Item	Description
User Name:	
Role Name:	
Profiles or Authorizations:	
UID:	
Primary Group:	
Secondary Groups:	
Comment:	
Default Shell:	
Password Status and Aging:	

Item	Description
Home Directory Path Name:	
Mounting Method:	
Permissions on Home Directory:	
Mail Server:	
Department Name:	
Department Administrator:	
Manager:	
Employee Name:	
Employee Title:	
Employee Status:	
Employee Number:	
Start Date:	
Add to These Mail Aliases:	
Desktop System Name:	

▼ How to Customize User Initialization Files

1 **Become the root user.**

```
$ su -
Password:
#
```

Note – This method works whether `root` is a user or a role.

---

2 **Create a skeleton directory for each type of user.**

`# mkdir /`*shared-dir*`/skel/`*user-type*

shared-dir	The name of a directory that is available to other systems on the network.
user-type	The name of a directory to store initialization files for a type of user.

3 **Copy the default user initialization files into the directories that you created for different types of users.**

4 **Edit the user initialization files for each user type and customize them based on your site's needs.**

For a detailed description on the ways to customize the user initialization files, see "Customizing a User's Work Environment" on page 30.

5 **Set the permissions for the user initialization files.**

```
# chmod 744 /shared-dir/skel/user-type/.*
```

6 **Verify that the permissions for the user initialization files are correct.**

```
# ls -la /shared-dir/skel/*
```

▼ How to Set Account Defaults

1 **Become the root user..**

2 **List the user defaults.**

```
# useradd -D
group=staff,10  project=default,3  basedir=/home
skel=/etc/skel  shell=/bin/sh  inactive=0
expire=  auths=  profiles=  roles=  limitpriv=
defaultpriv=  lock_after_retries=
```

3 **Change the default home directory.**

```
# useradd -D -b /export/home
```

For the command options, see the `roleadd(1M)` man page.

4 **List the new user defaults.**

```
# useradd -D
group=staff,10  project=default,3  basedir=/export/home
skel=/etc/skel  shell=/bin/sh  inactive=0
expire=  auths=  profiles=  roles=  limitpriv=
defaultpriv=  lock_after_retries=
```

Example 2–1 Changing the Account Defaults for All Roles

In this example, the administrator has customized a `roles` directory . The administrator changes the default home directory and skeleton directory for all roles. The

```
# roleadd -D
group=other,1  project=default,3  basedir=/home
skel=/etc/skel  shell=/bin/pfsh  inactive=0
expire=  auths=  profiles=All  limitpriv=
defaultpriv=  lock_after_retries=
# roleadd -D -b /export/home -k /etc/skel/roles
# roleadd -D
```

```
group=staff,10  project=default,3  basedir=/export/home
skel=/etc/skel/roles  shell=/bin/sh  inactive=0
expire=  auths=  profiles=  roles=  limitpriv=
defaultpriv=  lock_after_retries=
```

Future uses of the **roleadd** command create home directories in `/export/home`, and populate the roles' environment from the `/etc/skel/roles` directory.

▼ How to Add a User

1 **Become the root user.**

2 **Create a local user.**

Use the defaults that you modified in "How to Set Account Defaults" on page 44.

```
# useradd -m username
```

`useradd`	Creates an account for the specified user.
`-m`	Creates a local home directory on the system for the specified user.

Note – The account is locked until you assign the user a password.

---

3 **Assign the user a password.**

```
$ passwd username
New password:      Type user password
Re-enter new password:      Retype password
```

For more command options, see the `useradd`(1M) and `passwd`(1) man pages.

▼ How to Delete a User

1 **Become the root user.**

```
$ su -
Password:
#
```

Note – This method works whether `root` is a user or a role.

---

2 **Archive the user's home directory.**

3 Run one of the following commands:

- **If the user has a local home directory, delete the user and the home directory.**

  ```
  # userdel -r username
  ```

`usesrdel`	Deletes the account of the specified user.
`-r`	Removes the account from the system.

- **Otherwise, delete the user only.**

  ```
  # userdel username
  ```

  You must manually delete the user's home directory on the remote server.

For a full list of command options, see the `userdel(1M)` man page.

▼ How to Add a Role

1 Become the root user.

2 Create a local role.

Use the defaults that you modified in Example 2–1.

```
# roleadd -m rolename
```

`roleadd`	Administers a new role on the system.
`-m`	Creates the new role's home directory, if one does not already exist.

The account is locked until you assign the role a password.

3 Assign the role a password.

```
$ passwd rolename
New password:        Type role password
Re-enter new password:       Retype password
```

For more command options, see the `roleadd(1M)` and `passwd(1)` man pages.

Example 2–2 Creating a Role That Mounts a Remote Home Directory

In this example, a central server contains the home directories of users and roles. The administrator creates a role, but does not specify the home directory.

```
# roleadd -D
group=other,1  project=default,3  basedir=/export/home
skel=/etc/skel/roles  shell=/bin/pfsh  inactive=0
expire=  auths=  profiles=All  limitpriv=
defaultpriv=  lock_after_retries=
```

```
# roleadd audcontrol
```

Because no home directory was specified, no local home directory exists.

```
# ls /export/home
jdoe/ kdoe/ ldoe/
```

▼ How to Add a Group

1 Become the root user.

2 List the existing groups.

```
# cat /etc/group
root::0:
other::1:root
bin::2:root,daemon
sys::3:root,bin,adm
adm::4:root,daemon
uucp::5:root
mail::6:root
tty::7:root,adm
lp::8:root,adm
nuucp::9:root
staff::10:
daemon::12:root
sysadmin::14:
...
unknown::96:
nobody::60001:
noaccess::60002:
nogroup::65534:
pkg5srv::97:
```

3 Create a new group.

```
$ groupadd -g 18 exadata
```

`groupadd`	Creates a new group definition on the system by adding the appropriate entry to the `/etc/group` file.
`-g`	Assigns the group ID for the new group.

For more information, see the `groupadd`(1M) man page.

Example 2–3 Adding a Group and User With the `groupadd` and `useradd` Commands

The following example shows how to use the `groupadd` and `useradd` commands to add the group `scutters` and the user `scutter1` to files on the local system. These commands cannot be used to manage users in a name service environment.

```
# groupadd -g 102 scutters
# useradd -u 1003 -g 102 -d /export/home/scutter1 -s /bin/csh \
-c "Scutter 1" -m -k /etc/skel scutter1
64 blocks
```

For more information, see the groupadd(1M) and useradd(1M) man pages.

▼ How to Share a User's Home Directory

The following procedure shows how to share a user's home directory. Prior to this task, the user's home directory was created as a ZFS file system, as follows:

```
# zfs create -p -o mountpoint=/export/home/ripley rpool/export/home/username
```

1. **Become the root user.**

2. **Enable the `sharenfs` property on the file system that is to be shared.**

```
# share
-               /export/home/username   rw   ""
```

Manually Mounting a User's Home Directory.

User accounts that are created as ZFS file systems do not typically need to be manually mounted. With ZFS, file systems are automounted when they are created and then mounted at boot time from the SMF local file system service.

When creating user accounts, make sure that home directories are set up as they are in the name service, at `/home/`*username*. Then, make sure that the `auto_home` map indicates the NFS path to the user's home directory. For task-related information, see "Task Overview for Autofs Administration" in *System Administration Guide: Network Services*.

If you need to manually mount a user's home directory, use the `zfs mount` command. For example:

```
# zfs mount tank/home/username
```

Note – Make sure that the user's home directory is shared. For more information, see "How to Share a User's Home Directory" on page 48.

CHAPTER 3

Introduction to Shutting Down and Booting a System

Oracle Solaris is designed to run continuously, so that electronic mail and network resources are available to users. This chapter provides guidelines for shutting down and booting a system.

This is a list of the information that in this chapter:

- "What's New in Shutting Down and Booting a System" on page 49
- "Where to Find Shut Down and Boot Tasks" on page 52
- "Shut Down and Boot Terminology" on page 53
- "Guidelines for Shutting Down a System" on page 53
- "Guidelines for Booting a System" on page 54
- "When to Shut Down a System" on page 55
- "When to Boot a System" on page 56

For an overview of all of the boot features and methods that are available in the Oracle Solaris release, see Chapter 4, "Shutting Down and Booting a System (Overview)."

For instructions on booting a system, see Chapter 7, "Booting an Oracle Solaris System (Tasks)."

What's New in Shutting Down and Booting a System

The following are new boot features in the Oracle Solaris release:

- "Support for Fast Reboot on the SPARC Platform" on page 50
- "Automatic Boot Archive Recovery" on page 50
- "GNOME Restart Dialog Support for Fast Reboot" on page 51
- "Support for Fast Reboot on the x86 Platform" on page 51
- "iSCSI Boot" on page 51

Support for Fast Reboot on the SPARC Platform

The Fast Reboot feature of Oracle Solaris, previously introduced on the x86 platform, is now supported on the SPARC platform. The integration of Fast Reboot on the SPARC platform enables the `-f` option to be used with the `reboot` command to accelerate the boot process by skipping certain POST tests.

On both the x86 and SPARC platforms, Fast Reboot is managed through SMF and implemented through a boot configuration service, `svc:/system/boot-config`. The `boot-config` service provides a means for setting or changing the default boot configuration parameters. When the `config/fastreboot_default` property is set to `true`, the system performs a fast reboot automatically, without the need to use the `reboot -f` command. By default, this property value is set to `true` on the x86 platform and to `false` on the SPARC platform.

Note – On the SPARC platform, the `boot-config` service also requires the `solaris.system.shutdown` authorization as the `action_authorization` and `value_authorization`.

To make fast reboot the default behavior on the SPARC platform, use the `svccfg` and `svcadm` commands. For instructions, see "Managing the Boot Configuration Service" on page 116.

Automatic Boot Archive Recovery

In Oracle Solaris 11 Express, boot archive recovery on the SPARC platform is automatic.

To support auto-recovery of the boot archives on the x86 platform, a new `auto-reboot-safe` property has been added to the boot configuration service, `svc:/system/boot-config:default`. By default, the property's value is set to `false` to ensure that the system does not automatically reboot to an unknown boot device. If the system is configured to automatically point to the BIOS boot device and GRUB menu entry that Oracle Solaris is installed on, you can set the property's value to `true`. Setting the value to `true` enables an automatic reboot of the system for the purpose of recovering an out-of-date boot archive.

Note – If you are running Oracle Solaris 11 Express, auto-recovery on the x86 platform is handled by the Fast Reboot feature.

To set or change the value of the `auto-reboot-safe` property, use the `svccfg` and `svcadm` commands. For more information, see the `svccfg`(1M) and `svcadm`(1M) man pages.

For general information about this enhancement, see the `boot`(1M) man page.

For step-by-step instructions, see "x86: How to Clear Automatic Boot Archive Update Failures by Using the `auto-reboot-safe` Property" on page 138.

GNOME Restart Dialog Support for Fast Reboot

In Oracle Solaris 11 Express, the GNOME restart dialog has been updated to include support for Fast Reboot and for switching boot environments (BEs) during a reboot. You can now specify a fast or a slow reboot to any bootable Oracle Solaris GRUB menu entry through the restart dialog. In addition, the dialog now lists all of the system's bootable GRUB entries, from which you can select an entry to reboot.

Note – The fast reboot functionality is supported for OS entries *only*. For all other boot entries, a slow reboot is performed.

Support for Fast Reboot on the x86 Platform

The Fast Reboot feature enables you to reboot an x86 based system, bypassing the firmware and boot loader processes. Fast Reboot implements an in-kernel boot loader that loads the kernel into memory and then switches to that kernel, so that the reboot process occurs within seconds. This feature is implemented on both 32-bit and 64-bit kernels.

If you are running Oracle Solaris 11 Express, Fast Reboot is enabled by default on the x86 platform, without the need to use the `-f` option with the `reboot` command. For information about Fast Reboot behavior in the SPARC platform, see "Support for Fast Reboot on the SPARC Platform" on page 50.

The default behavior for the Fast Reboot and Panic Fast Reboot feature of Oracle Solaris is managed through SMF and implemented through a boot configuration service, `svc:/system/boot-config`. The `boot-config` service provides a means for setting or changing the default boot configuration parameters on the x86 platform. For overview and task-related information, see "x86: Support for Fast Reboot " on page 62 and "x86: Fast Reboot Feature Enhancements" on page 62.

For task-related information, see "Using Fast Reboot" on page 113.

iSCSI Boot

The iSCSI Boot feature of Oracle Solaris enables you to initialize an operating system over the network from a remote location, such as a storage disk array. The iSCSI Boot feature supports booting from both SPARC based and x86 based systems. iSCSI Boot is typically loaded onto an initiator , while the hard disk resides on a SCSI target that is attached to the network. Because the feature uses a standard Ethernet-based infrastructure, data, storage, and networking traffic can be consolidated on a standard network.

Using iSCSI boot on a SPARC based system to boot over the network differs from a typical SPARC network boot in the following ways:

- The iSCSi boot process is a combined process that enables booting over a network and a local disk.
- The Oracle Solaris OS boots from a local disk, rather than from networked locations.

Using iSCSI boot on an x86 based system to boot over the network differs from a typical x86 network boot in the following ways:

- A GRUB based network boot requires a DHCP server that is configured for PXE clients, whereas iSCSI boot does not. However, you have the option of using a DHCP server with iSCSI boot.
- A PXE boot requires a boot server to provide the ramdisk image, whereas iSCSI boot does not. For more information about booting a system from the network, see "x86: How to Perform a GRUB Based Boot From the Network" on page 121.

For more information about iSCSI boot, see "Solaris COMSTAR iSCSI Support" in *System Administration Guide: Devices and File Systems*.

Where to Find Shut Down and Boot Tasks

Use these references to find step-by-step instructions for shutting down and booting a system.

Shut Down and Boot Task	For More Information
Shut down a SPARC or an x86 based system	Chapter 5, "Shutting Down a System (Tasks)"
Modify boot behavior	Chapter 6, "Modifying Oracle Solaris Boot Behavior (Tasks)"
Boot a SPARC based system or an x86 based system	Chapter 7, "Booting an Oracle Solaris System (Tasks)"
Manage the Solaris boot archives	Chapter 9, "Managing the Oracle Solaris Boot Archives (Tasks)"
Troubleshoot boot behavior on a SPARC or an x86 based system	"Troubleshooting Booting on the SPARC Platform (Task Map)" on page 123

Shut Down and Boot Terminology

The following terminology is used when shutting down and booting a system:

Run levels and init states	A *run level* is a letter or digit that represents a system state in which a particular set of system services are available. The system is always running in one of a set of well-defined run levels. Run levels are also referred to as *init states* because the `init` process maintains the run level. System administrators use the `init` command or the `svcadm` command to initiate a run-level transition. This book refers to init states as run levels.
Boot options	A *boot option* describes how a system is booted.

Different boot options include the following:

- **Interactive boot** – You are prompted to provide information about how the system is booted, such as the kernel and device path name.
- **Reconfiguration boot** – The system is shutdown and rebooted to add new devices, if the devices are not hot-pluggable.
- **Recovery boot** – The system is hung or an invalid entry is prohibiting the system from booting successfully or from allowing users to log in.

For terminology that is specific to GRUB based booting, see "x86: GRUB Terminology" on page 144.

Guidelines for Shutting Down a System

Keep the following in mind when you shut down a system:

- Use the `init` and `shutdown` commands to shut down a system. Both commands perform a clean system shutdown, which means that all system processes and services are terminated normally.

x86 only – For x86 based systems that are running at least the Solaris 10 6/06 release, you can initiate a clean system shutdown by pressing and releasing the power button. Shutting down an x86 based system in this manner is equivalent to using the `init 5` command to shut down a system. On some x86 based systems, the BIOS configuration might prevent the power button from initiating a system shutdown. To use the power button, reconfigure the BIOS.

- Use the `shutdown` command to shut down a server. Logged-in users and systems that mount resources from the server are notified before the server is shut down. Additional notification of system shutdowns by electronic mail is also recommended so that users can prepare for system downtime.
- You need superuser privileges to use the `shutdown` or `init` command to shut down a system.
- Both `shutdown` and `init` commands take a run level as an argument.

  The three most common run levels are as follows:

  - **Run level 3** – All system resources are available and users can log in. By default, booting a system brings it to run level 3, which is used for normal day-to-day operations. This run level is also known as multiuser level with NFS resources shared.
  - **Run level 6** – Stops the operating system and reboots to the state that is defined by the `initdefault` entry in the `/etc/inittab` file.
  - **Run level 0** – The operating system is shut down, and it is safe to turn off power. You need to bring a system to run level 0 whenever you move a system, or add or remove hardware.

  Run levels are fully described in Chapter 11, "Managing Services (Overview)."

Guidelines for Booting a System

Keep the following in mind when you boot a system:

- After a SPARC based system is shut down, it is booted by using the `boot` command at the PROM level.
- After an x86 based system is shut down, it is booted by selecting an OS instance in the GRUB menu.
- In the Solaris 9 release and some Oracle Solaris 10 releases, after an x86 based system is shut down, it is booted by using the `boot` command at the Primary Boot Subsystem menu.
- A system can be rebooted by turning the power off and then back on.

Caution – This method is not considered a clean shutdown, unless you have an x86 based system that is running a release that supports this shutdown method. Use this shutdown method only as an alternative in emergency situations. Because system services and processes are terminated abruptly, file system damage is likely to occur. The work required to repair this type of damage could be substantial and might require the restoration of various user and system files from backup copies.

- SPARC and x86 based systems use different hardware components for booting. These differences are described in Chapter 10, "x86: GRUB Based Booting (Reference)."

When to Shut Down a System

The following table lists system administration tasks and the type of shutdown method that is required to initiate the task.

TABLE 3–1 Shutting Down a System

Reason for System Shutdown	Appropriate Run Level	For More Information
To turn off system power due to anticipated power outage.	Run level 0, where it is safe to turn off power	Chapter 5, "Shutting Down a System (Tasks)"
To change kernel parameters in the `/etc/system` file.	Run level 6 (reboot the system)	Chapter 5, "Shutting Down a System (Tasks)"
To perform file system maintenance, such as backing up or restoring system data.	Run level S (single-user level)	Chapter 5, "Shutting Down a System (Tasks)"
To repair a system configuration file such as `/etc/system`.	See "When to Boot a System" on page 56	N/A
To add or remove hardware from the system.	Reconfiguration boot (shut down and turn off power when adding or removing devices, if the devices are not hot-pluggable)	"Adding a Peripheral Device to a System" in *System Administration Guide: Devices and File Systems*
To repair an important system file that is causing system boot failure.	See "When to Boot a System" on page 56	N/A
To boot the kernel debugger (`kmdb`) to track down a system problem.	Run level 0, if possible	Chapter 5, "Shutting Down a System (Tasks)"
To recover from a hung system and force a crash dump.	See "When to Boot a System" on page 56	N/A
Reboot the system by using the kernel debugger (`kmdb`), if the debugger can't be loaded at runtime.	Run level 6 (reboot the system)	"SPARC: How to Boot the System With the Kernel Debugger (`kmdb`)" on page 129 ,"x86: How to Boot a System With the Kernel Debugger in the GRUB Boot Environment (`kmdb`)" on page 132

For examples of shutting down a server or a stand-alone system, see Chapter 5, "Shutting Down a System (Tasks)."

When to Boot a System

The following table lists system administration tasks and the corresponding boot option that is used to complete the task.

TABLE 3–2 Booting a System

Reason for System Reboot	Appropriate Boot Option	Information for SPARC Based Systems	Information for x86 Based Systems
Turn off system power due to anticipated power outage.	Turn system power back on	Chapter 5, "Shutting Down a System (Tasks)"	Chapter 5, "Shutting Down a System (Tasks)"
Change kernel parameters in the `/etc/system` file.	Reboot the system to run level 3 (multiuser level with NFS resources shared)	"SPARC: How to Boot a System to Run Level 3 (Multiuser Level)" on page 96	"x86: How to Boot a System to Run Level 3 (Multiuser)" on page 107
Perform file system maintenance, such as backing up or restoring system data.	Press Control-D from run level S to bring the system back to run level 3	"SPARC: How to Boot a System to Run Level S (Single-User Level)" on page 97	"x86: How to Boot a System to Run Level S (Single-User Level)" on page 108
Repair a system configuration file such as `/etc/system`.	Interactive boot	"SPARC: How to Boot a System Interactively" on page 98	"x86: How to Boot a System Interactively" on page 109
Add or remove hardware from the system.	Reconfiguration boot (turn on system power after adding or removing devices, if devices are not hot-pluggable)	"Adding a System Disk or a Secondary Disk (Task Map)" in *System Administration Guide: Devices and File Systems*	"Adding a System Disk or a Secondary Disk (Task Map)" in *System Administration Guide: Devices and File Systems*
Boot the system by using the kernel debugger (`kmdb`) to track down a system problem.	Booting `kmdb`	"SPARC: How to Boot the System With the Kernel Debugger (`kmdb`)" on page 129	"x86: How to Boot a System With the Kernel Debugger in the GRUB Boot Environment (`kmdb`)" on page 132
To recover from a hung system and force a crash dump.	Recovery boot	"SPARC: How to Force a Crash Dump and Reboot of the System" on page 125	"x86: How to Force a Crash Dump and Reboot of the System" on page 131

◆ ◆ ◆ CHAPTER 4

Shutting Down and Booting a System (Overview)

This chapter provides an overview of booting a system. The Oracle Solaris boot design, boot processes, and various methods of booting a system are described.

This is a list of the information that is in this chapter:

- "Fundamentals of the Oracle Solaris Boot Design" on page 57
- "Overview of the SPARC Boot Architecture" on page 59
- "x86: Overview of the GRUB Bootloader" on page 61
- "x86: Support for Fast Reboot " on page 62
- "Booting From an Oracle Solaris ZFS Root File System" on page 64

For more information on a specific topic, see the following references:

- For what's new in shutting down and booting a system, see "What's New in Shutting Down and Booting a System" on page 49.
- For instructions on booting an Oracle Solaris system, see Chapter 7, "Booting an Oracle Solaris System (Tasks)"
- For overview information and instructions on administering boot loaders and modifying boot behavior, see Chapter 6, "Modifying Oracle Solaris Boot Behavior (Tasks)."

Fundamentals of the Oracle Solaris Boot Design

Note – The information in this section applies to both the SPARC and x86 platforms.

The fundamental Oracle Solaris boot design includes the following characteristics:

- **Use of a boot archive**

  The boot archive is a ramdisk image that contains all of the files that are required for booting a system. For more information, see "Implementation of the Boot Archives on SPARC" on page 60.

- **Use of the bootadm command to manage the Oracle Solaris boot archives**

  The `bootadm` command handles the details of boot archive update and verification on both the SPARC and x86 platform automatically. During an installation or upgrade, the `bootadm` command creates an initial boot archive. During the process of a normal system shutdown, the shutdown process compares the boot archive's contents with the root file system. If there are any inconsistencies, the system rebuilds the boot archive to ensure that upon reboot, the boot archive and root file system are synchronized. You can use the `bootadm` command to manually update the boot archives. See "Using the `bootadm` Command to Manage the Boot Archives" on page 139.

  Note – Some options of the `bootadm` command cannot be used on the SPARC platform.

  For more information, see the `bootadm`(1M) and `boot`(1M) man pages.

- **Use of a ramdisk image as the root file system during installation**

  This process is the same on the SPARC and x86 platforms. The ramdisk image is derived from the boot archive and then transferred to the system from the boot device.

  Note – On the SPARC platform, the OpenBoot PROM continues to be used to access the boot device and to transfer the boot archive to the system's memory. Conversely, on the x86 platform, the system is initially controlled by the BIOS. The BIOS is used to initiate a transfer of the boot archive from a network device or to run a boot loader. In the Oracle Solaris OS, the x86 boot loader that is used to transfer the boot archive from disk is GRUB. See "x86: Boot Processes" on page 143.

  In the case of a software installation, the ramdisk image is the root file system that is used for the entire installation process. Using the ramdisk image for this purpose eliminates the need to boot the system from removable media. The ramdisk file system type can be a High Sierra File System (HSFS).

Overview of the SPARC Boot Architecture

The boot processes on the SPARC platform have been redesigned and improved to increase commonality with the x86 boot experience. The SPARC boot design enables the addition of new features, for example new file system types, without necessitating any changes to multiple portions of the boot chain. Changes also include the implementation of boot phase independence.

Highlights of these improvements include:

- Commonality in boot processes on the SPARC and x86 platforms
- Commonality in the network boot experience
- Boot architecture flexibility that enables booting a system from different file system types more easily

The following four boot phases are now independent of each other:

1. **Open Boot PROM (OBP) phase**

   The OBP phase of the boot process on the SPARC platform is unchanged.

   For disk devices, the firmware driver usually uses the OBP label package's *load* method, which parses the VTOC label at the beginning of the disk to locate the specified partition. Sectors 1-15 of the partition are then read into the system's memory. This area is commonly called the boot block and usually contains a file system reader.
2. **Booter phase**

   During this phase the boot archive is read and executed. Note that this is the only phase of the boot process that requires knowledge of the boot file system format. Protocols that are used for the transfer of the boot loader and the boot archive include local disk access, NFS, and HTTP.
3. **Ramdisk phase**

   The ramdisk is a boot archive that is comprised of kernel modules any other components that are required to boot an instance of Oracle Solaris..

   The SPARC boot archive is identical to an x86 boot archive. The boot archive file system format is private. Therefore, knowledge of the file system type that is used during a system boot, for example an HSFS file system, is not required by the booter or the kernel. The ramdisk extracts the kernel image from the boot archive and then executes it.
4. **Kernel phase**

   The kernel phase is the final stage of the boot process. During this phase, Oracle Solaris is initialized and a minimal root file system is mounted on the ramdisk that was constructed from the boot archive. In some environments, such as an installation, the ramdisk is used as the root (`/`) file system and remains mounted.The ramdisk contains a set of kernel files and drivers that is sufficient to mount the root file system on the specified root device.

The kernel then extracts the remainder of the primary modules from the boot archive, initializes itself, mounts the real root file system, then discards the boot archive.

Support for Booting Multiple Kernels

On SPARC based systems, when you boot the system from the `ok` prompt, the default boot device is automatically selected. An alternate boot device can be specified by changing the NVRAM variable for the `boot-device`. You can also specify an alternate boot device or an alternate kernel (boot file) from the command line at boot time. See "SPARC: How to Boot a Kernel Other Than the Default Kernel" on page 100.

Implementation of the Boot Archives on SPARC

The boot archives, previously only available on the x86 platform, are now an integral part of the SPARC boot architecture.

The `bootadm` command has been modified for use on the SPARC platform. This command functions the same as it does on the x86 platform. The `bootadm` command handles the details of archive update and verification. On the x86 platform the `bootadm` command updates the GRUB menu during an installation or system upgrade. You can also use the `bootadm` command to manually manage the boot archives.

The boot archive service is managed by the Service Management Facility (SMF). The service instance for the boot archive is `svc:/system/boot-archive:default`. To enable, disable, or refresh this service use the `svcadm` command. For information about managing services by using SMF, see Chapter 11, "Managing Services (Overview)."

The files that are included in the SPARC boot archives are located in the `/platform` directory.

The contents of the `/platform` directory is divided into two groups of files:

- Files that are required for a `sun4u` boot archive
- Files that are required for `sun4v` boot archive

For information about managing the boot archives, see "Managing the Oracle Solaris Boot Archives (Task Map)" on page 135.

x86: Overview of the GRUB Bootloader

The open source GRand Unified Bootloader (GRUB) is the default boot loader on x86 based systems. GRUB is responsible for loading a boot archive into the system's memory. A boot archive is a collection of critical files that is needed during system startup before the root file system is mounted. The boot archive is the interface that is used to boot Oracle Solaris. You can find more information about GRUB at `http://www.gnu.org/software/grub/grub.html`. See also the `grub`(5) man page.

GRUB Based Booting

After an x86 based system is powered on, the Basic Input/Output System (BIOS) initializes the CPU, the memory, and the platform hardware. When the initialization phase has completed, the BIOS loads the boot loader from the configured boot device and then transfers control of the system to the boot loader. The *boot loader* is the first software program that runs after you turn on a system. This program starts the boot process.

GRUB implements a menu interface that includes boot options that are predefined in a configuration file called the `menu.lst` file. GRUB also has a command-line interface that is accessible from the GUI menu interface that can be used to perform various boot functions, including modifying default boot behavior.

Because the Oracle Solaris kernel is fully compliant with the Multiboot Specification, you can boot x86 based systems by using GRUB. With GRUB, you can boot various operating systems that are installed on a single x86 based system. For example, you can individually boot Oracle Solaris, Linux, or Windows by selecting the boot entry in the GRUB menu at boot time, or by configuring the `menu.lst` file to boot a specific OS by default.

Because GRUB is intuitive about file systems and kernel executable formats, you can load an operating system without recording the physical position of the kernel on the disk. With GRUB-based booting, the kernel is loaded by specifying its file name, and the drive and the partition where the kernel resides. For more information see "Naming Conventions That Are Used for Configuring GRUB" on page 145.

For step-by-step instructions on booting a system with GRUB, see "Booting an x86 Based System (Task Map)" on page 107.

See also the following man pages:

- `boot`(1M)
- `bootadm`(1M)
- `grub`(5)
- `installgrub`(1M)

x86: Support for the findroot Command

The findroot command, which functions similarly to the root command previously used by GRUB, has enhanced capabilities for discovering a targeted disk, regardless of the boot device. The findroot command also supports booting from an Oracle Solaris ZFS root file system.

The most common format for the menu.lst entry for this command is as follows:

```
findroot (rootfs0,0,a)
kernel$ /platform/i86pc/kernel/$ISADIR/unix
module$ /platform/i86pc/$ISADIR/boot_archive
```

For more information, see "x86: Implementation of the findroot Command" on page 92.

For GRUB reference information, see Chapter 10, "x86: GRUB Based Booting (Reference)."

x86: Support for Fast Reboot

In Oracle Solaris 11 express, the Fast Reboot feature is supported on both the SPARC and x86 platform. The integration of Fast Reboot on the SPARC platform enables the -f option to be used with the reboot command to accelerate the boot process by skipping certain POST tests. On the x86 platform, Fast Reboot implements an in-kernel boot loader that loads the kernel into memory and then switches to that kernel. The firmware and boot loader processes are bypassed, which enables the system to reboot within seconds.

On both the x86 and SPARC platforms, the Fast Reboot feature is managed by SMF and implemented through a boot configuration service, svc:/system/boot-config. The boot-config service provides a means for setting or changing the default boot configuration parameters. When the config/fastreboot_default property is set to true, the system performs a fast reboot automatically, without the need to use the reboot -f command. This property's value is set to false on the SPARC platform and true on the x86 platform. For task-related information, including how to change the default behavior of Fast Reboot on the SPARC platform, see "Managing the Boot Configuration Service" on page 116.

x86: Fast Reboot Feature Enhancements

In this release, the following feature enhancements have been implemented:

- Fast Reboot and Panic Fast Reboot are the default operating mode on the x86 platform. All of the Fast Reboot features that were initially introduced are now enabled by default, without requiring the use of the -f option with the reboot command.

  To facilitate this support, a new `boot-config` service, `svc:/system/boot-config:default`, has been introduced. This service consists of the following properties, which are enabled by default.

  - `config/fastreboot_default`
  - `config/fastreboot_onpanic`

  For more information about managing the `boot-config` service, see "Managing the Boot Configuration Service" on page 116.

- Two other key feature enhancements include the following:
  - Capability for fast rebooting a system to the *n*th entry in the GRUB `menu.lst` file.
  - Capability for fast rebooting a system to a newly activated BE.
- Support for fast rebooting a system in the GNOME restart dialog

  The restart dialog that is displayed in the GNOME Desktop when you restart your system includes support for fast reboot functionality, as well as a mechanism for switching boot environments. When booting a system, you now have the option of specifying a fast or a slow reboot to any bootable GRUB menu entry through the restart dialog.

  Note – Only Oracle Solaris boot entries support the fast reboot functionality. For any other boot entries, a slow reboot is automatically initiated.

For task-related information, see "Using Fast Reboot (Task Map)" on page 112.

`quiesce` Function

The system's capability to bypass the firmware when booting a new OS image has dependencies on the device drivers' implementation of a new device operation entry point, `quiesce`. On supported drivers, this implementation `quiesces` a device, so that at completion of the function, the driver no longer generates interrupts or access memory. This implementation also resets the device to a hardware state, from which the device can be correctly configured by the driver's attach routine, without a power cycle of the system or being configured by the firmware. For more information about this functionality, see the `quiesce`(9E) and `dev_ops`(9S) man pages.

Note – Not all device drivers implement the `quiesce` function. For troubleshooting instructions, see "x86: Troubleshooting Conditions That Might Prevent Fast Reboot From Working" on page 117.

`uadmin` Function

The `uadmin` function, `AD_FASTREBOOT`, resets the system, thereby enabling the `reboot` command to bypass the BIOS and the boot loader phases.

Caution – Although the `uadmin 2 8` command can be used to fast reboot a system. note that when the command is used, neither the boot archive, nor the `menu.lst` file are updated. For this reason, the `reboot` or the `init 6` command is the preferred method for initiating a fast reboot of a system.

For more information, see the `uadmin(2)`man page.

Booting From an Oracle Solaris ZFS Root File System

In Oracle Solaris 11 Express, ZFS is the default root file system. Booting from an Oracle Solaris ZFS root file system works differently than booting from a UFS file system. Because ZFS applies several new concepts for installation and booting, some basic administrative practices for booting a system have changed. The most significant difference between booting from a ZFS root file system and booting from a UFS root file system is that with ZFS a device identifier does *not* uniquely identify a root file system, and thus a BE. With ZFS, a device identifier uniquely identifies a *storage pool*. A storage pool can contain multiple bootable datasets (root file systems). Therefore, in addition to specifying a boot device, a root file system within the pool that is identified by the boot device must also be specified.

On an x86 based system, if the boot device identified by GRUB contains a ZFS storage pool, the `menu.lst` file that is used to create the GRUB menu is located in the dataset at the root of that pool's dataset hierarchy. This dataset has the same name as the pool. There is one such dataset in each pool.

A *default bootable dataset* is the bootable dataset for the pool that is mounted at boot time and is defined by the root pool's `bootfs` property. When a device in a root pool is booted, the dataset that is specified by this property is then mounted as the root file system.

The new `bootfs` pool property is a mechanism that is used by the system to specify the default bootable dataset for a given pool. When a device in a root pool is booted, the dataset that is mounted by default as the root file system is the one that is identified by the `bootfs` pool property.

On a SPARC based system, the default `bootfs` pool property is overridden by using the new `-Z` *dataset* option of the `boot` command.

On an x86 based system, the default `bootfs` pool property is overridden by selecting an alternate boot environment in the GRUB menu at boot time.

For task-related information, see "SPARC: Booting From a Specified ZFS Root File System" on page 101 and "x86: Booting From a Specified ZFS Root File System" on page 110.

◆ ◆ ◆ CHAPTER 5

Shutting Down a System (Tasks)

This chapter describes the procedures for shutting down systems.

This is a list of the information that is in this chapter:

- "Shutting Down the System (Task Map)" on page 67
- "Shutting Down the System" on page 68
- "Turning Off Power to All Devices" on page 75

For overview information about system run levels, see Chapter 11, "Managing Services (Overview)."

For information on the procedures associated with run levels and boot files, see "Shutting Down the System (Task Map)" on page 67.

Shutting Down the System (Task Map)

Task	Description	For Instructions
Determine who is logged in to a system.	Use the `who` command to determine who is logged in to a system.	"How to Determine Who Is Logged in to a System" on page 70
Shut down a server.	Use the `shutdown` command with the appropriate options to shut down a server.	"How to Shut Down a Server" on page 70
Shut down a stand-alone system.	Use the `init` command and indicate the appropriate run-level to shut down a stand-alone system.	"How to Shut Down a Stand-Alone System" on page 73

Task	Description	For Instructions
Turn off power to all devices.	Powering down a system includes the following devices: ■ CPU ■ Monitor ■ External devices, such as disks, tapes, and printers	"How to Turn Off Power to All Devices" on page 75

Shutting Down the System

Oracle Solaris is designed to run continuously, so that the electronic mail and network software can work correctly. However, some system administration tasks and emergency situations require that the system be shut down to a level where it is safe to remove power. In some cases, the system needs to be brought to an intermediate level, where not all system services are available.

Such cases include the following:

- Adding or removing hardware
- Preparing for an expected power outage
- Performing file system maintenance, such as a backup

For a complete list of system administration tasks that require a system shutdown, see Chapter 4, "Shutting Down and Booting a System (Overview)."

For information on using your system's power management features, see the `pmconfig`(1M) man page.

System Shutdown Commands

The use of the `init` and `shutdown` commands are the primary ways to shut down a system. Both commands perform a *clean shutdown* of the system. As such, all file system changes are written to the disk, and all system services, processes, and the operating system are terminated normally.

The use of a system's Stop key sequence or turning a system off and then on are not clean shutdowns because system services are terminated abruptly. However, sometimes these actions are needed in emergency situations. For instructions on system recovery techniques, see Chapter 7, "Booting an Oracle Solaris System (Tasks)," andChapter 9, "Managing the Oracle Solaris Boot Archives (Tasks)."

Note – On x86 systems that are running at least the Solaris 10 6/06 release, pressing and releasing the power button initiates a clean system shutdown. This method is equivalent to using the `init 5` command.

The following table describes the various shutdown commands and provides recommendations for using them.

TABLE 5–1 Shutdown Commands

Command	Description	When To Use
`shutdown`	An executable shell script that calls the `init` program to shut down the system. The system is brought to run level S by default.	Recommended for servers operating at run level 3 because users are notified of the impending shutdown. Also notified are the systems that are mounting resources from the server that is being shut down.
`init`	An executable that kills all active processes and synchronizes the disks before changing run levels.	Recommended for stand-alone systems when other users will not be affected. Provides a faster system shutdown because users are not notified of the impending shutdown.
`reboot`	An executable that synchronizes the disks and passes boot instructions to the `uadmin` system call. In turn, this system call stops the processor.	The `init` command is the preferred method.
`halt`, `poweroff`	An executable that synchronizes the disks and stops the processor.	Not recommended because it doesn't shutdown all processes, and unmount any remaining file systems. Stopping the services, without doing a clean shutdown, should only be done in an emergency or if most of the services are already stopped.

User Notification of System Down Time

When the `shutdown` command is initiated, a warning followed by a final shutdown message is broadcast to all users who are currently logged in to the system and all systems that are mounting resources from the affected system.

For this reason, the `shutdown` command is preferred instead of the `init` command when you need to shut down a server. When you use either command, you might want to give users more notice by sending them a mail message about any scheduled system shutdown.

Use the `who` command to determine which users on the system need to be notified. This command is also useful for determining a system's current run level. For more information, see "Determining a System's Run Level" on page 158 and the `who(1)` man page.

▼ How to Determine Who Is Logged in to a System

1. **Log in to the system to be shut down.**

2. **Display all of the users who are logged in to the system.**

   ```
   $ who
   ```

Example 5–1 Determining Who Is Logged in to a System

The following example shows how to display who is logged in to the system.

```
$ who
holly       console      May  7 07:30
kryten      pts/0        May  7 07:35   (starlite)
lister      pts/1        May  7 07:40   (bluemidget)
```

- Data in the first column identifies the user name of the logged-in user.
- Data in the second column identifies the terminal line of the logged-in user.
- Data in the third column identifies the date and time that the user logged in.
- Data in the forth column, if present, identifies the host name if a user is logged in from a remote system.

▼ How to Shut Down a Server

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2. **Find out if users are logged in to the system.**

   ```
   # who
   ```

   A list of all logged-in users is displayed. You might want to send mail or broadcast a message to let users know that the system is being shut down.

3. **Shut down the system.**

   ```
   # shutdown -iinit-level -ggrace-period -y
   ```

   `-i`*init-level* Brings the system to an init level that is different from the default of S. The choices are 0, 1, 2, 5, and 6.

   Run levels 0 and 5 are reserved states for shutting the system down. Run level 6 reboots the system. Run level 2 is available as a multi-user operating state.

`-g`*grace-period*	Indicates a time (in seconds) before the system is shut down. The default is 60 seconds.
`-y`	Continues to shut down the system without intervention. Otherwise, you are prompted to continue the shutdown process after 60 seconds.

For more information, see the `shutdown(1M)` man page.

4 **If you are asked for confirmation, type `y`.**

```
Do you want to continue? (y or n): y
```

If you used the `shutdown -y` command, you will not be prompted to continue.

5 **Type the superuser password, if prompted.**

```
Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance): xxxxxx
```

6 **After you have finished the system administration tasks, press Control-D to return to the default system run level.**

7 **Use the following table to verify that the system is at the run level that you specified in the `shutdown` command.**

Specified Run Level	SPARC Based System Prompt	x86 Based System Prompt
S (single-user level)	#	#
0 (power-down level)	ok or >	`Press any key to reboot`
Run level 3 (multiuser level with remote resources shared)	*hostname* `console login:`	*hostname* `console login:`

Example 5–2 SPARC: Bringing a Server to Run Level S

In the following example, the `shutdown` command is used to bring a SPARC based system to run level S (single-user level) in three minutes.

```
# who
root   console      Jun 14 15:49    (:0)

# shutdown -g180 -y

Shutdown started.    Mon Jun 14 15:46:16 MDT 2004

Broadcast Message from root (pts/4) on venus Mon Jun 14 15:46:16...
The system venus will be shut down in 3 minutes .
.
.
Broadcast Message from root (pts/4) on venus Mon Jun 14 15:46:16...
The system venus will be shut down in 30 seconds .
```

```
.
.
INIT: New run level: S
The system is coming down for administration.  Please wait.
Unmounting remote filesystems: /vol nfs done.
Shutting down Solaris Management Console server on port 898.
Print services stopped.
Jun 14 15:49:00 venus syslogd: going down on signal 15
Killing user processes: done.

Requesting System Maintenance Mode
SINGLE USER MODE

Root password for system maintenance (control-d to bypass): xxxxxx
single-user privilege assigned to /dev/console.
Entering System Maintenance Mode
#
```

Example 5–3 SPARC: Bringing a Server to Run Level 0

In the following example, the `shutdown` command is used to bring a SPARC based system to run level 0 in 5 minutes without requiring additional confirmation.

```
# who
root       console       Jun 17 12:39
userabc         pts/4        Jun 17 12:39   (:0.0)
# shutdown -i0 -g300 -y
Shutdown started.    Thu Jun 17 12:40:25 MST 2004

Broadcast Message from root (console) on pretend Thu Jun 17 12:40:25...
The system pretend will be shut down in 5 minutes
.
.
.
Changing to init state 0 - please wait
#
INIT: New run level: 0
The system is coming down.  Please wait.
System services are now being stopped.
.
.
.
The system is down.
syncing file systems... done
Program terminated
Type  help  for more information
ok
```

If you are bringing the system to run level 0 to turn off power to all devices, see "How to Turn Off Power to All Devices" on page 75.

Example 5–4 SPARC: Rebooting a Server to Run Level 3

In the following example, the `shutdown` command is used to reboot a SPARC based system to run level 3 in two minutes. No additional confirmation is required.

```
# who
root           console      Jun 14 15:49    (:0)
userabc    pts/4        Jun 14 15:46    (:0.0)
# shutdown -i6 -g120 -y
Shutdown started.    Mon Jun 14 15:46:16 MDT 2004

Broadcast Message from root (pts/4) on venus Mon Jun 14 15:46:16...
The system venus will be shut down in 2 minutes


Changing to init state 6 - please wait
#
INIT: New run level: 6
The system is coming down.  Please wait.
.
.
.
The system is down.
syncing file systems... done
rebooting...
.
.
.
venus console login:
```

See Also Regardless of why you shut down a system, you will probably want to return to run level 3, where all file resources are available, and users can log in. For instructions on bringing a system back to a multiuser level, see Chapter 7, "Booting an Oracle Solaris System (Tasks)."

▼ How to Shut Down a Stand-Alone System

Use this procedure when you need to shut down a stand-alone system.

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2. **Shut down the system.**

   ```
   # init 5
   ```

   For more information, see the `init(1M)` man page.

   - **Alternately, you can use the `uadmin` command to shut down the system.**

     ```
     # uadmin 2 0
     ```

   - **If you have an x86 based system that is running at least the Solaris 10 6/06 release, you can press and release the power button to initiate a clean system shutdown and turn off the system.**

     This functionality is equivalent to using the `init 5` command to shut down a system. For more information, see "What's New in Shutting Down and Booting a System" on page 49.

3 **Use the following table to verify that the system is at the run level that you specified in the `init` command.**

Specified Run Level	SPARC Based System Prompt	x86 Based System Prompt
S (single-user level)	#	#
2 (multiuser level)	#	#
0 (power-down level)	`ok` or `>`	`Press any key to reboot`
3 (multiuser level with NFS resources shared)	*hostname* `console login:`	*hostname* `console login:`

Example 5–5 Using the `uadmin` command to Shut Down a System

```
# uadmin 2 0
syncing file systems... done
Program terminated
```

Example 5–6 Bringing a Stand-Alone System to Run Level 0

In this example, the `init` command is used to bring an x86 based stand-alone system to the level where it is safe to turn off power.

```
# init 0
#
INIT: New run level: 0
The system is coming down.  Please wait.
.

.

.
The system is down.
syncing file systems... [11] [10] [3] done
Press any key to reboot
```

If you are bringing the system to run level 0 to turn off power to all devices, see "How to Turn Off Power to All Devices" on page 75.

Example 5–7 SPARC: Bringing a Stand-Alone System to Run Level S

In this example, the `init` command is used to bring a SPARC based stand-alone system to run level S (single-user level).

```
# init s
#
INIT: New run level: S
The system is coming down for administration.  Please wait.
```

```
Unmounting remote filesystems: /vol nfs done.
Print services stopped.
syslogd: going down on signal 15
Killing user processes: done.

SINGLE USER MODE

Root password for system maintenance (control-d to bypass): xxxxxx
single-user privilege assigned to /dev/console.
Entering System Maintenance Mode
#
```

See Also Regardless of why you shut down the system, you will probably want to return to run level 3, where all file resources are available, and users can log in. For instructions on bringing a system back to a multiuser level, see Chapter 7, "Booting an Oracle Solaris System (Tasks)."

Turning Off Power to All Devices

You need to turn off power to all system devices when you do the following:

- Replace or add hardware.
- Move the system from one location to another.
- Prepare for an expected power outage or natural disaster such as an approaching electrical storm.

Turn off the power to system devices, including the CPU, the monitor, and external devices such as disks, tapes, and printers.

Before you turn off the power to all system devices, shut down the system cleanly, as described in the preceding sections.

▼ How to Turn Off Power to All Devices

1. **Select one of the following methods to shut down the system:**
   - **If you are shutting down a server, see "How to Shut Down a Server" on page 70.**
   - **If you are shutting down a stand-alone system, see "How to Shut Down a Stand-Alone System" on page 73.**
2. **Turn off the power to all devices after the system is shutdown. If necessary, also unplug the power cables.**

3 **After power can be restored, use the following steps to turn on the system and devices.**

a. **Plug in the power cables.**

b. **Turn on the monitor.**

c. **Turn on disk drives, tape drives, and printers.**

d. **Turn on the CPU.**

   The system is brought to run level 3.

CHAPTER 6

Modifying Oracle Solaris Boot Behavior (Tasks)

This chapter provides information about modifying boot behavior.

This is list of the information that is in this chapter:

- "Modifying Boot Behavior on SPARC Based Systems (Task Map)" on page 77
- "Modifying Boot Behavior on x86 Based Systems (Task Map)" on page 85

For what's new in booting and general overview information about the boot process, see Chapter 3, "Introduction to Shutting Down and Booting a System."

For step-by-step instructions on booting an Oracle Solaris system, see Chapter 7, "Booting an Oracle Solaris System (Tasks)."

Modifying Boot Behavior on SPARC Based Systems (Task Map)

Task	Description	For Instructions
Identify the PROM revision number.	Use the banner command at the `ok` prompt to display the PROM revision number for a system.	"SPARC: How to Find the PROM Revision Number for a System" on page 79
Identify devices on the system that can be booted.	Before modifying boot behavior by using the boot PROM, identify the devices on the system.	"SPARC: How to Identify Devices on a System" on page 79
Display the current boot device.	Use this procedure to determine the current default boot device from which the system will boot.	"SPARC: How to Determine the Default Boot Device" on page 81

Task	Description	For Instructions
Change the default boot device.	To change the default boot device, use one of the following methods: ■ Change the `boot-device` parameter at the boot PROM. ■ Change the `boot-device` parameter by using the `eeprom` command.	"SPARC: How to Change the Default Boot Device by Using the Boot PROM" on page 81 "SPARC: How to Change the Default Boot Device by Using the `eeprom` Command" on page 83
Reset the system.	When you reset the system, the system runs diagnostic tests on the hardware, then reboots.	"SPARC: Resetting the System" on page 83
Change the default boot file.	To change the default kernel that the system boots, use one of the following methods: ■ Change the `boot-file` parameter by using the boot PROM. ■ Change the`boot-file` parameter by using the `eeprom` command.	"SPARC: How to Change the Default Kernel by Using the Boot PROM" on page 83 "SPARC: How to Change the Default Kernel by Using the `eeprom` Command" on page 84

Modifying Boot Behavior on SPARC Based Systems

SPARC: Using the Boot PROM

The boot PROM is used to boot a system. You might need to change the way the system boots. For example, you might want to reset the device to boot from or run hardware diagnostics before you bring the system to a multiuser level.

System administrators typically use the PROM level to boot a system. You can also change the default boot file and boot device at the PROM level.

If you need to perform any of the following tasks, you need to change the default boot device:

- Add a new drive to the system either permanently or temporarily
- Change the network boot strategy
- Temporarily boot a stand-alone system from the network

For a complete list of PROM commands, see the `monitor`(1M) and `eeprom`(1M) man pages.

▼ SPARC: How to Find the PROM Revision Number for a System

● **Display a system's PROM revision number by using the `banner` command.**

```
ok banner
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 333MHz), No Keyboard
OpenBoot 3.15, 128 MB memory installed, Serial #number.
Ethernet address number, Host ID: number.
```

Hardware configuration information, including the revision number of the PROM, is displayed. In the preceding example, the PROM revision number is 3.15.

▼ SPARC: How to Identify Devices on a System

You might need to identify the devices on the system to determine what are the appropriate devices to boot from.

Before You Begin Before you can safely use the `probe` commands to determine what devices are attached to the system, you need to do the following:

- Change the PROM `auto-boot?` parameter to false.

  ```
  ok setenv auto-boot? false
  ```
- Issue the `reset-all` command to clear system registers.

  ```
  ok reset-all
  ```

You can view the `probe` commands that are available on your system by using the `sifting probe` command:

```
ok sifting probe
```

If you run the `probe` commands without clearing the system registers, the following message is displayed:

```
ok probe-scsi
This command may hang the system if a Stop-A or halt command
has been executed.  Please type reset-all to reset the system
before executing this command.
Do you wish to continue? (y/n) n
```

1 **Identify the devices on the system.**

```
ok probe-device
```

2 (Optional) If you want the system to reboot after a power failure or after using the `reset` command, then reset the `auto-boot?` parameter to true.

```
ok setenv auto-boot? true
auto-boot? =          true
```

3 Boot the system to multiuser mode.

```
ok reset-all
```

Example 6–1 SPARC: Identifying the Devices on a System

The following example shows how to identify the devices connected to an Ultra 10 system.

```
ok setenv auto-boot? false
auto-boot? =          false
ok reset-all
Resetting ...

Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 333MHz), No Keyboard
OpenBoot 3.15, 128 MB memory installed, Serial #10933339.
Ethernet address 8:0:20:a6:d4:5b, Host ID: 80a6d45b.

ok probe-ide
  Device 0  ( Primary Master )
          ATA Model: ST34321A

  Device 1  ( Primary Slave )
         Not Present

  Device 2  ( Secondary Master )
         Removable ATAPI Model: CRD-8322B

  Device 3  ( Secondary Slave )
         Not Present

ok setenv auto-boot? true
auto-boot? =          true
```

Alternatively, you can use the `devalias` command to identify the device aliases and the associated paths of devices that *might* be connected to the system. For example:

```
ok devalias
screen                   /pci@1f,0/pci@1,1/SUNW,m64B@2
net                      /pci@1f,0/pci@1,1/network@1,1
cdrom                    /pci@1f,0/pci@1,1/ide@3/cdrom@2,0:f
disk                     /pci@1f,0/pci@1,1/ide@3/disk@0,0
disk3                    /pci@1f,0/pci@1,1/ide@3/disk@3,0
disk2                    /pci@1f,0/pci@1,1/ide@3/disk@2,0
disk1                    /pci@1f,0/pci@1,1/ide@3/disk@1,0
disk0                    /pci@1f,0/pci@1,1/ide@3/disk@0,0
ide                      /pci@1f,0/pci@1,1/ide@3
floppy                   /pci@1f,0/pci@1,1/ebus@1/fdthree
ttyb                     /pci@1f,0/pci@1,1/ebus@1/se:b
ttya                     /pci@1f,0/pci@1,1/ebus@1/se:a
keyboard!                /pci@1f,0/pci@1,1/ebus@1/su@14,3083f8:forcemode
```

```
keyboard                 /pci@1f,0/pci@1,1/ebus@1/su@14,3083f8
mouse                    /pci@1f,0/pci@1,1/ebus@1/su@14,3062f8
name                     aliases
```

▼ SPARC: How to Determine the Default Boot Device

1. **Bring the system to the ok PROM prompt.**

   For more information, see "How to Shut Down a Stand-Alone System" on page 73.

2. **Use the `printenv` command to determine the default boot device.**

   ```
   ok printenv boot-device
   ```

`boot-device`	Identifies the parameter for setting the device from which to boot.
device[*n*]	Identifies the `boot-device` value such as a disk or the network. The *n* can be specified as the *disk number*.

   The default boot-device is displayed in a format that is similar to the following:

   ```
   boot-device = /pci@1f,4000/scsi@3/disk@1,0:a
   ```

   If the default boot-device is a network boot device, the output is similar to the following:

   ```
   boot-device = /sbus@1f,0/SUNW,fas@e,8800000/sd@a,0:a \
   /sbus@1f,0/SUNW,fas@e,8800000/sd@0,0:a disk net
   ```

▼ SPARC: How to Change the Default Boot Device by Using the Boot PROM

You might need to identify the devices on the system before you can change the default boot device to some other device. For information on identifying devices on the system, see "SPARC: How to Identify Devices on a System" on page 79.

1. **Change to run level 0.**

   ```
   # init 0
   ```

   The `ok` PROM prompt is displayed. For more information, see the `init`(1M) man page.

2. **Change the value of the `boot-device` parameter.**

   ```
   ok setenv boot-device device[n]
   ```

   Use one of the `probe` commands if you need help identifying the disk number.

3. **Verify that the default boot device has been changed.**

   ```
   ok printenv boot-device
   ```

4 **Save the new `boot-device` value.**

```
ok reset-all
```

The new `boot-device` value is written to the PROM.

Example 6–2 SPARC: Changing the Default Boot Device

In this example, the default boot device is set to disk.

```
# init 0
#
INIT: New run level: 0
.
.
.
The system is down.
syncing file systems... done
Program terminated
ok setenv boot-device /pci@1f,4000/scsi@3/disk@1,0
boot-device =         /pci@1f,4000/scsi@3/disk@1,0
ok printenv boot-device
boot-device           /pci@1f,4000/scsi@3/disk@1,0
ok boot
Resetting ...

screen not found.
Can't open input device.
Keyboard not present.  Using ttya for input and output.

Sun Enterprise 220R (2 X UltraSPARC-II 450MHz), No Keyboard
OpenBoot 3.23, 1024 MB memory installed, Serial #13116682.
Ethernet address 8:0:20:c8:25:a, Host ID: 80c8250a.

Rebooting with command: boot disk1
Boot device: /pci@1f,4000/scsi@3/disk@1,0  File and args:
```

In this example, the default boot device is set to the network.

```
# init 0
#
INIT: New run level: 0
.
.
.
The system is down.
syncing file systems... done
Program terminated
ok setenv boot-device net
boot-device =         net
ok printenv boot-device
boot-device           net                     disk
ok reset
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 333MHz), No Keyboard
OpenBoot 3.15, 128 MB memory installed, Serial #number.
Ethernet address number, Host ID: number.
```

```
Boot device: net  File and args:
.
.
.
pluto console login:
```

▼ SPARC: How to Change the Default Boot Device by Using the `eeprom` Command

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2. **Specify the alternate kernel to boot.**

   ```
   # eeprom boot-device new-boot-device
   ```

3. **Verify that the new parameter has been set.**

   ```
   # eeprom boot-device
   ```

   The output should display the new `eeprom` value for the `boot-device` parameter.

SPARC: Resetting the System

Run the following command from the `ok` prompt:

```
ok reset-all
```

The self-test program, which runs diagnostic tests on the hardware, is executed. Then, if the `auto-boot?` parameter is set to true, the system is rebooted.

▼ SPARC: How to Change the Default Kernel by Using the Boot PROM

1. **Change to run level 0.**

   ```
   # init 0
   ```

   The `ok` PROM prompt is displayed. For more information, see the`init(1M)` man page.

2 **Set the `boot-file` property to an alternate kernel.**

```
ok setenv boot-file boot-file
```

3 **Verify that the default boot device has been changed.**

```
ok printenv boot-file
```

4 **Save the new `boot-file` value.**

```
ok reset-all
```

The new `boot-file` value is written to the PROM.

▼ SPARC: How to Change the Default Kernel by Using the `eeprom` Command

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2 **Specify the alternate kernel to boot.**

```
# eeprom boot-file new boot-file
```

For example:

```
# eeprom boot-file=kernel.name/sparcv9/unix
```

3 **Verify that the new parameter has been set.**

```
# eeprom boot-file
```

The output should display the new `eeprom` value for the specified parameter.

Modifying Boot Behavior on x86 Based Systems (Task Map)

Task	Description	For Instructions
Set boot file parameters by using the `eeprom` command.	Modify boot behavior on an x86 based system by using the `eeprom` command. Boot options that are set by using the `eeprom` command persist over a system reboot, unless these options are overridden by modifying kernel behavior in the GRUB menu at boot time.	"x86: How to Modify Boot Behavior by Using the `eeprom` Command" on page 86
Modify boot behavior by editing the GRUB menu at boot time.	Modify boot behavior by editing GRUB menu at boot time. Boot options that are specified by modifying the boot behavior in the GRUB menu persist only until the next system reboot.	"x86: How to Modify Boot Behavior by Editing the GRUB Menu at Boot Time" on page 88
Modify boot behavior by manually editing the `menu.lst` file.	Modify boot behavior by editing the `menu.lst` configuration file to add new OS entries or redirect the console. Changes you make to the file persist over system reboots.	"x86: How to Modify Boot Behavior by Editing the `menu.lst` File" on page 90
Modify the `menu.lst` file to include entries that support the `findroot` command.	Additional menu entries that use the `findroot` command can be added to the `menu.lst` file menu after an installation or upgrade.	"x86: How to Add GRUB Menu Entries That Use the `findroot` Command" on page 93

Modifying Boot Behavior on x86 Based Systems

The primary methods for modifying boot behavior on an x86 based system are as follows:

- By using the `eeprom` command.

  The `eeprom` command is used to assign a different value to a standard set of properties. These values, which are equivalent to the SPARC OpenBoot PROM NVRAM variables, are stored either in the `/boot/solaris/bootenv.rc` file or in the `menu.lst` file. Changes that are made to boot behavior by using the `eeprom` command persist over each system reboot and are preserved during a software upgrade. See the `eeprom(1M)` man page for more information.

Caution – If you directly edit the `menu.lst` file, certain boot properties (boot-file, boot-args, and console) may not be changed at a later time by using the `eeprom` command.

- By editing the GRUB menu at boot time.

  Changes that are made by modifying the GRUB kernel behavior at boot time override options that you set by using the `eeprom` command. However, these changes only remain in effect until the next time you boot the system. See the `kernel`(1M) man page for more information.
- By manually editing the GRUB `menu.lst` file.

Caution – Any system-generated changes that are made to `menu.lst` entries are changed or lost during a system upgrade. However, any new boot entries that were manually added remain after an upgrade. You can override `eeprom` settings by editing the GRUB menu at boot time or by editing the `menu.lst` file. Changes made by editing the GRUB menu at boot time do not persist. Whereas, changes that are made by editing the `menu.lst` file persist over system reboots.

▼ x86: How to Modify Boot Behavior by Using the `eeprom` Command

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2. **Change the specified parameter.**

   ```
   # eeprom parameter=new-value
   ```

3. **Verify that the new parameter has been set.**

   ```
   # eeprom parameter
   ```

   The output should display the new `eeprom` value for the specified parameter.

Example 6–3 x86: Setting `boot-file` Parameters by Using the `eeprom` Command

This example shows how to manually specify that the system boot a 64-bit kernel. The system must support 64-bit computing.

```
# eeprom boot-file=kernel/amd64/unix
```

This example shows how to manually boot a 32-bit kernel on a 64-bit capable system.

```
# eeprom boot-file=kernel/unix
```

This example shows how to restore the default auto detected boot behavior on a system.

```
# eeprom boot-file=""
```

x86: Modifying Boot Behavior by Editing the GRUB Menu at Boot Time

---

Note – The information that is contained in the `menu.lst` file varies, depending on the Oracle Solaris release and the installation method that was used.

To edit a boot entry in the GRUB menu, use the arrow keys to select the entry, then type `e`.

```
GNU GRUB  version 0.95  (637K lower / 3144640K upper memory)
 +-------------------------------------------------------------------------+
findroot (BE_be1,0,a)
bootfs rpool/ROOT/szboot_0508
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/$ISADIR/boot_archive
 +-------------------------------------------------------------------------+
      Use the ^ and v keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the
      commands before booting, or 'c' for a command-line.
```

For instructions on editing the GRUB menu at boot time, see "x86: How to Modify Boot Behavior by Editing the GRUB Menu at Boot Time" on page 88.

Boot Arguments That Can Be Specified by Editing the GRUB Menu at Boot Time

The following list describes the boot arguments and options that you can specify by editing the GRUB menu at boot time:

unix	Specifies the kernel to boot
`-a`	Prompts the user for configuration information
`-s`	Boots the system in single-user mode
`-r`	Specifies a reconfiguration boot The system probes all attached hardware devices and then assigns nodes in the file system to represent only those devices that are actually found.

`-v`	Boots the system with verbose messages enabled
`-x`	Does not boot the system in clustered mode
`-k`	Boots the system with the kernel debugger enabled
`-m` *smf-options*	Controls the boot behavior of the Service Management Facility (SMF) There are two categories of options: recovery options and messages options.
`-i altinit`	Specifies an alternative executable as the primordial process. `altinit` is a valid path to an executable.
`-B` *prop=value [,prop=value]...*	Specifies kernel boot properties.

The following are various ways you can modify boot behavior in the GRUB menu by using the `-B` *prop=val* option:

`-B console=ttya`	Redirects the console to `ttya`.
`-B acpi-enum=off`	Disables Advanced Configuration and Power Interface (ACPI) enumeration of devices.
`-B console=ttya,acpi-enum=off`	Redirects the console to `ttya` and disables the ACPI enumeration of devices.
`-B acpi-user-options=0x2`	Disables ACPI entirely.

Note – When properties are specified by using the `eeprom` command *and* on the GRUB command line, the GRUB command takes precedence.

▼ x86: How to Modify Boot Behavior by Editing the GRUB Menu at Boot Time

When you modify the GRUB kernel behavior by editing the GRUB menu at boot time, the changes do not persist over a system reboot. Default boot behavior is restored the next time you boot the system.

1. **Reboot the system.**

   When the boot sequence begins, the GRUB main menu is displayed.

2. **Use the arrow keys to select the boot entry to edit.**

3. **Type `e` to access the GRUB edit menu.**

4 Select the `kernel` or `kernel$` line in this menu.

5 Type `e` to add boot arguments to the line.

6 Type any additional boot arguments.

7 Press Return to save your changes and return to the previous menu.

Note – Pressing the Escape key returns you to the GRUB main menu without saving your changes.

---

8 To boot the system, type `b`.

Changes you make take affect when the system is booted.

x86: Modifying Boot Behavior by Editing the `menu.lst` File

The GRUB menu, which is based on the `menu.lst` configuration file, can be customized. When you install or upgrade your system, the `bootadm` command automatically updates the `menu.lst` file to reflect menu entries that are supported for that particular release. Any newly installed OS that is listed in this file is displayed as a boot entry in the GRUB menu when the system is rebooted. Note that when installing an operating system other than Oracle Solaris, you need to manually add the menu entry to the `menu.lst` file afterwards.

A configurable timeout is available to boot the default OS entry. The default OS boot entry that is booted is configurable through the `default` command. The installation software typically sets this command to boot one of the valid boot entries. To boot a different instance of the Oracle Solaris OS (if applicable), or to boot a different OS, use the arrow keys to highlight a different boot entry. Then press Enter to boot that entry. Note that if the `default` command is not set, the first boot entry in the GRUB menu is booted.

Only the *active* `menu.lst` file is used to boot the system. To modify the GRUB menu that is displayed when you boot the system, edit the active GRUB `menu.lst` file. Changing any other `menu.lst` file has no effect on the menu that is displayed when you boot the system To determine the location of the active `menu.lst` file, use the `list-menu` subcommand of the `bootadm` command. For more information about using the `bootadm` command, see "Using the `bootadm` Command to Manage the Boot Archives" on page 139.

▼ x86: How to Modify Boot Behavior by Editing the `menu.lst` File

You might need to modify the `menu.lst` file for one of the following reasons:

- To add new OS entries
- To add GRUB console redirection information

Before You Begin Because only the *active* GRUB `menu.lst` file is used to boot the system, make sure you edit the correct file. Changing any other GRUB `menu.lst` file has no effect on the menu that is displayed when you boot the system.

The location of the active `menu.lst` file varies for systems that have a ZFS root is /*pool-name*/`boot/grub/menu.lst`.

You can determine the location of the active GRUB `menu.lst` file by using the `bootadm` command with the `list-menu` subcommand.

```
# bootadm list-menu
```

For more information about the `bootadm` command, see the `bootadm`(1M) man page.

1 **Become the root user..**

```
$ su -
Password:
#
```

2 **To add a new OS entry to the active `menu.lst` file, use a text editor to modify the file.**

The comments within the `menu.lst` file provide you with the necessary information for adding a new OS entry.

The following is an example of a `menu.lst` file for a system that is running a release with ZFS boot support. Boot entries in the `menu.lst` file vary, depending on the Oracle Solaris release that you are running.

```
#---------- ADDED BY BOOTADM - DO NOT EDIT ----------
title Solaris 11 s10x_90 X86
findroot (pool_rpool,0,a)
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/$ISADIR/boot_archive
#---------------------END BOOTADM--------------------
```

Caution – Do not directly edit the original contents of the `menu.lst` file. To make changes to any of the OS entries in the file, manually edit the file to duplicate the existing content. Then, make the modifications to the duplicated content.

Also note when manually adding new entries to the file, never include guard comments, for example, "Added by bootadm". These comments are reserved for use by the system. Not using these comments ensures that these entries remain intact during a software upgrade.

- **To specify a 32-bit kernel as the default kernel to boot, remove all instances of the `$ISADIR` keyword from the `menu.lst` file.**

If you have added any additional entries, beyond the default entries, make any equivalent changes manually.

The [`-B *`] and [`*`] flags must be preserved, if these flags exist in the original `menu.lst` file.

3 **After adding the required information, save the file.**

Note that any changes you make to the file take effect at the next system reboot.

Tip – If you are running Linux, and install Oracle Solaris, the Linux entry is not preserved in the GRUB menu when the system is rebooted. Before installing or upgrading your system, save a copy of the `menu.lst` file that contains the Linux information. After the installation, add the Linux information to the newly created `menu.lst` file in the Solaris partition.

Because changes you make to the `menu.lst` file are not directly related to the Oracle Solaris OS, you cannot make them by using the `eeprom` command. You must edit the file directly. Note that the software upgrade process preserves any changes that you make to the `menu.lst` file.

Caution – GRUB is capable of booting both Linux and Oracle Solaris. However, Linux GRUB is not capable of booting Oracle Solaris.

Always ensure that one of the following conditions are met:

- That the `fdisk` partition is active, that it has GRUB installed, and that the `menu.lst` file is the *active* GRUB menu.
- That Oracle Solaris GRUB is installed to the Master Boot Record (MBR), and that it refers to the `menu.lst` in the `fdisk` partition.

Example 6–4 `menu.lst` File on a System With an Oracle Solaris ZFS Boot Loader

The following examples show what a `menu.lst` file looks like on a system that has an Oracle Solaris ZFS boot loader. By default, this system will boot from a ZFS root file system. Note that the contents of the file varies, depending on the installation type.

New installation:

```
findroot (pool_rpool,0,a)
bootfs rpool/ROOT/nwambe-1
splashimage /boot/solaris.xpm
foreground d25f00
background 115d93
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/$ISADIR/boot_archive
```

x86: Locating the Active GRUB `menu.lst` File

On systems that have a ZFS root, the active `menu.lst` file is typically located in `/`*pool-name*`/boot/grub/menu.lst`.

To locate the active GRUB menu, use the `bootadm` command with the `list-menu` subcommand:

```
# bootadm list-menu
```

This command also lists the contents of the active `menu.lst` file:

For example:

```
# bootadm list-menu
The location for the active GRUB menu is: /rpool/boot/grub/menu.ls
default 1
timeout 30
0 oraclesolaris 11
1 nwambe-1
2 nwambe-2
3 newbet
```

For further instructions on using the `bootadm` command, see "Using the `bootadm` Command to Manage the Boot Archives" on page 139.

x86: Implementation of the `findroot` Command

All installation methods now use the `findroot` command for specifying which disk slice on an x86 based system to boot. This enhancement supports booting systems with Oracle Solaris ZFS roots. This information is located in the `menu.lst` file that is used by GRUB. Previously, the `root` command, `root` (`hd0.0.a`), was explicitly used to specify which disk slice to boot.

The installation methods include Oracle Solaris JumpStart, and the installation GUI program.

In addition to the `findroot` command, is a signature file on the slice, (*mysign*, `0`, `a`), where *mysign* is the name of a signature file that is located in the `/boot/grub/bootsign` directory. When booting a system from a ZFS root, the ZFS GRUB plug-in looks for and tries to mount a ZFS file system in slice `a` of `fdisk` partition `0`.

The name of the signature file varies, depending on the installation method that was used. For more information about the naming conventions that are used by the `findroot` command, see "Naming Conventions That Are Used by the `findroot` Command" on page 146.

Additional menu entries that also use the `findroot` command can be added to the GRUB menu after an installation or upgrade. For instructions, see "x86: How to Add GRUB Menu Entries That Use the `findroot` Command" on page 93.

Caution – The boot signature must be unique. Do not use or remove system-generated signatures or user signatures that are duplicated across multiple instances of the Oracle Solaris software. Doing so might result in booting an incorrect OS instance or prevent the system from booting.

Note that the `root` command can still be used in the `menu.lst` file in certain instances, for example to boot Windows. However, do not use the `root` command in cases where the `findroot` command is the preferred choice.

EXAMPLE 6–5 x86: Default `menu.lst` file That Supports an Oracle Solaris ZFS Boot Loader

This is an example of a `menu.lst` file on system that supports an Oracle Solaris ZFS boot loader.

```
findroot (pool_rpool,0,a)
bootfs rpool/ROOT/nwambe-1
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/$ISADIR/boot_archive
```

▼ x86: How to Add GRUB Menu Entries That Use the `findroot` Command

This procedure shows how to manually update the `menu.lst` file with user-defined entries that use the `findroot` command. Typically, these entries are added after an installation or an upgrade. For guidelines on adding user-defined entries that use the `findroot` command, see "x86: Implementation of the `findroot` Command" on page 92.

1 **Become the root user..**

2 Create a boot signature file on the root (`/`) file system or root pool that will be booted.

- **For a ZFS pool, *my-pool*, create the boot signature file in the `/`*my-pool*`/boot/grub/bootsign` directory.**

  ```
  # touch   /my-pool/boot/grub/bootsign/user-sign
  ```

Note – Make sure the file name that you choose for the boot signature is unique. Do not use system-generated signature names or user signature names that are duplicated across multiple instances of Oracle Solaris. Doing so might prevent the system from booting or cause the wrong Oracle Solaris instance to boot.

---

3 Add a menu entry that contains the `findroot` command.

a. Locate the active `menu.lst` file:

```
# bootadm list-menu
```

b. Using a text editor, edit the active `menu.lst` file to add the following entry:

```
title    User Solaris boot entry
findroot  (user-sign, 3, c)
kernel$   /platform/i86pc/kernel/$ISADIR/unix
module$  /platform/i86pc/$ISADIR/boot_archive
```

In the preceding example, the `3` represents the 4th `fdisk` partition (partitions start at 0). The `c` represents the slice within a Solaris `fdisk` partition (slices start with a).

4 Reboot the system.

The new entry is displayed in the GRUB menu and can be selected to boot the specified Oracle Solaris instance.

◆ ◆ ◆ CHAPTER 7

Booting an Oracle Solaris System (Tasks)

This chapter describes the procedures for booting Oracle Solaris on SPARC and x86 based systems.

This is a list of information that is in this chapter:

- "Booting a SPARC Based System (Task Map)" on page 95
- "Booting an x86 Based System (Task Map)" on page 107
- "Using Fast Reboot (Task Map)" on page 112

For overview information about the boot process, see Chapter 4, "Shutting Down and Booting a System (Overview)."

Booting a SPARC Based System (Task Map)

Task	Description	For Instructions
Boot a SPARC based system to run level 3.	Use this boot method after shutting down the system or performing a system hardware maintenance task.	"SPARC: How to Boot a System to Run Level 3 (Multiuser Level)" on page 96
Boot a SPARC based system to run level S.	Use this boot method to boot the system after performing a system maintenance task such as backing up a file system. At this level, only local file systems are mounted and users cannot log in to the system.	"SPARC: How to Boot a System to Run Level S (Single-User Level)" on page 97
Boot a SPARC based system interactively.	Use this boot method after making temporary changes to a system file or the kernel for testing purposes.	"SPARC: How to Boot a System Interactively" on page 98

Task	Description	For Instructions
Boot a kernel other than default kernel.	Use this procedure to boot a kernel other than the default kernel. Alternately, you can obtain a copy of an alternate boot file, change the default kernel to the new kernel, then set the `boot-file` parameter to boot the new default boot device.	"SPARC: How to Boot a Kernel Other Than the Default Kernel" on page 100
Display a list of the available ZFS bootable datasets on a SPARC based system.	Use the `boot -L` command to display a list of the available BEs within a ZFS pool on a system. **Note** – This option is only supported for boot devices that contain a ZFS pool.	"SPARC: How to List Available Bootable Datasets Within a ZFS Root Pool" on page 102
Boot a SPARC based system from a specified ZFS root file system.	Use the `boot -Z` option to boot a specified ZFS dataset. **Note** – This option is only supported for boot devices that contain a ZFS pool.	"SPARC: How to Boot From a Specified ZFS Root File System" on page 103
Boot a SPARC based system from the network.	Use this boot method to boot a system from the network. Note that this method is also used for booting a diskless client.	"SPARC: How to Boot a System From the Network" on page 105

Booting a SPARC Based System

If a system is turned off, turning it on starts the multiuser boot sequence. The following procedures show how to boot to different run levels from the `ok` PROM prompt. These procedures assume that the system has been cleanly shut down, unless stated otherwise.

Use the `who -r` command to verify that the system is brought to the specified run level. For a description of run levels, see Chapter 11, "Managing Services (Overview)."

▼ SPARC: How to Boot a System to Run Level 3 (Multiuser Level)

Use this procedure to boot a system that is currently at run level 0 to run level 3.

1 **Boot the system to run level 3.**

```
ok boot
```

The automatic boot procedure displays a series of startup messages, and brings the system to run level 3. For more information, see the `boot(1M)` man page.

2 **Verify that the system has booted to run level 3.**

The login prompt is displayed when the boot process has finished successfully.

```
hostname console login:
```

Example 7–1 SPARC: Booting a System to Run Level 3 (Multiuser Level)

The following example displays the messages from booting a system to run level 3.

```
ok boot
Resetting ...

Sun Ultra 2 UPA/SBus (2 X UltraSPARC-II 296MHz), No Keyboard
OpenBoot 3.25, 512 MB memory installed, Serial #10342381.
Ethernet address 8:0:20:xx:cf:ed, Host ID: 80xxcfed.


Rebooting with command: boot
Boot device: /sbus@1f,0/SUNW,fas@e,8800000/sd@a,0:a  File and args: kadb
Loading kmdb...
SunOS Release 5.10        64-bit
Copyright 1983-2007 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms
WARNING: consconfig: cannot find driver for screen device /SUNW,ffb@1e,0
Hostname: dancehallgirl
NIS domain name is boulder.Central.Sun.COM
/dev/rdsk/c0t10d0s7 is clean
Reading ZFS config: done.

dancehallgirl console login:
```

▼ SPARC: How to Boot a System to Run Level S (Single-User Level)

Use this procedure to boot a system that is currently at run level 0 to run level S. This run level is used for system maintenance tasks, such as backing up a file system.

1 **Boot the system to run level S.**

```
ok boot -s
```

2 **Type the superuser password when the following message is displayed:**

```
SINGLE USER MODE

Root password for system maintenance (control-d to bypass): xxxxxx
```

3 **Verify that the system is at run level S.**

```
# who -r
```

4 **Perform the maintenance task that required the run level change to S.**

5 **After you complete the system maintenance task, type Control-D to bring the system to the multiuser state.**

Example 7–2 SPARC: Booting a System to Run Level S (Single-User Level)

The following example displays the messages from booting a system to run level S.

```
ok boot -s
Resetting ...

Sun Ultra 2 UPA/SBus (2 X UltraSPARC-II 296MHz), No Keyboard
OpenBoot 3.25, 512 MB memory installed, Serial #10342381.
Ethernet address 8:0:20:xx:cf:ed, Host ID: 80xxcfed.

Rebooting with command: boot -s
Boot device: /sbus@1f,0/SUNW,fas@e,8800000/sd@a,0:a  File and args: -s
SunOS Release 5.11
Copyright 1983-2007 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms
WARNING: consconfig: cannot find driver for screen device /SUNW,ffb@1e,0

Root password for system maintenance (control-d to bypass):
svc.startd: Returning to milestone all.
NIS domain name is boulder.Central.Sun.COM
/dev/rdsk/c0t10d0s7 is clean
Reading ZFS config: done.
dancehallgirl console login:
```

▼ SPARC: How to Boot a System Interactively

Use this boot option when you need to specify an alternate kernel or `/etc/system` file.

Before You Begin To specify an alternate `/etc/system` file when booting a SPARC based system interactively by using the `boot -a` command, you must perform the following steps before the system is booted.

- 1. Make backup copies of the `/etc/system` and `boot/solaris/filelist.ramdisk` files. For example:

  ```
  # cp /etc/system /etc/system.bak
  # cp /boot/solaris/filelist.ramdisk /boot/solaris/filelist.ramdisk.orig
  ```
- 2. Add the `etc/system.bak` file name to the `/boot/solaris/filelist.ramdisk` file.

  ```
  # echo "etc/system.bak" >> /boot/solaris/filelist.ramdisk
  ```
- 3. Update the boot archive.

  ```
  # bootadm update-archive -v
  ```

1 **Boot the system interactively.**

```
ok boot -a
```

2 Answer the following system prompts:

a. When prompted, enter the name of the kernel to use for booting.

Press enter to use the default kernel file name. Otherwise, provide the name of an alternate kernel, press Enter.

b. When prompted, provide an alternate path for the `modules` directories.

Press enter to use the default module directories. Otherwise, provide the alternate paths to module directories, press Enter.

c. When prompted, provide the name of an alternate system file.

Type `/dev/null` if your `/etc/system` file has been damaged.

d. When prompted, enter the `root` filesystem type.

e. When prompted, enter the physical name of `root` device.

Provide an alternate device name or press return to use the default.

3 If you are not prompted to answer these questions, verify that you typed the `boot -a` command correctly.

Example 7–3 SPARC: Booting a System Interactively

In this example, the default choices (shown in square brackets `[]`) are accepted. For instructions and an example of booting an alternate file system by using the `boot -a` command, see "SPARC: How to Boot a System Interactively" on page 98.

```
ok boot -a
Resetting ...

Sun Ultra 2 UPA/SBus (2 X UltraSPARC-II 296MHz), No Keyboard
OpenBoot 3.25, 512 MB memory installed, Serial #10342381.
Ethernet address 8:0:20:9d:cf:ed, Host ID: 809dcfed.



Rebooting with command: boot -a
Boot device: /sbus@1f,0/SUNW,fas@e,8800000/sd@a,0:a  File and args: -a

Boot device: /sbus@1f,0/SUNW,fas@e,8800000/sd@a,0:a  File and args: -a
Name of system file [/etc/system]:
SunOS Release 5.11 Version zwicky:nbsclean-build:12/04/2007 64-bit
Copyright 1983-2007 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Retire store [/etc/devices/retire_store] (/dev/null to bypass):
root filesystem type [ufs]:
Enter physical name of root device
[/sbus@1f,0/SUNW,fas@e,8800000/sd@a,0:a]:
```

```
WARNING: consconfig: cannot find driver for screen device /SUNW,ffb@1e,0
Hostname: dancehallgirl
NIS domain name is boulder.Central.Sun.COM
/dev/rdsk/c0t10d0s7 is clean
Reading ZFS config: done.
dancehallgirl login:
```

▼ SPARC: How to Boot a Kernel Other Than the Default Kernel

1. **Become the root user.**

2. **Obtain a copy of an existing Oracle Solaris kernel and rename it.**

3. **Add the kernel that you copied and renamed in Step 2 to the `/etc/boot/solaris/filelist.ramdisk` file.**

   ```
   # echo "kernel.name" >> /boot/solaris/filelist.ramdisk
   ```

4. **Verify that the alternate kernel has been added to the `/etc/boot/solaris/filelist.ramdisk` file.**

   ```
   # cat > /etc/boot/solaris/filelist.ramdisk
   ```

5. **Update the boot archive by using the `bootadm` command.**

   ```
   # bootadm update-archive
   ```

6. **Change to run level 0.**

   ```
   # init 0
   ```

   The ok PROM prompt is displayed.

7. **Boot the alternate kernel.**

   ```
   ok boot alternate-kernel
   ```

   For example:

   ```
   ok boot kernel.myname/sparcv9/unix
   ```

   - **To boot the alternate kernel by default, follow these steps:**

     a. **Set the boot-file parameter to the new kernel.**

     ```
     ok setenv boot-file kernel.name/sparc9/unix
     ```

     b. **Verify that the boot-file property has been changed.**

     ```
     ok printenv boot-file
     ```

c. **Reboot the system.**

```
ok boot
```

8 **After the system has booted, verify that the alternate kernel that was booted.**

```
# prtconf -vp | grep whoami
```

Example 7–4 Booting an Alternate Kernel by Changing the Default Boot File

```
# cp -r /platform/sun4v/kernel /platform/sun4vu/kernel.caiobella
# echo "kernel.caiobela" >> /boot/solaris/filelist.ramdisk

# cat > /etc/boot/solaris/filelist.ramdisk
/platform/sun4v/kernel.caiobella
^D (control D)

ok setenv boot-file kernel.caiobells/sparcv9/unix
ok printenv boot-file
boot-file = kernel.caiobella/sparcv9/unix

ok boot

SC Alert: Host System has Reset

SC Alert: Host system has shut down.


Sun Fire T200, No KeyboardCopyright 2006 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.25.0.build_01***PROTOTYPE BUILD***, 32760 MB memory available, Serial
#69060038.
Ethernet address 0:x:4f:x:c5:c6, Host ID: 8xxc5c6.



Rebooting with command: boot
Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a  File and
args: kernel.caiobella/sparcv9/unix
#
#
# prtconf -vp | grep whoami
        whoami:  '/platform/sun4v/kernel.caiobella/sparcv9/unix'
```

SPARC: Booting From a Specified ZFS Root File System

The following two options support booting from a ZFS root file system on the SPARC platform:

`-L` Displays a list of available bootable datasets within a ZFS pool.

Note – The boot `-L` command is executed from the OBP, *not* from the command line.

`-Z` *dataset* Boots the root file system for the specified ZFS bootable dataset.

If you are booting a system from a ZFS root file system, first use the `boot` command with the `-L` option from the OBP to print a list of the available BEs on the system. Then, use the `-Z` option to boot the specified BE.

For more information, see the `boot`(1M) man page.

▼ SPARC: How to List Available Bootable Datasets Within a ZFS Root Pool

On SPARC based systems, the `menu.lst` file contains the following two GRUB commands:

- `title` – Provides a title for a boot environment (BE)
- `bootfs` – Specifies the full name of the bootable dataset

To display a list of the bootable datasets within a ZFS pool, choose from the following methods:

- Use the `boot -L` command. This command displays a list of the available BEs in a given ZFS pool and provides instructions for booting the system.

The following procedure describes how to use the `boot -L` command to list available BEs on a system. To boot a specified BE after running this command, follow the instructions that are printed on the screen.

1 **Become the root user..**

2 **Bring the system to the ok PROM prompt.**

```
# init 0
```

3 **List the available BEs in a ZFS pool:**

```
ok boot device-specifier -L
```

4 **To boot one of the entries that is displayed, type the number that corresponds to the entry.**

5 **Boot the specified BE by following the directions that are printed on the screen.**

For instructions, see "SPARC: How to Boot From a Specified ZFS Root File System" on page 103.

Example 7–5 SPARC: Displaying a List of Available BEs on a System by Using `boot -L`

```
# init 0
# svc.startd: The system is coming down. Please wait.
svc.startd: 94 system services are now being stopped.
svc.startd: The system is down.
syncing file systems... done
Program terminated
ok boot -L
```

```
.
.
.
Boot device: /pci@1f,0/pci@1/scsi@8/disk@0,0 File and args: -L
zfs-file-system
Loading: /platformsun4u/bootlst
1.s10s_nbu6wos
2 zfs2BE
Select environment to boot: [ 1 - 2 ]: 2

to boot the selected entry, invoke:
boot [<root-device] -Z rpool/ROOT/zfs2BE
```

See Also For more information, see Chapter 5, "Managing ZFS Root Pool Components," in *Oracle Solaris ZFS Administration Guide*.

▼ SPARC: How to Boot From a Specified ZFS Root File System

When booting from ZFS, a device specifier identifies a storage pool, *not* a single root file system. A storage pool can contain multiple bootable datasets, or root file systems. Therefore, when booting from ZFS, you must also identify a root file system within the pool that is identified by the boot device as the default. By default, the default boot device is identified by the pool's `bootfs` property. This procedure shows how to boot the system by specifying a ZFS bootable dataset. See the `boot`(1M) man page for a complete description of all the boot options that are available.

Note – In Oracle Solaris 11 Express, a ZFS root file system is booted by default. Use this procedure to specify a ZFS root file system to boot.

For more information, see `zpool`(1M) man page.

1. **Become the root user.**

2. **Bring the system to the ok PROM prompt.**

   ```
   # init 0
   ```

3. **(Optional) To display a list of available BEs, use the `boot` command with the `-L` option.**

   For instructions, see "SPARC: How to List Available Bootable Datasets Within a ZFS Root Pool" on page 102.

4. **To boot a specified entry, type the number of the entry and press Return:**

   ```
   Select environment to boot: [1 - 2]:
   ```

5 **To boot the system, follow the instructions that are printed to the screen:**

```
To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/dataset

ok boot -Z rpool/ROOT/dataset
```

For example:

```
# boot -Z rpool/ROOT/zfs2BE
```

6 **After the system has booted, type the following command to verify the active BE:**

```
# prtconf -vp | grep whoami
```

- **To display the boot path for the active BE, type:**

  ```
  # prtconf -vp | grep bootpath
  ```

- **Alternately, you can use the `df -lk` command to determine whether the correct BE was booted.**

Example 7–6 SPARC: Booting From a Specified ZFS Root File System

This example shows how to use the `boot -Z` command to boot a ZFS dataset on a SPARC based system.

```
# init 0
# svc.startd: The system is coming down. Please wait.
svc.startd: 79 system services are now being stopped.
svc.startd: The system is down.
syncing file systems... done
Program terminated
ok boot -Z rpool/ROOT/zfs2BEe
Resetting
LOM event: =44d+21h38m12s host reset
g ...

rProcessor Speed = 648 MHz
Baud rate is 9600
8 Data bits, 1 stop bits, no parity (configured from lom)

.
.
.
Environment monitoring: disabled
Executng last command: boot -Z rpool/ROOT/zfs2BE
Boot device: /pci@1f,0/pci@1/scsi@8/disk@0,0 File and args: -Z rpool/ROOT/zfs2Be
zfs-file-system
Loading: /platform/SUNW,UltraAX-i2/boot_archive
Loading: /platform/sun4u/boot_archive
ramdisk-root hsfs-file-system
Loading: /platform/SUNW,UltraAX-i2/kernel/sparcv9/unix
Loading: /platform/sun4u/kernel/sparcv9/unix
```

```
.
.
.
Hostname: mallory
NIS domainname is ...
Reading ZFS config: done.
Mounting ZFS filesytems: (6/6)

mallory console login:
```

Booting a SPARC Based System From the Network

You might need to boot a system from the network under the following conditions:

- When the system is first installed
- If the system won't boot from the local disk
- If the system is a diskless client

Two network configuration boot strategies are available:

- Reverse Address Resolution Protocol (RARP) and ONC+ RPC Bootparams Protocol
- Dynamic Host Configuration Protocol (DHCP)

For network devices, the process for booting over a local area network (LAN) and booting over a wide area network (WAN) is slightly different. In both network boot scenarios, the PROM downloads the booter from a boot server or an installation server, which is `inetboot` in this case.

When booting over a (LAN), the firmware uses RARP and BOOTP or DHCP to discover the boot or installation server. TFTP is then used to download the booter, which is `inetboot` in this case.

When booting over a WAN, the firmware uses either DHCP or NVRAM properties to discover the installation server, the router, and the proxies that are required for the system to boot from the network. The protocol that is used to download the booter is HTTP. In addition, the booter's signature might be checked with a predefined private key.

▼ SPARC: How to Boot a System From the Network

Any system can boot from the network, if a boot server is available. You might want to boot a stand-alone system from the network if the system cannot boot from the local disk. For information on changing or resetting the default boot device, see “SPARC: How to Change the Default Boot Device by Using the Boot PROM” on page 81.

Two network configuration boot strategies are available on sun–4u systems:

- RARP – Reverse Address Resolution Protocol and ONC+ RPC Bootparams Protocol
- DHCP – Dynamic Host Configuration Protocol

The default network boot strategy is set to RARP. You can use either protocol, depending on whether a RARP boot server or a DHCP boot server is available on your network.

Note – Sun Ultra systems must have at least PROM version 3.25.*nn* to use the DHCP network boot strategy. For information on determining your PROM version, see "SPARC: How to Find the PROM Revision Number for a System" on page 79.

If both protocols are available, you can temporarily specify which protocol to use in the `boot` command. Or, you can save the network boot strategy across system reboots at the PROM level by setting up an NVRAM alias. The following example uses the `nvalias` command to set up a network device alias for booting with DHCP by default on a Sun Ultra 10 system.

```
ok nvalias net    /pci@1f,4000/network@1,1:dhcp
```

As a result, when you type `boot net`, the system boots by using DHCP.

Note – Do not use the `nvalias` command to modify the `NVRAMRC` file, unless you are very familiar with the syntax of this command and the `nvunalias` command. For information about using these commands, see the *OpenBoot 3.x Command Reference Manual*.

Before You Begin To boot with either protocol successfully, you must have already set up a RARP or DHCP boot server on your network.

1 **If necessary, shut down the system.**

2 **Determine the method for booting from the network, then select one of the following:**

a. **Boot the system from the network by using the DHCP strategy.**

```
ok boot net[:dhcp]
```

If you have changed the PROM setting to boot with DHCP by default, as in the preceding `nvalias` example, you only have to specify `boot net`.

b. **Boot the system from the network by using the RARP strategy.**

```
ok boot net[:rarp]
```

Because RARP is the default network boot strategy, you only have to specify `boot net:rarp` if you have changed the PROM value to boot DHCP.

Booting an x86 Based System (Task Map)

Task	Description	For Instructions
Boot an x86 based system to run level 3, multiuser level.	Use this boot method to bring the system back to multiuser level after shutting down the system or performing a system hardware maintenance task.	"x86: How to Boot a System to Run Level 3 (Multiuser)" on page 107
Boot an x86 based system the in single-user mode.	Use this boot method to perform a system maintenance task, such as backing up a file system.	"x86: How to Boot a System to Run Level S (Single-User Level)" on page 108
Boot an x86 based system interactively.	Use this boot method after making temporary changes to a system file or the kernel for testing purposes.	"x86: How to Boot a System Interactively" on page 109
Display a list a ZFS bootable datasets on an x86 based system.	Use the `bootadm list-menu` command to display the available BEs on an x86 based system that has a ZFS root file system:	"x86: How to Display a List of the Available ZFS Boot Environments" on page 111
Boot an x86 based system from a specified ZFS root file system.	If you install or upgrade your system to an Oracle Solaris release that supports a ZFS boot loader, the GRUB menu entry for the default ZFS BE contains the `-B $ZFS-BOOTFS` boot argument. The system therefore automatically boots from aZFS root. **Note –** This option is supported *only* for boot devices that contain a ZFS pool.	"x86: How to Boot From a Specified ZFS Root File System" on page 111
Boot an x86 based system from the network by using GRUB.	Use this method to boot a PXE or non-PXE device from the network with the default network configuration strategy. This method is also used for booting a diskless client.	"x86: How to Perform a GRUB Based Boot From the Network" on page 121

Booting an x86 Based System

▼ x86: How to Boot a System to Run Level 3 (Multiuser)

Use this procedure to boot a system that is currently at run level 0 to run level 3.

1 **Reboot the system.**

```
# reboot
```

If the system displays the `Press any key to reboot` prompt, press any key to reboot the system.

You can also use the Reset button at this prompt. If the system is shut down, turn the system on with the power switch.

When the boot sequence begins, the GRUB menu is displayed.

2 **When the GRUB menu is displayed, press Enter to boot the default OS instance.**

If you do not choose an entry within 10 seconds, the system automatically boots to run level 3.

The login prompt is displayed when the boot process has finished successfully.

3 **Log in to the system.**

hostname `console login:`

4 **Verify that the system booted to run level 3.**

```
# who -r
system% who -r
   .       run-level 3  Mar  2 09:44     3      0  S
```

▼ x86: How to Boot a System to Run Level S (Single-User Level)

Use this procedure to boot a system that is at run level 0 to run level S. The single-user level is used for performing system maintenance.

Note – This procedure can be used for all GRUB implementations. However, the boot entries in the GRUB main menu vary, depending on which Oracle Solaris release you are running.

For a description of all of the kernel options that you can specify at boot time, see "x86: Modifying Boot Behavior by Editing the GRUB Menu at Boot Time" on page 87.

1 **Reboot the system.**

```
# reboot
```

If the system displays the `Press any key to reboot` prompt, press any key to reboot the system.

You can also use the Reset button at this prompt. If the system is shut down, turn the system on with the power switch.

When the boot sequence begins, the GRUB menu is displayed.

2 **When the GRUB main menu is displayed, type e to edit the GRUB menu.**

3 **Depending on the release you are running, use the arrow keys to choose the `kernel` or `kernel$` line.**

If you cannot use the arrow keys, use the caret key (^) key to scroll up and the letter v key to scroll down.

4 **Type e again to edit the boot entry.**

From here, you can add options and arguments to the `kernel` or `kernel$` line.

5 **To boot the system in single-user mode, type -s at the end of the boot entry line, then press Return to go back to the previous screen.**

- **To specify other boot behaviors, replace the -s option with the appropriate boot option.**

  The following alternate boot behaviors can be specified in this manner:

  - Perform a reconfiguration boot
  - Boot a 64-bit capable system in 32-bit mode
  - Boot the system with the kernel debugger
  - Redirect the console

  For more information, see the `boot`(1M) man page.

6 **To boot the system in single-user mode, type b.**

7 **When prompted, type the root password.**

Note – You might need to also enter a user name *before* entering the root password. The account name can be root or any other privileged account, such as "jack" on the Live CD, or an account that you created during the installation.

---

8 **Verify that the system is at run level S.**

```
# who -r
.       run-level S  Jun 13 11:07     S      0  0
```

9 **Perform the system maintenance task that required the run level change to S.**

10 **After you complete the system maintenance task, reboot the system.**

▼ x86: How to Boot a System Interactively

Use this procedure to boot a system, if you need to specify an alternate kernel or an alternate `/etc/system` file.

Before You Begin To specify an alternate `/etc/system` file when booting an x86 based system interactively by using the `boot -a` command, follow these steps:

- 1. Make backup copies of the `/etc/system` and the `boot/solaris/filelist.ramdisk` files. For example:

  ```
  # cp /etc/system /etc/system.bak
  # cp /boot/solaris/filelist.ramdisk /boot/solaris/filelist.ramdisk.orig
  ```

- 2. Add the `/etc/system.bak` file name to the `/boot/solaris/filelist.ramdisk` file.

  ```
  # echo "etc/system.bak" >> /boot/solaris/filelist.ramdisk
  ```
- 3. Update the boot archive.

  ```
  # bootadm update-archive -v
  ```

1 **Reboot the system.**

```
# reboot
```

If the system displays the `Press any key to reboot` prompt, press any key to reboot the system.

You can also use the Reset button at this prompt. If the system is shut down, turn the system on with the power switch.

When the boot sequence begins, the GRUB main menu is displayed.

2 **To access the GRUB edit menu, type `e`.**

3 **Use the arrow keys to select the `kernel` or `kernel$` line.**

4 **Type `e` to edit the boot entry line.**

5 **Type `-a` to boot the system interactively, then, press Enter to return to the previous menu.**

6 **To boot the system interactively, type `b`.**

7 **Type a default directory for modules, or press Enter to accept the default.**

```
Enter default directory for modules [/platform/i86pc/kernel /kernel /usr/kernel]:
```

8 **Type an alternate system file name,** *alternate-file***.**

```
Name of system file [etc/system]: /etc/system.bak
```

Pressing Enter without providing an alternate file accepts the default.

Repair the damaged `/etc/system` file.

9 **Reboot the system to run level 3.**

x86: Booting From a Specified ZFS Root File System

The `$ZFS-BOOTFS` keyword enables you to boot from an Oracle Solaris ZFS root file system on the x86 platform. If a root device contains a ZFS pool, this keyword is assigned a value, which is then passed to the kernel with the `-B` option. This option identifies which dataset to boot. If you install an Oracle Solaris release that supports a ZFS boot loader, the GRUB `menu.lst` file, as well as the GRUB boot menu, contain this information by default.

▼ x86: How to Display a List of the Available ZFS Boot Environments

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2 **To display a list of available BEs on the system, type the following command:**

```
# bootadm list-menu
```

Example 7–7 x86: Displaying a List of Available ZFS Bootable Datasets by Using the `bootadm` Command

In this example, the output of the `bootadm list-menu` command lists all of the available bootable datasets (or boot entries) that are on an x86 based system. The default boot environment is listed first.

```
# bootadm list-menu
the location for the active GRUB menu is: /rpool/boot/grub/menu.lst
default 0
timeout 30
0 Oracle Solaris 10 9/10
1 Oracle Solaris 11 Express
2 zfsbe2
3 BE3
```

▼ x86: How to Boot From a Specified ZFS Root File System

This procedure describes how to boot from a ZFS root file system on an x86 system that supports a ZFS boot loader.

Note that if you install or upgrade your system to an Oracle Solaris release that supports a ZFS boot loader, the GRUB menu entry contains the `-B $ZFS-BOOTFS` boot argument by default, so the system boots from ZFS without requiring any additional boot arguments.

1 **Reboot the system.**

```
# reboot
```

If the system displays the `Press any key to reboot` prompt, press any key to reboot the system.

You can also use the Reset button at this prompt. If the system is shut down, turn the system on with the power switch.

When the boot sequence begins, the GRUB main menu is displayed. If the default boot entry is a ZFS file system menu is similar to the following:

```
findroot (pool_rpool,0,a)
bootfs rpool/ROOT/nwambe-1
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS,console=graphics
module$ /platform/i86pc/$ISADIR/boot_archive
```

2 **When the GRUB menu is displayed, press Enter to boot the default OS instance.**

If you do not choose an entry within 10 seconds, the system automatically boots to run level 3.

3 **To boot another BE, use the arrow keys to highlight the specified boot entry.**

4 **Type b to boot this entry or e to edit the entry.**

If you type e to edit the entry, the default menu for booting a system with a ZFS root would look similar to the following output:

```
findroot (BE_be10,0,a)
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/$ISADIR/boot-archive
```

For more information about GRUB menu entries at boot time, see“x86: How to Modify Boot Behavior by Editing the GRUB Menu at Boot Time” on page 88.

Using Fast Reboot (Task Map)

Task	Description	For Instructions
Initiate a fast reboot of a SPARC based system.	Use the reboot command with the -f option to initiate a fast reboot of the system.	“How to Initiate a Fast Reboot of a SPARC Based System” on page 113
Initiate a fast reboot of an x86 based system.	Because Fast Reboot is the default boot mode in this release, you can use either the reboot or the init 6 command to initiate a fast reboot of the system.	“How to Initiate a Fast Reboot of an x86 Based System” on page 114
Initiate a fast reboot of an x86 based system to a boot entry other than the default.	To reboot to a specific GRUB boot entry, use the reboot *n* command, where *n* is the number of the BE that you want to boot.	“x86: How to Fast Reboot a System to the *n*th Entry in the GRUB menu.lst File” on page 114
Initiate a fast reboot of an x86 based system by specifying an alternate BE.	You can fast reboot an x86 based system to an alternate BE by specifying the BE with the reboot command.	“x86: Initiating a Fast Reboot of a System by Specifying an Alternate Boot Environment” on page 115
Perform a slow reboot of a system.	Use the reboot command with -p option to perform a slow reboot of the system.	“Performing a Slow Reboot of a System” on page 116

Task	Description	For Instructions
Configure the default behavior for the Fast Reboot feature on the SPARC or x86 platform, and the Panic Fast Reboot feature on the x86 platform.	On the SPARC platform, Fast Reboot behavior is disabled by default. You can configure the `boot-config` service to perform a fast reboot of a SPARC based system by default. On the x86 platform, Fast Reboot and Panic Fast Reboot features are enabled on your system by default and are managed by the `boot-config` service. You can optionally disable either or both of these features.	"Managing the Boot Configuration Service" on page 116
Debug an early panic that might occur on an x86 based system.	Because the `boot-config` service has dependencies on the multiuser milestone, early panics on a system can occur. To debug possible early panics on the x86 platform, you can patch the global `fastreboot_onpanic` variable in the `/etc/system` file.	"x86: Debugging Early Panics That Might Occur" on page 117
Troubleshoot conditions that might prevent the Fast Reboot feature from working.	Under certain conditions, the fast reboot capability on the x86 platform does not work as expected. In some situations, a workaround is available.	"x86: Troubleshooting Conditions That Might Prevent Fast Reboot From Working" on page 117

Using Fast Reboot

The following tasks apply to both the SPARC and x86 platform, unless otherwise specified.

▼ How to Initiate a Fast Reboot of a SPARC Based System

Use the following procedure to fast reboot a SPARC based system when the `config/fastreboot_default` property of the `boot-config` service is set to `false`, which is the default behavior. To change the default behavior of the Fast Reboot feature, so that a fast reboot is automatically performed when the system reboots, see "Managing the Boot Configuration Service" on page 116.

1 **Assume the root role.**

2 **Initiate a fast reboot of the system by typing the following command:**

```
# reboot -f
```

▼ How to Initiate a Fast Reboot of an x86 Based System

Note – Starting with the Oracle Solaris 11 Express release, Fast Reboot and Panic Fast Reboot are the default operating mode on the x86 platform. Previously, to fast reboot an x86 based system the `-f` option needed to be specified with the `reboot`command. This option is no longer required to fast reboot a system.

---

1 **Assume the root role.**

2 **In a terminal window, initiate a fast reboot of the system by typing either of the following commands:**

```
# reboot
```

```
# init 6
```

▼ x86: How to Fast Reboot a System to the *n*th Entry in the GRUB `menu.lst` File

The boot entries in the `menu.lst` file have a corresponding number that you can specify with the `reboot` command.

1 **Assume the root role.**

2 **In a terminal window, list the boot entries that are on the system, along with their corresponding numbers.**

```
# bootadm list-menu
the location for the active GRUB menu is: /rpool/boot/grub/menu.lst
default 1
0 Oracle Solaris
1 Oracle Solaris 10
2 zfsbe2
3 BE3
```

3 **Fast reboot the system by specifying the number of the boot entry.**

`# reboot` *entry-number*

For example, to reboot the entry, `zfsbe2`, you would type:

```
# reboot 2
```

Example 7–8 x86: Fast Rebooting a System to the *n*th Entry in the `menu.lst` File

The following example shows how to fast reboot to the `zfsbe2` BE by specifying its corresponding entry number.

```
# bootadm list-menu
the location for the active GRUB menu is: /rpool/boot/grub/menu.lst
default 1
0 Oracle Solaris 11
1 Oralce Solaris 10
2 zfsbe2
3 BE3

# reboot 2
```

x86: Initiating a Fast Reboot of a System by Specifying an Alternate Boot Environment

There are several ways that you can perform a fast reboot of an x86 based system to an alternate BE. The following examples illustrate some of these methods.

EXAMPLE 7–9 x86: Fast Rebooting a System by Specifying a Newly Activated BE

The following example shows how to fast reboot a system to a newly activated BE, `oraclesolaris-4`.

```
# bootadm list-menu
the location for the active GRUB menu is: /rpool/boot/grub/menu.lst
default 1
0 oracle solaris
1 oraclesolaris-4
2 zfsbe2
3 BE3

# beactivate oraclesolaris-4
# reboot
```

EXAMPLE 7–10 x86: Fast Rebooting a System by Specifying an Alternate BE

To fast reboot a system to an alternate BE, for example `zfsbe1`, you would type:

```
# reboot -- 'rpool/zfsbe1'
```

To fast reboot a system to a dataset named `rpool/zfsbe1`, you would type:

```
# reboot -- 'rpool/zfsbe1'
```

To fast reboot a system to an alternate ZFS root dataset, you would type:

```
# reboot -- 'rpool/ROOT/zfsroot2'
```

EXAMPLE 7–11 x86: Fast Rebooting a System to an Alternate BE in 64–Bit Mode

To fast reboot a system to the `zfsbe3` BE, in 64-bit mode with the kernel debugger enabled, you would type:

EXAMPLE 7–11 x86: Fast Rebooting a System to an Alternate BE in 64–Bit Mode *(Continued)*

```
# reboot -- 'rpool/zfsbe3 /platform/i86pc/kernel/amd64/unix -k'
```

EXAMPLE 7–12 x86: Fast Rebooting a System to a New 64–Bit Kernel

To fast reboot a system to a new 64–bit kernel named `my-kernel`, you would type:

```
# reboot -- '/platform/i86pc/my-kernel/amd64/unix -k'
```

EXAMPLE 7–13 x86: Performing a Fast Reboot of a Mounted Disk or a Mounted Dataset

To perform a fast reboot of a mounted disk or a mounted dataset, you would type:

```
# reboot -- '/mnt/platform/i86pc/my-kernel/amd64/unix -k'
```

EXAMPLE 7–14 x86: Fast Rebooting a System in Single-User Mode With the Kernel Debugger Enabled

To fast reboot a system in single-user mode with the kernel debugger enabled, you would type:

```
# reboot -- '-ks'
```

Performing a Slow Reboot of a System

To reboot a SPARC or x86 based system that has Fast Reboot enabled by default, without reconfiguring the `boot-config` service to disable the feature, use the `-p` option with the `reboot` command, as shown in the following example:

```
# reboot -p
```

Managing the Boot Configuration Service

Starting with the Oracle Solaris 11 Express release, Fast Reboot is supported on the SPARC platform, as well as the x86 platform. On both platforms, this feature is controlled by the SMF and implemented through a boot configuration service, `svc:/system/boot-config`. The `boot-config` service provides a means for setting or changing the default boot configuration parameters.

The `fastreboot_default` property of the `boot-config` service enables an automatic fast reboot of the system when either the `reboot` or the `init 6` command is used. When the `config/fastreboot_default` property is set to `true` the system automatically performs a fast reboot, without the need to use the `reboot -f` command. By default, this property's value is set to `false` on the SPARC platform and to `true` on the x86 platform.

On the x86 platform, an additional property, `config/fastreboot_onpanic`, is implemented by default. This property enables a fast reboot of an x86 system in the event of a system panic.

The properties that are part of the `boot-config` service can be configured by using the `svccfg` and `svcadm` commands.

The following example shows how to set the property's value to `true` on the SPARC platform, so that a fast reboot is initiated by default:

```
# svccfg -s "system/boot-config:default" setprop config/fastreboot_default=true
# svcadm refresh svc:/system/boot-config:default
```

The following example shows how to disable the default behavior of the `fastreboot_onpanic` property on an x86 based system by setting the property's value to `false`:

```
# svccfg -s "system/boot-config:default" setprop config/fastreboot_onpanic=false
# svcadm refresh svc:/system/boot-config:default
```

Note that on the x86 platform, changing one property's default behavior does not affect the default behavior of the other property.

For information about managing the boot configuration service through the SMF, see the `svcadm`(1M) and the `svccfg`(1M) man pages.

x86: Debugging Early Panics That Might Occur

Because the `boot-config` service has dependencies on the multiuser milestone, users who need to debug early panics can patch a global variable, `fastreboot_onpanic` in the `/etc/system` file, as shown in the following example:

```
# echo "set fastreboot_onpanic=1" #x26;#x26;#x3e;#x26;#x26;#x3e; /etc/system
# echo "fastreboot_onpanic/W" | mdb -kw
```

x86: Troubleshooting Conditions That Might Prevent Fast Reboot From Working

The following are possible conditions under which the Fast Reboot feature might not work:

- GRUB menu cannot be processed.
- Driver does not implement the `quiesce` function.

  If you attempt a fast reboot of a system with an unsupported driver, a message similar to the following is displayed:

  ```
  Sep 18 13:19:12 too-cool genunix: WARNING: nvidia has no quiesce()
  reboot: not all drivers have implemented quiesce(9E)
  ```

  If the graphics drivers are the only drivers that do not support the `quiesce` function, you can attempt to force a fast reboot by running the following commands:

```
# echo "force_fastreboot/W 1" | mdb -kw# echo "set force_fast \
reboot = 1" #x26;#x26;#x3e;#x26;#x26;#x3e; /etc/system
```

Note – If the driver for the network interface card (NIC) does not implement the `quiesce` function, try to unplumb the interface first, then attempt a fast reboot of the system.

- Insufficient memory

  If there is not enough memory on the system, below 1G (0x40000000) for building the page tables, or not enough free memory to load the new kernel and the boot archive, the fast reboot attempt fails with the following messages, then falls back to a regular reboot.

  ```
  Fastboot: Couldn't allocate size below PA 1G to do fast reboot
  Fastboot: Couldn't allocate size below PA 64G to do fast reboot
  ```
- Unsupported environment

  Fast reboot functionality is not supported in the following environments:

  - An Oracle Solaris release that is running as a paravirtualized (PV) guest domain
  - Non-global zones

For more information, see the following man pages:

- `reboot`(1M)
- `init`(1M)
- `quiesce`(9E)
- `uadmin`(2)
- `dev_ops`(9S)

Booting an x86 Based System from the Network

This section describes the requirements and warnings for performing a GRUB based boot from the network.

Any system can boot from the network, if a boot server is available. You might need to boot a stand-alone system from the network for recovery purposes if the system cannot boot from the local disk. You can boot an x86 based system directly from a network that supports the PXE network boot protocol.

Note – The PXE network boot is available only for devices that implement the Intel Preboot Execution Environment specification.

The default network boot strategy that is used for a GRUB based PXE network boot is DHCP. For non-PXE devices, you can use either the DHCP or the RARP boot strategy. The strategy that you use depends on which type of boot server is available on your network. If no PXE or DHCP server is available, you can load GRUB from a diskette, a CD-ROM, or a local disk.

To perform a GRUB based network boot, a DHCP server that is configured for PXE clients is required. A boot server that provides `tftp` service is also required. The DHCP server supplies the information that the client needs to configure its network interface.

The DHCP server must be able to respond to the DHCP classes, `PXEClient` and `GRUBClient` with the following information:

- IP address of the file server
- Name of the boot file (`pxegrub`)

The sequence for performing a PXE network boot of the Oracle Solaris OS is as follows:

1. The BIOS is configured to boot from a network interface.
2. The BIOS sends a DHCP request.
3. The DHCP server replies with the server address and the name of the boot file.
4. The BIOS downloads `pxegrub` by using `tftp` and executes `pxegrub`.
5. The system downloads a GRUB menu file by using `tftp`.

   This file displays the boot menu entries that are available.
6. After you select a menu entry, the system begins to load the Oracle Solaris OS.

See “How to Set Up a Network Configuration Server” in *System Administration Guide: IP Services* for more information.

Running the `installadm` command creates the `/tftpboot_01`*ethernet-address* file. This file is linked to `pxegrub` and the`/tftpboot/menu.lst.01`*ethernet-address* file. The `/tftpboot/menu.lst.01`*ethernet-address* file is the GRUB menu file. If this file does not exist, then `pxegrub` reverts to using DHCP Option 150, if this option is specified, or the `/tftpboot/boot/grub/menu.lst` file. Typically, a single system is set up to serve both functions. In this instance, the `installadm` command sets up the `/tftpboot` file with the correct `pxegrub` menu file and the Oracle Solaris files. DHCP service is handled separately by using the `installadm` command. The setup only needs to be completed once per client. See “x86: About DHCP Macros” on page 119 and “x86: How to Perform a GRUB Based Boot From the Network” on page 121 for more information.

x86: About DHCP Macros

When you use the `installadm client-create` command to add clients on the install server, the command reports DHCP configuration information to standard output. You can use this information when you create the options and macros that are needed to pass network installation information to clients.

To install DHCP clients with a DHCP server over the network, you must create DHCP options. This information is needed to install the Oracle Solaris OS.

When a client sends a DHCP request, the server must have the following client information:

- Client's ID, which is typically the Ethernet address
- Class of the client request
- Subnet on which the client resides

The DHCP server forms a response. This response is based on the following *macros*, which matches the client request:

class macro	The class macro is based on a *class string* that is contained in the DHCP request. On x86 based systems, the BIOS already makes a DHCP request with the class `PXEClient:Arch:00000:UNDI:002001`. If a macro by this name is defined in the DHCP server configuration, then the macro content is sent to the x86 based clients.
network macro	The network macro is named by the IP address of the subnet that the client resides on. If the macro `129.146.87.0` is defined on the DHPC server, the macro content is sent to all clients on that subnet. The macro content is sent, regardless of the class of the request. If an option is defined in both the class macro and the network macro, the network macro takes precedence.
IP macro	The IP macro is named by an IP address. This macro is rarely used.
client macro	The client macro is named by the client type (01 for Ethernet) and the MAC address of the client, in uppercase letters. For a client with the Ethernet address `0:0:39:fc:f2:ef`, the corresponding macro name is `01000039FCEF`. Note the absence of colons in the client macro.

For example, for a client on the subnet `192.168.100.0`, with the Ethernet address `0:0:39:fc:f2:ef`, making a DHCP request of class `PXEClient`, the DHCP server has the following matching macro:

```
PXEClient
    BootSrvA:  192.168.100.0
    BootFile:  pxegrub
  129.146.87.0
    Router:    129.146.87.1
    NISdmain:  sunsoft.eng.sun.com
  01000039FCEF
    BootFile:  01000039FCEF
The actual DHCP response will be
    BootSrvA:  192.168.100.0
    BootFile:  01000039FCEF
    Router:    129.146.87.1
    NISdmain:  sunsoft.eng.sun.com
```

Note that the `BootFile` in the client macro overrides the `BootFile` in the class macro.

For more detailed information, see Chapter 6, "Setting Up DHCP for AI," in *Oracle Solaris 11 Express Automated Installer Guide*.

▼ x86: How to Perform a GRUB Based Boot From the Network

To perform a GRUB based network boot, a DHCP server that is configured for PXE clients is required. A boot server that provides `tftp` service is also required. The DHCP server must be able respond to the DHCP classes, `PXEClient` and `GRUBClient`, to obtain the IP address of the file server and the boot file (`pxegrub`). By default, the menu file is `/tftpboot/menu.lst.01`*ethernet-address*. If this file does not exist, then `pxegrub` reverts to DHCP Option 150, if this option is specified, or the `/tftpboot/boot/grub/menu.lst` file.

If you are booting the system from the Solaris software media, the system boots automatically.

Before You Begin Before performing a network boot on an x86 based system with GRUB, do the following:

- Run the appropriate commands on the installation server to enable the system to boot from the network.
- Add the client system as an install client.

For more information, see *Oracle Solaris 11 Express Automated Installer Guide*.

1 **On the DHCP server, create a client macro for the DHCP service with the following two options:**

- `BootSrvA`: *svr-addr*
- `BootFile`: *client-macro*

  Note that you must have superuser privileges on the DHCP server to run the `dhtadm` command.

  where *svr-addr* is the IP address of the server, and *client-macro* is named by the client's Ethernet type (01) and the MAC address, in uppercase letters. This number is also the name of the file that is used in the `/tftpboot` directory on the installation server.

Note – The notation for the *client-macro* should not contain any colons.

You can create the client macro from the DHCP GUI or from command-line interface.

To create the client macro from the command-line, type:

```
# dhtadm -[MA] -m client macro -d
":BootFile=client-macro:BootSrvA=svr-addr:"
```

2. **Reboot the system.**
3. **Instruct the BIOS to boot from the network.**
   - **If your system uses a specific keystroke sequence to boot from the network, type the keystrokes when the BIOS screen is displayed.**
   - **If you need to manually modify the BIOS settings to boot from the network, type the keystroke sequence to access the BIOS setup utility. Then, modify the boot priority to boot from the network.**
4. **When the GRUB menu is displayed, select the network installation image that you want to install.**

CHAPTER 8

Troubleshooting Booting an Oracle Solaris System (Tasks)

This chapter describes the procedures for booting Oracle Solaris on SPARC and x86 based systems.

The following is a list of the information that is in this chapter:

- "Troubleshooting Booting on the SPARC Platform (Task Map)" on page 123
- "Troubleshooting Booting on the x86 Platform (Task Map)" on page 130

Troubleshooting Booting on the SPARC Platform (Task Map)

Task	Description	For Instructions
Stop a system for recovery purposes.	If a damaged file is preventing the system from booting normally, first stop the system to attempt recovery.	"SPARC: How to Stop the System for Recovery Purposes" on page 124
Force a crash dump of and reboot of the system.	You can force a crash dump and reboot of the system as a troubleshooting measure.	"SPARC: How to Force a Crash Dump and Reboot of the System" on page 125
Boot a SPARC based system for recovery purposes.	Boot to repair an important system file that is preventing the system from booting successfully.	"SPARC: How to Boot a System for Recovery Purposes" on page 126
Boot a SPARC based system that has an Oracle Solaris ZFS root for recovery purposes.	Boot a system to recover the `root` password or a similar problem that prevents you from successfully logging into an Oracle Solaris ZFS root environment, you will need to boot failsafe mode or boot from alternate media, depending on the severity of the error. **Note –** The failsafe mode is not currently supported.	"SPARC: How to Boot a ZFS Root Environment to Recover From a Lost Password or Similar Problem" on page 128

Task	Description	For Instructions
Boot a system with the kernel debugger.	You can the system with the kernel debugger to troubleshoot booting problems. Use the `kmdb` command to boot the system.	"SPARC: How to Boot the System With the Kernel Debugger (`kmdb`)" on page 129

You might need to use one or more of the following methods to troubleshoot problems that prevent the system from booting successfully.

- Troubleshoot error messages when the system boots.
- Stop the system to attempt recovery.
- Boot a system for recovery purposes.
- Force a crash dump and reboot of the system.
- Boot the system with the kernel debugger by using the `kmdb` command.

Troubleshooting Booting on the SPARC Platform

▼ SPARC: How to Stop the System for Recovery Purposes

1 **Type the Stop key sequence for your system.**

The monitor displays the `ok` PROM prompt.

```
ok
```

The specific Stop key sequence depends on your keyboard type. For example, you can press Stop-A or L1-A. On terminals, press the Break key.

2 **Synchronize the file systems.**

```
ok sync
```

3 **When you see the `syncing file systems...` message, press the Stop key sequence again.**

4 **Type the appropriate `boot` command to start the boot process.**

For more information, see the `boot`(1M) man page.

5 **Verify that the system was booted to the specified run level.**

```
# who -r
 .       run-level s  May  2 07:39     3      0  S
```

Example 8–1 SPARC: Stopping the System for Recovery Purposes

```
    Press Stop-A
ok sync
```

```
syncing file systems...
        Press Stop-A
ok boot
```

SPARC: Forcing a Crash Dump and Reboot of the System

Forcing a crash dump and reboot of the system are sometimes necessary for troubleshooting purposes. The `savecore` feature is enabled by default.

For more information about system crash dumps, see Chapter 8, "Managing System Crash Information (Tasks)," in *System Administration Guide: Advanced Administration*.

▼ SPARC: How to Force a Crash Dump and Reboot of the System

Use this procedure to force a crash dump of the system. The example that follows this procedure shows how to use the `halt -d` command to force a crash dump of the system. You will need to manually reboot the system after running this command.

1 **Type the stop key sequence for your system.**

The specific stop key sequence depends on your keyboard type. For example, you can press Stop-A or L1-A. On terminals, press the Break key.

The PROM displays the `ok` prompt.

2 **Synchronize the file systems and write the crash dump.**

```
> n
ok sync
```

After the crash dump is written to disk, the system will continue to reboot.

3 **Verify the system boots to run level 3.**

The login prompt is displayed when the boot process has finished successfully.

hostname `console login:`

Example 8–2 SPARC: Forcing a Crash Dump and Reboot of the System by Using the `halt -d` Command

This example shows how to force a crash dump and reboot of the system `jupiter` by using the `halt -d` and `boot` command. Use this method to force a crash dump and reboot of the system.

```
# halt -d
Jul 21 14:13:37 jupiter halt: halted by root
```

```
panic[cpu0]/thread=30001193b20: forced crash dump initiated at user request

000002a1008f7860 genunix:kadmin+438 (b4, 0, 0, 0, 5, 0)
  %l0-3: 0000000000000000 0000000000000000 0000000000000004 0000000000000004
  %l4-7: 00000000000003cc 0000000000000010 0000000000000004 0000000000000004
000002a1008f7920 genunix:uadmin+110 (5, 0, 0, 6d7000, ff00, 4)
  %l0-3: 0000030002216938 0000000000000000 0000000000000001 0000004237922872
  %l4-7: 000000423791e770 0000000000004102 0000030000449308 0000000000000005

syncing file systems... 1 1 done
dumping to /dev/dsk/c0t0d0s1, offset 107413504, content: kernel
100% done: 5339 pages dumped, compression ratio 2.68, dump succeeded
Program terminated
ok boot
Resetting ...

Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 333MHz), No Keyboard
OpenBoot 3.15, 128 MB memory installed, Serial #10933339.
Ethernet address 8:0:20:a6:d4:5b, Host ID: 80a6d45b.
.
.
.
Rebooting with command: boot
Boot device: /pci@1f,0/pci@1,1/ide@3/disk@0,0:a
File and args: kernel/sparcv9/unix
configuring IPv4 interfaces: hme0.
add net default: gateway 172.20.27.248
Hostname: jupiter
The system is coming up.  Please wait.
NIS domain name is example.com
.
.
.
System dump time: Wed Jul 21 14:13:41 2004
Jul 21 14:15:23 jupiter savecore: saving system crash dump
in /var/crash/jupiter/*.0
Constructing namelist /var/crash/jupiter/unix.0
Constructing corefile /var/crash/jupiter/vmcore.0
100% done: 5339 of 5339 pages saved

Starting Sun(TM) Web Console Version 2.1-dev...
.
.
.
```

▼ SPARC: How to Boot a System for Recovery Purposes

Use this procedure when an important file, such as `/etc/passwd`, has an invalid entry and causes the boot process to fail.

Use the stop sequence described in this procedure if you do not know the `root` password or if you can't log in to the system. For more information, see "SPARC: How to Stop the System for Recovery Purposes" on page 124.

Substitute the device name of the file system to be repaired for the *device-name* variable in the following procedure. If you need help identifying a system's device names, refer to "Displaying Device Configuration Information" in *System Administration Guide: Devices and File Systems*.

1. **Stop the system by using the system's Stop key sequence.**

2. **Boot the system in single-user mode.**
   - Boot the system from the Oracle Solaris installation media:
     - Insert the Oracle Solaris installation media into the drive.
     - Boot from the installation media in single-user mode.

       ```
       ok boot cdrom -s
       ```
   - Boot the system from the network if an installation server or remote CD or DVD drive is not available.

     ```
     ok boot net -s
     ```

3. **Mount the file system that contains the file with an invalid entry.**

   ```
   # mount /dev/dsk/device-name /a
   ```

4. **Change to the newly mounted file system.**

   ```
   # cd /a/file-system
   ```

5. **Set the terminal type.**

   ```
   # TERM=sun
   # export TERM
   ```

6. **Remove the invalid entry from the file by using an editor.**

   ```
   # vi filename
   ```

7. **Change to the root (/) directory.**

   ```
   # cd /
   ```

8. **Unmount the /a directory.**

   ```
   # umount /a
   ```

9. **Reboot the system.**

   ```
   # init 6
   ```

10. **Verify that the system booted to run level 3.**

    The login prompt is displayed when the boot process has finished successfully.

    ```
    hostname console login:
    ```

Example 8–3 SPARC: Booting a System for Recovery Purposes (Damaged Password File)

The following example shows how to repair an important system file (in this case, `/etc/passwd`) after booting from a local CD-ROM.

```
ok boot cdrom -s
# mount /dev/dsk/c0t3d0s0 /a
# cd /a/etc
# TERM=vt100
# export TERM
# vi passwd
    (Remove invalid entry)
# cd /
# umount /a
# init 6
```

Example 8–4 SPARC: Booting a System If You Forgot the `root` Password

The following example shows how to boot the system from the network when you have forgotten the `root` password. This example assumes that the network boot server is already available. Be sure to apply a new `root` password after the system has rebooted.

```
ok boot net -s
# mount /dev/dsk/c0t3d0s0 /a
# cd /a/etc
# TERM=vt100
# export TERM
# vi shadow
    (Remove root's encrypted password string)
# cd /
# umount /a
# init 6
```

▼ SPARC: How to Boot a ZFS Root Environment to Recover From a Lost Password or Similar Problem

1 **Boot the system in single-user mode either from an installation CD or from the network.**

```
ok boot cdrom -s
```

Note – If you do not use the `-s` option, you will need to exit the installation program.

---

2 **Become the root user.**

3 **Import the root pool and specify an alternate mount point.**

```
# zpool import -R /a rpool
```

4 **Mount the ZFS BE.**

```
# zfs mount rpool/ROOT/zfsBE
```

The ZFS BE must be specifically mounted because the `canmount` property is set to `noauto` by default.

5 **Change to the `/a/etc` directory.**

```
# cd /a/etc
```

6 **Using a text editor, correct the `passwd` or `shadow` file. For example:**

```
# vi passwd
```

7 **Reboot the system.**

```
# init 6
```

▼ SPARC: How to Boot the System With the Kernel Debugger (`kmdb`)

This procedure shows you the basics for loading the kernel debugger (`kmdb`). For more detailed information, see the *Oracle Solaris Modular Debugger Guide*.

Note – Use the `reboot` and `halt` command with the `-d` option if you do not have time to debug the system interactively. To run the `halt` command with the `-d` option requires a manual reboot of the system afterwards. Whereas, if you use the `reboot` command, the system boots automatically. See the `reboot`(1M) for more information.

---

1 **Halt the system, causing it to display the `ok` prompt.**

To halt the system gracefully, use the `/usr/sbin/halt` command.

2 **Type either `boot kmdb` or `boot -k` to request the loading of the kernel debugger. Press return.**

3 **Enter the kernel debugger.**

The method used to enter the debugger is dependent upon the type of console that is used to access the system:

- If a locally attached keyboard is being used, press Stop-A or L1–A, depending upon the type of keyboard.
- If a serial console is being used, send a break by using the method that is appropriate for the type of serial console that is being used.

A welcome message is displayed when you enter the kernel debugger for the first time.

```
Rebooting with command: kadb
Boot device: /iommu/sbus/espdma@4,800000/esp@4,8800000/sd@3,0
.
.
.
```

Example 8–5 SPARC: Booting a System With the Kernel Debugger (`kmdb`)

```
ok boot kmdb
Resetting...

Executing last command: boot kmdb -d
Boot device: /pci@1f,0/ide@d/disk@0,0:a File and args: kmdb -d
Loading kmdb...
```

Troubleshooting Booting on the x86 Platform (Task Map)

Task	Description	For Instructions
Stop a system for recovery purposes.	If a damaged file is preventing the system from booting normally, first stop the system to attempt recovery.	"x86: How to Stop a System for Recovery Purposes" on page 130
Force a crash dump of and reboot of the system.	You can force a crash dump and reboot of the system as a troubleshooting measure.	"x86: How to Force a Crash Dump and Reboot of the System" on page 131
Boot a system with the kernel debugger.	You can the system with the kernel debugger to troubleshoot booting problems. Use the `kmdb` command to boot the system.	"x86: How to Boot a System With the Kernel Debugger in the GRUB Boot Environment (`kmdb`)" on page 132

Troubleshooting on the x86 Platform

▼ x86: How to Stop a System for Recovery Purposes

1 Stop the system by using one of the following commands, if possible:

- If the keyboard and mouse are functional, become an administrator. Then, type `init 0` to stop the system. After the `Press any key to reboot` prompt appears, press any key to reboot the system.
- If the keyboard and mouse are functional, become an administrator, then, type `init 6` to reboot the system.

2 **If the system does not respond to any input from the mouse or the keyboard, press the Reset key, if it exists, to reboot the system.**

Or, you can use the power switch to reboot the system.

x86: Forcing a Crash Dump and Reboot of the System

Forcing a crash dump and reboot of the system are sometimes necessary for troubleshooting purposes. The `savecore` feature is enabled by default.

For more information about system crash dumps, see Chapter 8, "Managing System Crash Information (Tasks)," in *System Administration Guide: Advanced Administration*.

▼ x86: How to Force a Crash Dump and Reboot of the System

If you cannot use the `reboot -d` or the `halt -d` command, you can use the kernel debugger, `kmdb`, to force a crash dump. The kernel debugger must have been loaded, either at boot, or with the `mdb -k` command, for the following procedure to work.

Note – You must be in text mode to access the kernel debugger (`kmdb`). So, first exit any window system.

---

1 **Access the kernel debugger.**

The method used to access the debugger is dependent upon the type of console that you are using to access the system.

- If you are using a locally attached keyboard, press F1–A.
- If you are using a serial console, send a break by using the method appropriate to that type of serial console.

The `kmdb` prompt is displayed.

2 **To induce a crash, use the `systemdump` macro.**

```
[0]> $<systemdump
```

Panic messages are displayed, the crash dump is saved, and the system reboots.

3 **Verify that the system has rebooted by logging in at the console login prompt.**

Example 8–6 x86: Forcing a Crash Dump and Reboot of the System by Using `halt -d`

This example shows how to force a crash dump and reboot of the x86 based system `neptune` by using the `halt -d` and `boot` commands. Use this method to force a crash dump of the system. Reboot the system afterwards manually.

```
# halt -d
4ay 30 15:35:15 wacked.Central.Sun.COM halt: halted by user

panic[cpu0]/thread=ffffffff83246ec0: forced crash dump initiated at user request

fffffe80006bbd60 genunix:kadmin+4c1 ()
fffffe80006bbec0 genunix:uadmin+93 ()
fffffe80006bbf10 unix:sys_syscall32+101 ()

syncing file systems... done
dumping to /dev/dsk/c1t0d0s1, offset 107675648, content: kernel
NOTICE: adpu320: bus reset
100% done: 38438 pages dumped, compression ratio 4.29, dump succeeded

Welcome to kmdb
Loaded modules: [ audiosup crypto ufs unix krtld s1394 sppp nca uhci lofs
genunix ip usba specfs nfs md random sctp ]
[0]>
kmdb: Do you really want to reboot? (y/n) y
```

▼ x86: How to Boot a System With the Kernel Debugger in the GRUB Boot Environment (kmdb)

This procedure shows the basics for loading the kernel debugger (`kmdb`). The `savecore` feature is enabled by default. For more detailed information about using the kernel debugger, see the *Oracle Solaris Modular Debugger Guide*.

1 **Boot the system.**

The GRUB menu is displayed when the system is booted.

2 **When the GRUB menu is displayed, type e to access the GRUB edit menu.**

3 **Use the arrow keys to select the `kernel$` line.**

If you cannot use the arrow keys, use the ^ key to scroll up and the v key to scroll down.

4 **Type e to edit the line.**

The boot entry menu is displayed. In this menu, you can modify boot behavior by adding additional boot arguments to the end of the `kernel$` line.

5 **Type `-k` at the end of the line.**

6 **Press enter to return to the GRUB main menu.**

7 **Type b to boot the system with the kernel debugger enabled.**

8 **Access the kernel debugger.**

The method used to access the debugger is dependent upon the type of console that you are using to access the system:

- If you are using a locally attached keyboard, press F1–A.
- If you are using a serial console, send a break by using the method appropriate to that type of serial console.

A welcome message is displayed when you access the kernel debugger for the first time.

Example 8–7 x86: Booting a System With the Kernel Debugger (Support for Directly Loading and Booting the `unix` Kernel)

The following examples apply to an Oracle Solaris release with GRUB support for directly loading and booting the `unix` kernel.

This example shows how to boot a 64-bit capable x86 based system, with the kernel debugger enabled.

```
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS -k
```

This example shows how to boot a 64-bit capable x86 based system in 32-bit mode, with the kernel debugger enabled.

```
kernel$ /platform/i86pc/kernel/unix -B $ZFS-BOOTFS -k
```

◆ ◆ ◆ CHAPTER 9

Managing the Oracle Solaris Boot Archives (Tasks)

This chapter describes how the Oracle Solaris boot archives are managed. Procedures for using the `bootadm` command are described in detail.

The following is a list of the information that is in this chapter:

For overview information about the boot process, see Chapter 4, "Shutting Down and Booting a System (Overview)." For step-by-step instructions on booting a system, see Chapter 7, "Booting an Oracle Solaris System (Tasks)."

Managing the Oracle Solaris Boot Archives (Task Map)

TABLE 9–1 Oracle Solaris Boot Archive Management: Task Map

Task	Description	For Information
x86: Clear a boot archive update failure by using the `auto-reboot-safe` property.	Use this procedure in cases where the boot archive update on an x86 based system fails because the `auto-reboot-safe` property is set to `false`.	"x86: How to Clear Automatic Boot Archive Update Failures by Using the `auto-reboot-safe` Property" on page 138
Clear a boot archive update failure by using the `bootadm` command.	Use this procedure to manually clear boot archive update failures on the SPARC platform, and on the x86 platform, if the `auto-reboot-safe` property is set to `true`.	"How to Clear Automatic Boot Archive Update Failures by Using the `bootadm` Command" on page 139

TABLE 9–1 Oracle Solaris Boot Archive Management: Task Map *(Continued)*

Task	Description	For Information
Manage the `boot-archive` service.	The `boot-archive` service is controlled by the Service Management Facility (SMF). Use the `svcadm` command to enable and disable services. Use the `svcs` command to verify whether the `boot-archive` service is running.	"Managing the `boot-archive` Service" on page 137
Update the boot archives by using the `bootadm` command.	Use the `bootadm update-archive` command to manually update the boot archive.	"How to Manually Update the Boot Archive by Using the `bootadm` Command" on page 140
List the contents of the boot archives by using the `bootadm` command.	Use the `bootadm list-archive` command to list the contents of the boot archive.	"How to List Contents of the Boot Archive" on page 140
x86 only: Locate the active GRUB menu by using the `bootadm` command.	Use the `bootadm list-menu` command to determine the location of the active GRUB menu.	"x86: How to Locate the Active GRUB Menu and List Current Menu Entries" on page 141
x86 only: Set the default boot entry in the GRUB menu by using the `bootadm` command.	Use the `bootadm set-menu` command to set the default boot entry in the GRUB menu.	"x86: How to Set the Default Boot Entry for the Active GRUB Menu" on page 141

Description of the Oracle Solaris Boot Archives

When you install Oracle Solaris, the `bootadm` command creates a boot archive on your system.

A *boot archive* is a subset of a root (`/`) file system. This boot archive contains all of the kernel modules, `driver.conf` files, in addition to a few configuration files. These files are located in the `/etc` directory. The files in the boot archive are read by the kernel before the root (`/`) file system is mounted. After the root (`/`) file system is mounted, the boot archive is discarded by the kernel from memory. Then, file I/O is performed against the root device

---

Note – In some releases, two primary boot archives (one 32-bit archive and one 64-bit archive) are created at installation time. The 32-bit archive is located in `/platform/i86pc/boot_archive`. The 64-bit archive is located in `/platform/i86pc/amd64/boot_archive`.

The files that make up the SPARC boot archives are located in the `/platform` directory.

The contents of this directory are divided into three groups of files:

- Files that are required for a `sun4u` boot archive
- Files that are required for a `sun4v` boot archive

- Files that are required for a sun4us boot archive

The files that make up the x86 boot archives are located in the /platform/i86pc directory.

In certain releases, the contents of this directory are divided into two groups of files:

- 32-bit boot archive files, which are located in /platform/i86pc/boot_archive
- 64-bit boot archive files, which are located in /platform/i86pc/amd64/boot_archive

To list the files and directories that are included in the boot archives, use the bootadm list-archive command.

If any files in the archive are updated, the boot archive must be rebuilt. For modifications to take effect, the rebuild of the archive must take place before the next system reboot.

Managing the boot-archive Service

The boot-archive service is controlled by the Service Management Facility (SMF). The boot-archive service instance is svc:/system/boot-archive:default. The svcadm command is used to enable and disable services.

To determine if the boot-archive service is running, use the svcs command.

For more information, see the svcadm(1M) and the svcs(1) man pages.

▼ How to Enable or Disable the boot-archive Service

1 **Become an administrator.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2 **To enable or disable the boot-archive service, type:**

```
# svcadm enable | disable system/boot-archive
```

3 **To verify the state of the boot-archive service, type:**

```
% svcs boot-archive
```

If the service is running, the output displays an online service state.

```
STATE          STIME    FMRI
online          9:02:38 svc:/system/boot-archive:default
```

If the service is not running, the output indicates the service is offline.

Troubleshooting For information about clearing automatic boot archive update failures, see "Administering Automatic Boot Archive Recovery" on page 138.

Administering Automatic Boot Archive Recovery

Starting with the Oracle Solaris 11 Express release, boot archive recovery on the SPARC platform is fully automated. If you are running Oracle Solaris 11 Express, boot archive recovery on the x86 platform is automated through the Fast Reboot feature.

To support auto-recovery of the boot archives on the x86 platform, a new `auto-reboot-safe` property has been added to the boot configuration SMF service, `svc:/system/boot-config:default`. By default, this property's value is set to `false`, which prevents the system from automatically rebooting to an unknown boot device. However, if your system is configured to automatically reboot to the BIOS boot device and default GRUB menu entry that the Oracle Solaris OS is installed on, you can enable automatic recovery of the boot archives by setting this property's value to `true`. The following procedure describes how to clear automatic boot archive update failures on the x86 platform.

For information about how to clear automatic boot archive update failures by using the `bootadm` command, see "How to Clear Automatic Boot Archive Update Failures by Using the `bootadm` Command" on page 139.

▼ x86: How to Clear Automatic Boot Archive Update Failures by Using the `auto-reboot-safe` Property

On an x86 based systems, during the process of booting the system, if a warning similar to the following is displayed, take action as described in the following procedure.

```
WARNING: Reboot required.
The system has updated the cache of files (boot archive) that is used
during the early boot sequence. To avoid booting and running the system
with the previously out-of-sync version of these files, reboot the
system from the same device that was previously booted.
```

The system then enters system maintenance mode.

1. **Become the root user.**

2. **Reboot the system.**

   ```
   # reboot
   ```

   To prevent this type of failure, if the active BIOS boot device and the GRUB menu entries point to the current boot instance, do the following:

3. **Set the `auto-reboot-safe` property of the `svc:/system/boot-config` SMF service to `true`, as follows:**

   ```
   # svccfg -s svc:/system/boot-config:default setprop config/auto-reboot-safe = true
   ```

4 **Verify that the `auto-reboot-safe` property is set correctly.**

```
# svccfg -s svc:/system/boot-config:default listprop |grep config/auto-reboot-safe
config/auto-reboot-safe            boolean  true
```

▼ How to Clear Automatic Boot Archive Update Failures by Using the `bootadm` Command

During the process of booting the system, if a warning message that is similar to the following is displayed, take action accordingly:

```
WARNING: Automatic update of the boot archive failed.
Update the archives using 'bootadm update-archive'
command and then reboot the system from the same device that
was previously booted.
```

The following procedure describes how to manually update an out-of-date boot archive by using the `bootadm` command.

1 **Become the root user.**

2 **To update the boot archive, type:**

```
# bootadm update-archive
```

`bootadm`	Manages the boot archives on a system.
`update-archive`	Updates the current boot archive, if required. Applies to both SPARC and x86 based systems.

3 **Reboot the system.**

```
# reboot
```

Using the `bootadm` Command to Manage the Boot Archives

The `/sbin/bootadm` command enables you to perform the following tasks:

- Manually update the current boot archives on a system.
- List the files and directories that are included in the boot archives on a system.
- **x86 only:** Maintain the GRUB menu.
- **x86 only:** Locate the active GRUB menu, as well as the current GRUB menu entries.

The syntax of the command is as follows:

`/sbin/bootadm [`*subcommand***`] [-`***option***`] [-R`** ***altroot*****`]`**

For more information about the `bootadm` command, see the `bootadm`(1M) man page.

How to Manually Update the Boot Archive by Using the `bootadm` Command

1. **Become the root user.**

2. **To update the boot archive, type:**

```
# bootadm update-archive
```

`bootadm`	Manages the boot archives on a system.
`update-archive`	Updates the current boot archive, if required. Applies to both SPARC and x86 based systems.

- **To update the boot archive on an alternate root, type:**

```
# bootadm update-archive -R /a
```

`-R` *altroot*	Specifies an alternate `root` path to apply to the `update-archive` subcommand.

Note – The root (`/`) file system of any non-global zone must not be referenced with the `-R` option. Doing so might damage the global zone's file system, compromise the security of the global zone, or damage the non-global zone's file system. See the `zones`(5) man page.

3. **Reboot the system.**

```
# reboot
```

How to List Contents of the Boot Archive

1. **Become the root user.**

2. **To list the files and directories that are included in the boot archive, type:**

```
# bootadm list-archive
```

`list-archive`	Lists the files and directories that are included in the boot archive or archives. Applies to both SPARC and x86 based systems.

▼ x86: How to Locate the Active GRUB Menu and List Current Menu Entries

Use this procedure to determine the location of the active GRUB menu and to list current GRUB menu entries.

1. **Become the root user.**

2. **To list the location of the active GRUB menu and current GRUB menu entries, type:**

```
# bootadm list-menu
```

`list-menu`	Lists the location of the active GRUB menu, as well as the current GRUB menu entries. Information about the `autoboot-timeout`, the default entry number, and the title of each entry is included in this listing. Applies to x86 based systems *only*.

Example 9–1 Listing the Location of the Active GRUB Menu and Current GRUB Menu Entries

```
# bootadm list-menu
The location for the active GRUB menu is: /stubboot/boot/grub/menu.lst
default=0
timeout=30
(0) Oracle Solaris10
(1) Oracle Solaris11 Express
(2) Linux
```

▼ x86: How to Set the Default Boot Entry for the Active GRUB Menu

1. **Become the root user.**

2. **To set the default boot entry in the active GRUB menu, type:**

```
# bootadm set-menu menu-entry
```

`set-menu`	Maintains the GRUB menu. The location of the active GRUB menu is `boot/grub/menu.lst`. Applies to x86 bases systems *only*.
menu-entry	Specifies the GRUB menu entry to set as the default.

3. **To verify default menu entry has been changed, type:**

```
# bootadm list-menu
```

The new default menu entry should be displayed.

Example 9–2 Switching the GRUB Default Menu Entry

This example shows how to switch the default GRUB menu to one of the menu entries that is displayed in the previous example. The menu entry that is selected is The Linux, menu entry 2.

```
# bootadm set-menu default=2
```

◆ ◆ ◆ CHAPTER 10

x86: GRUB Based Booting (Reference)

This chapter contains information about x86 boot processes, including GRUB implementation details and additional GRUB reference information.

For overview information, see Chapter 4, “Shutting Down and Booting a System (Overview).”

For step-by-step instructions on booting a system, see Chapter 7, “Booting an Oracle Solaris System (Tasks).”

x86: Boot Processes

This section includes information about boot processes that are unique to booting an x86 based system.

x86: System BIOS

When an x86 based system is powered on, it is controlled by the read-only-memory (ROM) Basic Input/Output System (BIOS). The BIOS is the firmware interface on Oracle Solaris operating systems that have x86 64-bit and 32-bit support.

Hardware adapters usually have an on-board BIOS that displays the physical characteristics of the device. The BIOS is used to access the device. During the startup process, the system BIOS checks for the presence of any adapter BIOS. If any adapters are found, the system then loads and executes each adapter BIOS. Each adapter's BIOS runs self-test diagnostics and then displays device information.

The BIOS on most systems has a user interface, where you can select an ordered list of boot devices that consists of the following selections:

- Diskette
- CD or DVD

- Hard disk
- Network

The BIOS attempts to boot from each device, in turn, until a valid device with a bootable program is found.

x86: Support for GRUB in the Oracle Solaris OS

The following sections contain additional reference information for administering GRUB in the Oracle Solaris OS.

x86: GRUB Terminology

To thoroughly grasp GRUB concepts, an understanding of the following terms is essential.

Note – Some of the terms that are described in this list are not exclusive to GRUB based booting.

boot archive
A collection of critical files that is used to boot the Oracle Solaris OS. These files are needed during system startup before the root file system is mounted. Multiple boot archives are maintained on a system:

- A *primary boot archive* is used to boot the Oracle Solaris OS on an x86 based system.

  Note – On the x86 platform, when you install the Oracle Solaris, two primary boot archives are created, one 32-bit archive and one 64-bit archive.

boot loader
The first software program that runs after you power on a system. This program begins the booting process.

GRUB
GNU GRand Unified Bootloader (GRUB) is an open-source boot loader with a menu interface. The menu displays a list of the operating systems that are installed on a system. GRUB enables you to easily boot these various operating systems, such as the Oracle Solaris OS, Linux, or Windows.

GRUB main menu
A boot menu that lists the operating systems that are installed on a system. From this menu, you can easily boot an operating system without modifying the BIOS or `fdisk` partition settings.

GRUB edit menu	A submenu of the GRUB main menu. GRUB commands are displayed on this submenu. These commands can be edited to change boot behavior.
`menu.lst` **file**	A configuration file that lists all the operating systems that are installed on a system. The contents of this file dictate the list of operating systems that is displayed in the GRUB menu. From the GRUB menu, you can easily boot an operating system without modifying the BIOS or `fdisk` partition settings.
primary boot archive	See boot archive.

x86: Functional Components of GRUB

GRUB consists of the following functional components:

- `stage1` – Is an image that is installed on the first sector of the `fdisk` partition. You can optionally install `stage1` on the master boot sector by specifying the `-m` option with the `installgrub` command. See the `installgrub(1M)` man page and "Disk Management in the GRUB Boot Environment" in *System Administration Guide: Devices and File Systems* for more information.
- `stage2` – Is an image that is installed in a reserved area in the `fdisk` partition. The `stage2` image is the core image of GRUB.
- `menu.lst` file – Is typically located in the /*pool-name*/`boot/grub` directory on systems with a ZFS root. This file is read by the GRUB `stage2` file. For more information, see the section, "x86: Modifying Boot Behavior by Editing the `menu.lst` File" on page 89.

You cannot use the `dd` command to write `stage1` and `stage2` images to disk. The `stage1` image must be able to receive information about the location of the `stage2` image that is on the disk. Use the `installgrub` command, which is the supported method for installing GRUB boot blocks.

Naming Conventions That Are Used for Configuring GRUB

GRUB uses device-naming conventions that are slightly different from previous releases. Understanding the GRUB device-naming conventions can assist you in correctly specifying drive and partition information when you configure GRUB on your system.

The following table describes the GRUB device-naming conventions for this Oracle Solaris release.

TABLE 10–1 Conventions for GRUB Devices

Device Name	Description
`(fd0)`	First diskette
`(fd1)`	Second diskette
`(nd)`	Network device
`(hd0,0)`	First `fdisk` partition on first hard disk
`(hd0,1)`	Second `fdisk` partition on first hard disk
`(hd0,0,a)`,	Slice `a` on first `fdisk` partition on first hard disk
`(hd0,0,b)`	Slice `b` on first `fdisk` partition on first hard disk

Note – All GRUB device names must be enclosed in parentheses.

For more information about `fdisk` partitions, see "Guidelines for Creating an fdisk Partition" in *System Administration Guide: Devices and File Systems*.

Naming Conventions That Are Used by the `findroot` Command

Starting with the Solaris 10 10/08 release, the `findroot` command replaces the `root` command that was previously used by GRUB. The `findroot` command provides enhanced capabilities for discovering a targeted disk, regardless of the boot device. The `findroot` command also supports booting from a ZFS root file system.

The following is a description of the device naming convention that is used by the `findroot` command for various GRUB implementations:

- Standard system upgrades and new installations for systems with ZFS support:

  ```
  findroot(pool_p,0,a)
  ```

  The *p* variable is the name of the root pool.

◆ ◆ ◆ CHAPTER 11

Managing Services (Overview)

This chapter provides an overview of the Service Management Facility (SMF). In addition, information that is related to run levels is provided.

The following is a list of the information that is in this chapter:

For information on the procedures associated with SMF, see "Managing Services (Task Map)" on page 161. For information on the procedures associated with run levels, see "Using Run Control Scripts (Task Map)" on page 176.

Introduction to SMF

SMF provides an infrastructure that augments the traditional UNIX start-up scripts, `init` run levels, and configuration files. SMF provides the following functions:

- Automatically restarts failed services in dependency order, whether they failed as the result of administrator error, software bug, or were affected by an uncorrectable hardware error. The dependency order is defined by dependency statements.
- Makes services objects that can be viewed, with the new `svcs` command, and managed, with `svcadm` and `svccfg` commands. You can also view the relationships between services and processes using `svcs -p`, for both SMF services and legacy `init.d` scripts.
- Makes it easy to backup, restore, and undo changes to services by taking automatic snapshots of service configurations.

- Makes it easy to debug and ask questions about services by providing an explanation of why a service isn't running by using `svcs -x`. Also, this process is eased by individual and persistent log files for each service.
- Allows for services to be enabled and disabled using `svcadm`. These changes can persist through upgrades and reboots. If the `-t` option is used, the changes are temporary.
- Enhances the ability of administrators to securely delegate tasks to non-root users, including the ability to modify properties and enable, disable, or restart services on the system.
- Boots faster on large systems by starting services in parallel according to the dependencies of the services. The opposite process occurs during shutdown.
- Allows you to customize the boot console output to either be as quiet as possible, which is the default, or to be verbose by using `boot -m verbose`.
- Preserves compatibility with existing administrative practices wherever possible. For example, most customer and ISV-supplied rc scripts still work as usual.

Dependency statements define the relationships between services. These relationships can be used to provide precise fault containment by restarting only those services that are directly affected by a fault, rather than restarting all of the services. Another advantage of dependency statements is that the statements allow for scalable and reproducible initialization processes. In addition, by defining all of the dependencies, you can take advantage of modern, highly parallel machines, because all independent services can be started in parallel.

SMF defines a set of actions that can be invoked on a service by an administrator. These actions include enable, disable, refresh, restart, and maintain. Each service is managed by a service restarter which carries out the administrative actions. In general, the restarters carry out actions by executing methods for a service. Methods for each service are defined in the service configuration repository. These methods allow the restarter to move the service from one state to another state.

The service configuration repository provides a per-service snapshot at the time that each service is successfully started so that fallback is possible. In addition, the repository provides a consistent and persistent way to enable or disable a service, as well as a consistent view of service state. This capability helps you debug service configuration problems.

Changes in Behavior When Using SMF

Most of the features that are provided by SMF happen behind the scenes, so users are not aware of them. Other features are accessed by new commands. Here is a list of the behavior changes that are most visible.

- The boot process creates many fewer messages now. Services do not display a message by default when they are started. All of the information that was provided by the boot messages can now be found in a log file for each service that is in `/var/svc/log`. You can use the `svcs` command to help diagnose boot problems. In addition, you can use the `-v` option to the `boot` command, which generates a message when each service is started during the boot process.

- Since services are automatically restarted if possible, it may seem that a process refuses to die. If the service is defective, the service will be placed in maintenance mode, but normally a service is restarted if the process for the service is killed. The `svcadm` command should be used to stop the processes of any SMF service that should not be running.
- Many of the scripts in `/etc/init.d` and /etc/rc*.d have been removed. The scripts are no longer needed to enable or disable a service. Entries from `/etc/inittab` have also been removed, so that the services can be administered using SMF. Scripts and `inittab` entries that are provided by an ISV or are locally developed will continue to run. The services may not start at exactly the same point in the boot process, but they are not started before the SMF services, so that any service dependencies should be OK.

SMF Concepts

This section presents terms and their definitions within the SMF framework. These terms are used throughout the documentation. To grasp SMF concepts, an understanding of these terms is essential.

SMF Service

The fundamental unit of administration in the SMF framework is the *service instance*. Each SMF service has the potential to have multiple versions of it configured. As well, multiple instances of the same version can run on a single system. An *instance* is a specific configuration of a service. A web server is a service. A specific web server daemon that is configured to listen on port 80 is an instance. Each instance of the web server service could have different configuration requirements. The service has system-wide configuration requirements, but each instance can override specific requirements, as needed. Multiple instances of a single service are managed as child objects of the service object.

Services are not just the representation for standard long-running system services such as `in.dhcpd` or `nfsd`. Services also represent varied system entities that include ISV applications such as Oracle software. In addition, a service can include less traditional entities such as the following:

- A physical network device
- A configured IP address
- Kernel configuration information
- Milestones that correspond to system init state, such as the multiuser run level

Generically, a service is an entity that provides a list of capabilities to applications and other services, local and remote. A service is dependent on an implicitly declared list of local services.

A *milestone* is a special type of service. Milestone services represent high-level attributes of the system. For example, the services which constitute run levels S, 2, and 3 are each represented by milestone services.

Service Identifiers

Each service instance is named with a Fault Management Resource Identifier or FMRI. The FMRI includes the service name and the instance name. For example, the FMRI for the `rlogin` service is `svc:/network/login:rlogin`, where `network/login` identifies the service and `rlogin` identifies the service instance.

Equivalent formats for an FMRI are as follows:

- `svc://localhost/system/system-log:default`
- `svc:/system/system-log:default`
- `system/system-log:default`

In addition, some SMF commands can use the following FMRI format: `svc:/system/system-log`. Some commands infer what instance to use, when there is no ambiguity. See the SMF command man pages, such as `svcadm`(1M) or `svcs`(1), for instructions about which FMRI formats are appropriate.

The service names usually include a general functional category. The categories include the following:

- `application`
- `device`
- `milestone`
- `network`
- `platform`
- `site`
- `system`

Legacy `init.d` scripts are also represented with FMRIs that start with `lrc` instead of `svc`, for example: `lrc:/etc/rcS_d/S35cacheos_sh`. The legacy services can be monitored using SMF. However, you cannot administer these services.

When booting a system for the first time with SMF, services listed in `/etc/inetd.conf` are automatically converted into SMF services. The FMRIs for these services are slightly different. The syntax for a converted `inetd` services is:

`network/`*<service-name>*`/`*<protocol>*

In addition, the syntax for a converted service that uses the RPC protocol is:

`network/rpc-`*<service-name>*`/rpc_`*<protocol>*

Where <*service-name*> is the name defined in `/etc/inetd.conf` and <*protocol*> is the protocol for the service. For instance, the FMRI for the `rpc.cmsd` service is `network/rpc-100068_2-5/rpc_udp`.

Service States

The `svcs` command displays the state, start time, and FMRI of service instances. The state of each service is one of the following:

- `degraded` – The service instance is enabled, but is running at a limited capacity.
- `disabled` – The service instance is not enabled and is not running.
- `legacy_run` – The legacy service is not managed by SMF, but the service can be observed. This state is only used by legacy services.
- `maintenance` – The service instance has encountered an error that must be resolved by the administrator.
- `offline` – The service instance is enabled, but the service is not yet running or available to run.
- `online` – The service instance is enabled and has successfully started.
- `uninitialized` – This state is the initial state for all services before their configuration has been read.

SMF Manifests

An SMF *manifest* is an XML file that that describes a service and a set of instances. Manifests are imported to load the properties of that service and its instances into the repository.

The preferred location for manifests is `/lib/svc/manifest`. Manifests stored there will be imported and upgraded during the boot process before any services start. Running the import process early ensures that the repository will contain information from the latest manifests before the services are started. At other times you can import information from these manifests by running this command: `svcadm restart manifest-import`. `/var/svc/manifest` remains available for compatibility purposes, but manifests located there will not be imported or upgraded until the `svc:/system/manifest-import:default` service runs, which is significantly later in the boot process.

The site subdirectory of `/lib/svc/manifest` and `/var/svc/manifest` is reserved for site-specific use. Manifests in the site directory may be modified directly. Other manifests included in the software release should not be modified since those modifications will be lost during software upgrades. If changes need to be made to the set of properties included in the generic manifests, either create a profile or use the `svccfg` command.

See the `service_bundle(4)` man page for a complete description of the contents of the SMF manifests. If you need to change the properties of a service, see the `svccfg(1M)` or `inetadm(1M)` man pages.

SMF Profiles

An SMF *profile* is an XML file that lists a set of service instances and whether each should be enabled or disabled. Some profiles which are delivered with the release include:

- `/etc/svc/profile/generic_open.xml` – This profile enables the standard services that have been started by default in earlier releases.
- `/etc/svc/profile/generic_limited_net.xml` – This profile disables many of the internet services that have be started by default in earlier releases. The `network/ssh` service is enabled to provide network connectivity.
- `/etc/svc/profile/ns_*.xml` – These profiles enable services associated with the name service that is configured to run on the system.
- `/etc/svc/profile/platform_*.xml` – These profiles enable services associated with particular hardware platforms.

During the first boot after a new installation or an upgrade, some profiles are automatically applied. To be specific, the `/etc/svc/profile/generic.xml` profile is applied. This file is usually symbolically linked to `generic_open.xml` or `generic_limited_net.xml`. Also, if a profile called `site.xml` is in `/etc/svc/profile` during the first boot or is added between boots, the contents of this profile are applied. By using the `site.xml` profile, the initial set of enabled services may be customized by the administrator.

Like manifests, profiles in `/etc/svc/profile` will be applied during the early manifest import. Profiles in `/var/svc/profile` will be applied during the later manifest import.

For more information about using profiles, see "How to Apply an SMF Profile" on page 170.

Service Configuration Repository

The *service configuration repository* stores persistent configuration information as well as SMF runtime data for services. The repository is distributed among local memory and local files. SMF is designed so that eventually, service data can be represented in the network directory service. The network directory service is not yet available. The data in the service configuration repository allows for the sharing of configuration information and administrative simplicity across many instances. The service configuration repository can only be manipulated or queried using SMF interfaces. For more information about manipulating and accessing the repository, see the `svccfg(1M)` and `svcprop(1)` man pages. The service configuration repository daemon is covered in the `svc.configd(1M)` man page. The service configuration library is documented in the `libscf(3LIB)` man page.

SMF Repository Backups

SMF automatically takes the following backups of the repository:

- The boot backup is taken immediately before the first change to the repository is made during each system startup.
- The `manifest_import` backups occur after `svc:/system/early-manifest-import:default` or `svc:/system/manifest-import:default` completes, if the service imported any new manifests or ran any upgrade scripts.

Four backups of each type are maintained by the system. The system deletes the oldest backup, when necessary. The backups are stored as `/etc/svc/repository-`*type*-*YYYYMMDD_HHMMSWS*, where *YYYYMMDD* (year, month, day) and *HHMMSS* (hour, minute, second), are the date and time when the backup was taken. Note that the hour format is based on a 24–hour clock.

You can restore the repository from these backups, if an error occurs. To do so, use the `/lib/svc/bin/restore_repository` command. For more information, see "How to Repair a Corrupt Repository" on page 179.

SMF Snapshots

The data in the service configuration repository includes *snapshots*, as well as a configuration that can be edited. Data about each service instance is stored in the snapshots. The standard snapshots are as follows:

- `initial` – Taken on the first import of the manifest
- `running` – Used when the service methods are executed
- `start` – Taken at the last successful start

The SMF service always executes with the `running` snapshot. This snapshot is automatically created if it does not exist.

The `svcadm refresh` command, sometimes followed by the `svcadm restart` command, incorporates current property values into the running snapshot. The `svccfg` command is used to view or revert to instance configurations in a previous snapshot. See "How to Revert to Another SMF Snapshot" on page 168 for more information.

SMF Administrative and Programming Interfaces

This section introduces the interfaces that are available when you use SMF.

SMF Command-Line Administrative Utilities

SMF provides a set of command-line utilities that interact with SMF and accomplish standard administrative tasks. The following utilities can be used to administer SMF.

TABLE 11–1 Service Management Facility Utilities

Command Name	Function
`inetadm`	Provides the ability to observe or configure services controlled by `inetd`
`svcadm`	Provides the ability to perform common service management tasks, such as enabling, disabling, or restarting service instances
`svccfg`	Provides the ability to display and manipulate the contents of the service configuration repository
`svcprop`	Retrieves property values from the service configuration repository with a output format appropriate for use in shell scripts
`svcs`	Gives detailed views of the service state of all service instances in the service configuration repository

Service Management Configuration Library Interfaces

SMF provides a set of programming interfaces that are used to interact with the service configuration repository through the `svc.configd` daemon. This daemon is the arbiter of all requests to the local repository datastores. A set of fundamental interfaces is defined as the lowest level of interaction possible with services in the service configuration repository. The interfaces provide access to all service configuration repository features such as transactions and snapshots.

Many developers only need a set of common tasks to interact with SMF. These tasks are implemented as convenience functions on top of the fundamental services to ease the implementation burden.

SMF Components

SMF includes a master restarter daemon and delegated restarters.

SMF Master Restarter Daemon

The `svc.startd` daemon is the master process starter and restarter. The daemon is responsible for managing service dependencies for the entire system. The daemon takes on the previous responsibility that `init` held of starting the appropriate `/etc/rc*.d` scripts at the appropriate run levels. First, `svc.startd` retrieves the information in the service configuration repository. Next, the daemon starts services when their dependencies are met. The daemon is also responsible for restarting services that have failed and for shutting down services whose dependencies are no longer satisfied. The daemon keeps track of service state through an operating system view of availability through events such as process death.

SMF Delegated Restarters

Some services have a set of common behaviors on startup. To provide commonality among these services, a delegated restarter might take responsibility for these services. In addition, a delegated restarter can be used to provide more complex or application-specific restarting behavior. The delegated restarter can support a different set of methods, but exports the same service states as the master restarter. The restarter's name is stored with the service. A current example of a delegated restarter is `inetd`, which can start Internet services on demand, rather than having the services always running.

SMF and Booting

SMF provides new methods for booting a system. For instance:

- There is a additional system state which is associated with the `all` milestone. With the `all` milestone, all of the services with a defined dependency on the `multi-user-server` milestone are started, as well as any services that do not have a defined dependency. If you have added services, such as third party products, they may not be started automatically unless you use the following command:

  ```
  ok boot -m milestone=all
  ```
- When booting a system, you can choose to use the verbose option to see more messages. By default, the system will not display these messages. To boot in the verbose mode, use the following command:

  ```
  ok boot -mverbose
  ```

- There is a new system state which is associated with the none milestone. Only `init`, `svc.startd` and `svc.configd` are started if you boot a system using this milestone. This state can be very useful for debugging booting problems. In particular, debugging any problems with the configuration of SMF services is made simpler, because none of the services are started. See "How to Boot Without Starting Any Services" on page 182 for instructions on how to use the none milestone.

SMF Compatibility

While many standard services are now managed by SMF, the scripts placed in `/etc/rc*.d` continue to be executed on run-level transitions. Most of the `/etc/rc*.d` scripts that were included in previous releases have been removed as part of SMF. The ability to continue to run the remaining scripts allows for third-party applications to be added without having to convert the services to use SMF.

In addition, `/etc/inittab` and `/etc/inetd.conf` must be available for packages to amend. These are called legacy-run services. The `inetconv` command is run to add these legacy-run services to the service configuration repository. The status of these services can be viewed, but no other changes are supported through SMF. Applications that use this feature will not benefit from the precise fault containment provided by SMF.

Applications converted to utilize SMF should no longer make modifications to the `/etc/inittab` and `/etc/inetd.conf` files. The converted applications will not use the `/etc/rc*.d` scripts. Also, the new version of `inetd` does not look for entries in `/etc/inetd.conf`.

Run Levels

A system's *run level* (also known as an *init state*) defines what services and resources are available to users. A system can be in only one run level at a time.

The release has eight run levels, which are described in the following table. The default run level is specified in the `/etc/inittab` file as run level 3.

TABLE 11–2 Oracle Solaris Run Levels

Run Level	Init State	Type	Purpose
0	Power-down state	Power-down	To shut down the operating system so that it is safe to turn off power to the system.
s or S	Single-user state	Single-user	To run as a single user with some file systems mounted and accessible.
1	Administrative state	Single-user	To access all available file systems. User logins are disabled.

TABLE 11–2 Oracle Solaris Run Levels *(Continued)*

Run Level	Init State	Type	Purpose
2	Multiuser state	Multiuser	For normal operations. Multiple users can access the system and all file system. All daemons are running except for the NFS server daemons.
3	Multiuser level with NFS resources shared	Multiuser	For normal operations with NFS resources shared. This is the default run level.
4	Alternative multiuser state		Not configured by default, but available for customer use.
5	Power-down state	Power-down	To shut down the operating system so that it is safe to turn off power to the system. If possible, automatically turns off power on systems that support this feature.
6	Reboot state	Reboot	To shut down the system to run level 0, and then reboot to multiuser level with NFS resources shared (or whatever level is the default in the `inittab` file).

In addition, the `svcadm` command can be used to change the run level of a system, by selecting a milestone at which to run. The following table shows which run level corresponds to each milestone.

TABLE 11–3 Run Levels and SMF Milestones

Run Level	SMF Milestone FMRI
S	`milestone/single-user:default`
2	`milestone/multi-user:default`
3	`milestone/multi-user-server:default`

When to Use Run Levels or Milestones

Under most circumstances, using the `init` command with a run level to change the system state is sufficient. Using milestones to change system state can be confusing and can lead to unexpected behavior. In addition, the `init` command allows for the system to be shutdown, so `init` is the best command for changing system state.

However, booting a system using the none milestone, can be very useful when debugging startup problems. There is no equivalent run level to the none milestone. See "How to Boot Without Starting Any Services" on page 182 for specific instructions.

Determining a System's Run Level

Display run level information by using the who -r command.

```
$ who -r
```

Use the who -r command to determine a system's current run level for any level.

EXAMPLE 11–1 Determining a System's Run Level

This example displays information about a system's current run level and previous run levels.

```
$ who -r
 .     run-level 3  Dec 13 10:10  3  0 S
$
```

Output of who -r command	Description
run-level 3	Identifies the current run level
Dec 13 10:10	Identifies the date of last run level change
3	Also identifies the current run level
0	Identifies the number of times the system has been at this run level since the last reboot
S	Identifies the previous run level

/etc/inittab File

When you boot the system or change run levels with the init or shutdown command, the init daemon starts processes by reading information from the /etc/inittab file. This file defines these important items for the init process:

- That the init process will restart
- What processes to start, monitor, and restart if they terminate
- What actions to take when the system enters a new run level

Each entry in the /etc/inittab file has the following fields:

id:*rstate*:*action*:*process*

The following table describes the fields in an inittab entry.

TABLE 11–4 Fields Descriptions for the inittab File

Field	Description
id	Is a unique identifier for the entry.
rstate	Lists the run levels to which this entry applies.
action	Identifies how the process that is specified in the process field is to be run. Possible values include: sysinit, boot, bootwait, wait, and respawn. For a description of the other action keywords, see inittab(4).
process	Defines the command or script to execute.

EXAMPLE 11–2 Default inittab File

The following example shows a default inittab file that is installed with the release. A description for each line of output in this example follows.

```
ap::sysinit:/sbin/autopush -f /etc/iu.ap      (1)
sp::sysinit:/sbin/soconfig -f /etc/sock2path                 (2)
smf::sysinit:/lib/svc/bin/svc.startd    >/dev/msglog 2<>/dev/msglog         (3)
p3:s1234:powerfail:/usr/sbin/shutdown -y -i5 -g0 >/dev/msglog 2<>/dev/...    (4)
```

1. Initializes STREAMS modules
2. Configures socket transport providers
3. Initializes the master restarter for SMF
4. Describes a power fail shutdown

What Happens When the System Is Brought to Run Level 3

1. The init process is started and reads the /etc/default/init file to set any environment variables. By default, only the TIMEZONE variable is set.
2.

   Then, init reads the inittab file and does the following:

   a. Executes any process entries that have sysinit in the action field so that any special initializations can take place before users login.

   b. Passes the startup activities to svc.startd.

   For a detailed description of how the init process uses the inittab file, see init(1M).

◆ ◆ ◆ CHAPTER 12

Managing Services (Tasks)

This chapter covers the tasks required to manage and monitor the Service Management Facility (SMF). In addition, information that is related to managing run level scripts is provided. The following topics are covered:

- "Managing Services (Task Map)" on page 161
- "Monitoring SMF Services" on page 162
- "Managing SMF Services" on page 165
- "Configuring SMF Services" on page 171
- "Using Run Control Scripts" on page 176
- "Troubleshooting the Service Management Facility" on page 179

Managing Services (Task Map)

The following task map describes the procedures that are needed to use SMF.

Task	Description	For Instructions
Display the status of a service instance.	Displays the status of all running service instances.	"How to List the Status of a Service" on page 162
Display the service dependents.	Display the services that are dependent on the specified service.	"How to Show Which Services Are Dependent on a Service Instance" on page 164
Display the dependencies of a service.	Display the services that a specified service is dependent on. This information can be used to help identify what is preventing a service from starting.	"How to Show Which Services a Service Is Dependent On" on page 164
Disable a service instance.	Turns off a service that is not functioning properly or needs to be off to increase security.	"How to Disable a Service Instance" on page 166

Task	Description	For Instructions
Enable a service instance	Starts a service.	"How to Enable a Service Instance" on page 166
Restart a service instance.	Restart a service, without having to use separate commands to disable and then enable the service.	"How to Restart a Service" on page 167
Modify a service instance.	Modifies the configuration parameters of a specified service instance.	"How to Modify a Service" on page 171
	Changes a configuration property of a service controlled by `inetd`.	"How to Change a Property for an `inetd` Controlled Service" on page 173
	Changes the startup options of a service controlled by `inetd`.	"How to Modify a Command-Line Argument for an `inetd` Controlled Service" on page 174
Convert `inetd.conf` entries.	Converts `inetd` services into legacy-run services that can be monitored using SMF.	"How to Convert `inetd.conf` Entries" on page 175
Repair a corrupt service configuration repository.	Replaces a corrupt repository with a default version.	"How to Repair a Corrupt Repository" on page 179
Boot a system without starting any services.	Boots a system without starting any services so that configuration problems that prevent booting can be fixed.	"How to Boot Without Starting Any Services" on page 182

Monitoring SMF Services

The following tasks show how to monitor SMF services.

▼ How to List the Status of a Service

This procedure can be used to show what services are running.

● **Run the `svcs` command.**

Running this command without any options displays a status report of the service specified by the FMRI.

```
% svcs -l FMRI
```

Example 12–1 Showing the Status of the `rlogin` Service

This example shows the status of a service that includes many contracts.

```
% svcs -l network/login:rlogin
fmri         svc:/network/login:rlogin
enabled      true
state        online
next_state   none
restarter    svc:/network/inetd:/default
contract_id  42325 41441 40776 40348 40282 40197 39025 38381 38053\
 33697 28625 24652 23689 15352 9889 7194 6576 6360 5387 1475 3015\
 6545 6612 9302 9662 10484 16254 19850 22512 23394 25876 26113 27326\
 34284 37939 38405 38972 39200 40503 40579 41129 41194
```

Example 12–2 Showing the Status of the `sendmail` Service

This example shows the status of a service that includes dependencies.

```
% svcs -l network/smtp:sendmail
fmri         svc:/network/smtp:sendmail
enabled      true
state        online
next_state   none
restarter    svc:/system/svc/restarter:default
contract_id  29462
dependency   require_all/refresh file://localhost/etc/nsswitch.conf (-)
dependency   require_all/refresh file://localhost/etc/mail/sendmail.cf (-)
dependency   optional_all/none svc:/system/system-log (online)
dependency   require_all/refresh svc:/system/identity:domain (online)
dependency   require_all/refresh svc:/milestone/name-services (online)
dependency   require_all/none svc:/network/service (online)
dependency   require_all/none svc:/system/filesystem/local (online)
```

Example 12–3 Showing the Status of all Services

The following command lists all services that are installed on the system as well as the status of each service. The command displays those services that are disabled as well as those that are enabled.

```
% svcs -a
```

Example 12–4 Showing the Status of Services Controlled by `inetd`

The following command lists services that are controlled by `inetd`. Each service's FMRI is listed, along with the run state and whether the service is enabled or disabled.

```
% inetadm
```

▼ How to Show Which Services Are Dependent on a Service Instance

This procedure shows how to determine which service instances depend on the specified service.

● **Display the service dependents.**

```
% svcs -D FMRI
```

Example 12–5 Displaying the Service Instances That Are Dependent on the Multiuser Milestone

The following example shows how to determine which service instances are dependent on the multiuser milestone.

```
% svcs -D milestone/multi-user
STATE          STIME    FMRI
online         Apr_08   svc:/milestone/multi-user-server:default
```

▼ How to Show Which Services a Service Is Dependent On

This procedure shows how to determine which services a specified service instance is dependent on.

● **Display the service dependencies.**

```
% svcs -d FMRI
```

Example 12–6 Displaying the Service Instances That the Multiuser Milestone Is Dependent On

The following example shows the services instances that the multiuser milestone is dependent on.

```
% svcs -d milestone/multi-user:default
STATE          STIME    FMRI
disabled       Aug_24   svc:/platform/sun4u/sf880drd:default
online         Aug_24   svc:/milestone/single-user:default
online         Aug_24   svc:/system/utmp:default
online         Aug_24   svc:/system/system-log:default
online         Aug_24   svc:/system/system-log:default
online         Aug_24   svc:/system/rmtmpfiles:default
online         Aug_24   svc:/network/rpc/bind:default
online         Aug_24   svc:/milestone/name-services:default
online         Aug_24   svc:/system/filesystem/local:default
online         Aug_24   svc:/system/mdmonitor:default
```

Managing SMF Services (Task Map)

Task	Description	For Instructions
Disable a service instance.	Stops a running service and prevents the service from restarting.	"How to Disable a Service Instance" on page 166
Enable a service instance.	Starts a service. In addition, the service will be restarted during subsequent reboots.	"How to Enable a Service Instance" on page 166
Restarting a service.	Stops and starts a service with one command.	"How to Restart a Service" on page 167
Restoring a service in maintenance state.	Shows how to clean up and restart a service that is in maintenance state.	"How to Restore a Service That Is in the Maintenance State" on page 167
Revert to a snapshot.	Uses a previous snapshot to correct problems with a service.	"How to Revert to Another SMF Snapshot" on page 168
Create an profile.	Create a profile to disable or enable services as needed.	"How to Create an SMF Profile" on page 169
Apply a profile.	Uses the information in a profile to disable or enable services as needed.	"How to Apply an SMF Profile" on page 170
Change the services and their configuration using the `netservices` command.	Uses the information in the `generic_limited.xml` or `generic_open.xml` profiles to disable or enable services and make configuration changes to those services, as well.	"Changing Services Offered to the Network with `generic*.xml`" on page 171

Managing SMF Services

This section includes information on managing SMF services.

Using RBAC Rights Profiles With SMF

You can use RBAC rights profiles to allow users to manage some of the SMF services, without having to give the user `root` access. The rights profiles define what commands the user can run. For SMF, the following profiles have been created:

- `Service Management`: User can add, delete or modify services.
- `Service Operator`: User can request state changes of any service instance, such as restart and refresh.

For specific information about the authorizations, see the `smf_security(5)` man page. For instructions to assign a rights profile, see "How to Change the RBAC Properties of a User" in *System Administration Guide: Security Services*.

▼ How to Disable a Service Instance

Use the following procedure to disable a service. The service status change is recorded in the service configuration repository. Once the service is disabled, the disabled state will persist across reboots. The only way to get the service running again is to enable it.

1. **Become an administrator or assume a role that includes the `Service Management` rights profile.**

   For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2. **Check the dependents of the service you want to disable.**

   If this service has dependents that you need, then you cannot disable this service.

   ```
   # svcs -D FMRI
   ```

3. **Disable the service.**

   ```
   # svcadm disable FMRI
   ```

Example 12–7 Disabling the `rlogin` Service

The output from the first command shows that the `rlogin` service has no dependents. The second command in this example disables the `rlogin` service. The third command shows that the state of the `rlogin` service instance is disabled.

```
# svcs -D network/login:rlogin
# svcadm disable network/login:rlogin
STATE          STIME    FMRI
# svcs network/login:rlogin
STATE          STIME    FMRI
disabled         11:17:24 svc:/network/login:rlogin
```

▼ How to Enable a Service Instance

Use the following procedure to enable a service. The service status change is recorded in the service configuration repository. Once the service is enabled, the enabled state will persist across system reboots if the service dependencies are met.

1. **Become an administrator or assume a role that includes the `Service Management` rights profile.**

   For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2. **Determine whether service dependencies are satisfied.**

   If the service is enabled, then the service dependencies are satisfied. If not, use `svcadm enable -r` FMRI to recursively enable all dependencies.

   ```
   # svcs -l FMRI|grep enabled
   ```

3 **Enable a service.**

```
# svcadm enable FMRI
```

Example 12–8 Enabling the `rlogin` Service

The second command in this example enables the `rlogin` service. The third command shows that the state of the `rlogin` service instance is online.

```
# svcs -l network/login:rlogin|grep enabled
enabled      false
# svcadm enable network/login:rlogin
# svcs network/login:rlogin
STATE          STIME    FMRI
online         12:09:16 svc:/network/login:rlogin
```

Example 12–9 Enabling a Service in Single-user Mode

The following command enables `rpcbind`. The `-t` option starts the service in temporary mode which does not change the service repository. The repository is not writable in single-user mode. The `-r` option recursively starts all the dependencies of the named service.

```
# svcadm enable -rt rpc/bind
```

▼ How to Restart a Service

If a service is currently running but needs to be restarted due to a configuration change or some other reason, the service can be restarted without you having to type separate commands to stop and start the service. The only reason to specifically disable and then enable a service is if changes need to be made before the service is enabled, and after the service is disabled.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Restart a service.**

```
# svcadm restart FMRI
```

▼ How to Restore a Service That Is in the Maintenance State

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2. **Determine if any process that are dependent to the service have not stopped.**

   Normally, when a service instance is in a maintenance state, all processes associated with that instance have stopped. However, you should make sure before you proceed. The following command lists all of the processes that are associated with a service instance as well as the PIDs for those processes.

   ```
   # svcs -p FMRI
   ```

3. **(Optional) Kill any remaining processes.**

   Repeat this step for all processes that are displayed by the `svcs` command.

   ```
   # pkill -9 PID
   ```

4. **If necessary, repair the service configuration.**

   Consult the appropriate service log files in `/var/svc/log` for a list of errors.

5. **Restore the service.**

   ```
   # svcadm clear FMRI
   ```

▼ How to Revert to Another SMF Snapshot

If the service configuration is wrong, the problem can be fixed by reverting to the last snapshot that started successfully. In this procedure, a previous snapshot of the `console-login` service is used.

1. **Become an administrator or assume a role that includes the `Service Management` rights profile.**

   For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*.

2. **Run the `svccfg` command.**

   ```
   # svccfg
   svc:>
   ```

   a. **Select the service instance that you want to fix.**

   Note – You must use an FMRI that fully defines the instance. No shortcuts are allowed.

   ```
   svc:> select system/console-login:default
   svc:/system/console-login:default>
   ```

   b. **Generate a list of available snapshots.**

   ```
   svc:/system/console-login:default> listsnap
   initial
   running
   ```

```
start
svc:/system/console-login:default>
```

c. **Select to revert to the `start` snapshot.**

The `start` snapshot is the last snapshot in which the service successfully started.

```
svc:/system/console-login:default> revert start
svc:/system/console-login:default>
```

d. **Quit `svccfg`.**

```
svc:/system/console-login:default> quit
#
```

3 **Update the information in the service configuration repository.**

This step updates the repository with the configuration information from the `start` snapshot.

```
# svcadm refresh system/console-login
```

4 **Restart the service instance.**

```
# svcadm restart system/console-login
```

▼ How to Create an SMF Profile

A profile is an XML file which lists SMF services and whether each should be enabled or disabled. Profiles are used to enable or disable many services at once. Not all services need to be listed in a profile. Each profile only needs to include those services that need to be enabled or disabled to make the profile useful.

1 **Create a profile.**

In this example, the `svccfg` command is used to create a profile which reflects which services are enabled or disabled on the current system. Alternately, you could make a copy of an existing profile to edit.

```
# svccfg extract> profile.xml
```

If you are using JumpStart, if you have large numbers of identical systems, or if you want to archive the system configuration for later restoration, you may want to use this procedure to create a unique version of a SMF profile.

2 **Edit the `profile.xml` file to make any required changes.**

a. **Change the name of the profile in the `service_bundle` declaration.**

In this example the name is changed to `profile`.

```
# cat profile.xml
  ...
<service_bundle type=‘profile‘ name=‘profile‘
    xmIns::xi=’http://www.w3.org/2003/XInclude’
  ...
```

b. **Remove any services that should not be managed by this profile.**

For each service, remove the three lines that describe the service. Each service description starts with `<service` and ends with `</service`. This example shows the lines for the LDAP client service.

```
# cat profile.xml
 ...
 <service name='network/ldap/client' version='1' type='service'>
        <instance  name='default' enabled='true'/>
 </service>
```

c. **Add any services that should be managed by this profile.**

Each service needs to be defined using the three line syntax shown above.

d. **If necessary, change the enabled flag for selected services.**

In this example, the `sendmail` service is disabled.

```
# cat profile.xml
  ...
  <service  name='network/smtp' version='1' type='service'>
    <instance  name='sendmail' enabled='false'/>
  </service>
  ...
```

3 **When necessary, apply the new profile.**

See "How to Apply an SMF Profile" on page 170 for instructions.

▼ How to Apply an SMF Profile

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Apply an profile.**

In this example, the `profile.xml` profile is used.

```
# svccfg apply profile.xml
```

Note – For specific instructions for switching between the `generic_limited_net.xml` and `generic_open.xml` and the properties that need to be applied when making this switch, please see "Changing Services Offered to the Network with `generic*.xml`" on page 171

▼ Changing Services Offered to the Network with `generic*.xml`

The `netservices` command switches system services between minimal network exposure and the traditional network exposure (as in previous releases). The switch is done with the `generic_limited.xml` and `generic_open.xml` profiles. In addition, some services properties are changed by the command to limit some services to a local-only mode or to the traditional mode, as appropriate.

Note – The `generic_limited_net` profile and the local-mode only service properties are applied by default.

---

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Run the `netservices` command.**

In this example, the open or traditional network exposure is selected.

```
# /usr/sbin/netservices open
```

Example 12–10 Limiting Network Service Exposure

This command changes properties to run some services in local mode, as well as restricts which services are enabled with the `generic_limited_net` profile. The command should only be used if the `generic_open.xml` profile had been applied.

```
# /usr/sbin/netservices limited
```

Configuring SMF Services

▼ How to Modify a Service

The following procedure shows how to change the configuration of a service that is not managed by the `inetd` service.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Make changes to the configuration files, as needed.**

Many of the services have one or more configuration files that are used to define the startup or other configuration information. These files can be changed while the service is running. The contents of the files is only checked when the service is started.

3 **Restart the service.**

```
# svcadm restart FMRI
```

Example 12–11 Sharing an NFS File System

To share a file system using the NFS service, you must define the file system in the `/etc/dfs/dfstab` file and then restart the NFS service. This example shows you what the `dfstab` file could look like, as well as how to restart the service.

```
# cat /etc/dfs/dfstab
  .
  .
share -F nfs -o rw /export/home
# svcadm restart svc:/network/nfs/server
```

▼ How to Change an Environment Variable for a Service

This procedure shows how to modify `cron` environment variables to help with debugging.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Verify that the service is running.**

```
# svcs system/cron
STATE          STIME    FMRI
online         Dec_04   svc:/system/cron:default
```

3 **Set environment variables.**

In this example the `UMEM_DEBUG` and `LD_PRELOAD` environment variables are set. For information about the `setenv` subcommand refer to the `svccfg`(1M) man page.

```
# svccfg -s system/cron:default setenv UMEM_DEBUG default
# svccfg -s system/cron:default setenv LD_PRELOAD libumem.so
```

4 **Refresh and restart the service.**

```
# svcadm refresh system/cron
# svcadm restart system/cron
```

5 **Verify that the change has been made.**

```
# pargs -e ‘pgrep -f /usr/sbin/cron‘
100657: /usr/sbin/cron
envp[0]: LOGNAME=root
envp[1]: LD_PRELOAD=libumem.so
envp[2]: PATH=/usr/sbin:/usr/bin
envp[3]: SMF_FMRI=svc:/system/cron:default
envp[4]: SMF_METHOD=/lib/svc/method/svc-cron
envp[5]: SMF_RESTARTER=svc:/system/svc/restarter:default
envp[6]: TZ=GB
envp[7]: UMEM_DEBUG=default
#
```

▼ How to Change a Property for an `inetd` Controlled Service

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see “How to Obtain Administrative Rights” in *System Administration Guide: Security Services*

2 **List the properties for the specific service.**

This command displays all of the properties for the service identified by the FMRI.

```
# inetadm -l FMRI
```

3 **Change the property for the service.**

Each property for an `inetd` controlled service is defined by a property name and an assigned value. Supplying the property name without a specified value resets the property to the default value. Specific information about the properties for a service should be covered in the man page associated with the service.

```
# inetadm -m FMRI property-name=value
```

4 **Verify that the property has changed.**

List the properties again to make sure that the appropriate change has occurred.

```
# inetadm -l FMRI
```

5 **Confirm that the change has taken effect.**

Confirm the property change that the change has the desired effect.

Example 12–12 Changing the `tcp_trace` Property for `telnet`

The following example shows how to set the `tcp_trace` property for `telnet` to `true`. Checking the `syslog` output after running a `telnet` command shows that the change has taken effect.

```
# inetadm -l svc:/network/telnet:default
SCOPE    NAME=VALUE
         name="telnet"
 .
 .
default  inherit_env=TRUE
default  tcp_trace=FALSE
default  tcp_wrappers=FALSE
# inetadm -m svc:/network/telnet:default tcp_trace=TRUE
# inetadm -l svc:/network/telnet:default
SCOPE    NAME=VALUE
         name="telnet"
 .
 .
default  inherit_env=TRUE
         tcp_trace=TRUE
default  tcp_wrappers=FALSE
# telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
login: root
Password:
Last login: Mon Jun 21 05:55:45 on console
Sun Microsystems Inc.   SunOS 5.10      s10_57  May 2004
# ^D
Connection to localhost closed by foreign host.
# tail -1 /var/adm/messages
Jun 21 06:04:57 yellow-19 inetd[100308]: [ID 317013 daemon.notice] telnet[100625]
    from 127.0.0.1 32802
```

▼ How to Modify a Command-Line Argument for an `inetd` Controlled Service

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **List the `exec` property for the specific service.**

This command displays all the properties for the service identified by the FMRI. Adding the `grep` command restricts the output to the `exec` property for the service.

```
# inetadm -l FMRI|grep exec
```

3 **Change the `exec` property for the service.**

The *command-syntax* set with the `exec` property defines the command string that is run when the service is started.

```
# inetadm -m FMRI exec="command-syntax
"
```

4 **Verify that the property has changed.**

List the properties again to make sure that the appropriate change has occurred.

```
# inetadm -l FMRI
```

Example 12–13 Adding the Connection Logging (`-l`) Option to the `ftp` Command

In this example, the `-l` option is added to the `ftp` daemon when it is started. The effect of this change can be seen by reviewing the `syslog` output after a `ftp` login session has been completed.

```
# inetadm -l svc:/network/ftp:default | grep exec
        exec="/usr/sbin/in.ftpd -a"
# inetadm -m svc:/network/ftp:default exec="/usr/sbin/in.ftpd -a -l"
# inetadm -l svc:/network/ftp:default
SCOPE    NAME=VALUE
         name="ftp"
         endpoint_type="stream"
         proto="tcp6"
         isrpc=FALSE
         wait=FALSE
         exec="/usr/sbin/in.ftpd -a -l"
  .
  .
# ftp localhost
Connected to localhost.
220 yellow-19 FTP server ready.
Name (localhost:root): mylogin
331 Password required for mylogin.
Password:
230 User mylogin logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 236 bytes in 0 transfers.
221-Thank you for using the FTP service on yellow-19.
221 Goodbye.
# tail -2 /var/adm/messages
Jun 21 06:54:33 yellow-19 ftpd[100773]: [ID 124999 daemon.info] FTP LOGIN FROM localhost
     [127.0.0.1], mylogin
Jun 21 06:54:38 yellow-19 ftpd[100773]: [ID 528697 daemon.info] FTP session closed
```

▼ How to Convert `inetd.conf` Entries

The following procedure converts `inetd.conf` entries into SMF service manifests. This procedure needs to be run any time a third-party application that depends on `inetd` is added to a system. Also run this procedure, if you need to make configuration changes to the entry in `/etc/inetd.conf`.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Convert the `inetd.conf` entries.**

The `inetconv` command converts each entry in the selected file into service manifests.

```
# inetconv -i filename
```

Example 12–14 Converting `/etc/inet/inetd.conf` Entries into SMF Service Manifests

```
# inetconv -i /etc/inet/inetd.conf
```

Using Run Control Scripts (Task Map)

Task	Description	For Instructions
Stop or start a service.	Use a run control script to stop or start a service.	"How to Use a Run Control Script to Stop or Start a Legacy Service" on page 176
Add a run control script.	Create a run control script and add it to the `/etc/init.d` directory.	"How to Add a Run Control Script" on page 177
Disable a run control script.	Disable a run control script by renaming the file.	"How to Disable a Run Control Script" on page 178

Using Run Control Scripts

▼ How to Use a Run Control Script to Stop or Start a Legacy Service

One advantage of having individual scripts for each run level is that you can run scripts in the `/etc/init.d` directory individually to stop system services without changing a system's run level.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Stop the system service.**

```
# /etc/init.d/filename
stop
```

3 **Restart the system service.**

```
# /etc/init.d/filename
start
```

4 **Verify that the service has been stopped or started.**

```
# pgrep -f service
```

Example 12–15 Using a Run Control Script to Stop or Start a Service

For example, you can stop the NFS server daemons by typing the following:

```
# /etc/init.d/nfs.server stop
# pgrep -f nfs
```

Then, you can restart the NFS server daemons by typing the following:

```
# /etc/init.d/nfs.server start
# pgrep -f nfs
101773
101750
102053
101748
101793
102114
# pgrep -f nfs -d, | xargs ps -fp
     UID    PID   PPID   C    STIME TTY         TIME CMD
  daemon 101748      1   0   Sep 01 ?           0:06 /usr/lib/nfs/nfsmapid
  daemon 101750      1   0   Sep 01 ?          26:27 /usr/lib/nfs/lockd
  daemon 101773      1   0   Sep 01 ?           5:27 /usr/lib/nfs/statd
    root 101793      1   0   Sep 01 ?          19:42 /usr/lib/nfs/mountd
  daemon 102053      1   0   Sep 01 ?        2270:37 /usr/lib/nfs/nfsd
  daemon 102114      1   0   Sep 01 ?           0:35 /usr/lib/nfs/nfs4cbd
```

▼ How to Add a Run Control Script

If you want to add a run control script to start and stop a service, copy the script into the `/etc/init.d` directory. Then, create links in the `rc`*n* `.d` directory where you want the service to start and stop.

See the `README` file in each `/etc/rc` *n*`.d` directory for more information on naming run control scripts. The following procedure describes how to add a run control script.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Add the script to the `/etc/init.d` directory.**

```
# cp filename/etc/init.d
# chmod 0744 /etc/init.d/filename
# chown root:sys /etc/init.d/filename
```

3 **Create links to the appropriate `rc n.d` directory.**

```
# cd /etc/init.d
# ln filename /etc/rc2.d/Snnfilename
# ln filename /etc/rcn.d/Knnfilename
```

4 **Verify that the script has links in the specified directories.**

```
# ls /etc/init.d/*filename /etc/rc2.d/*filename /etc/rcn.d/*filename
```

Example 12–16 Adding a Run Control Script

The following example shows how to add a run control script for the `xyz` service.

```
# cp xyz /etc/init.d
# chmod 0744 /etc/init.d/xyz
# chown root:sys /etc/init.d/xyz
# cd /etc/init.d
# ln xyz /etc/rc2.d/S99xyz
# ln xyz /etc/rc0.d/K99xyz
# ls /etc/init.d/*xyz /etc/rc2.d/*xyz /etc/rc0.d/*xyz
```

▼ How to Disable a Run Control Script

You can disable a run control script by renaming it with an underscore (_) at the beginning of the file name. Files that begin with an underscore or dot are not executed. If you copy a file by adding a suffix to it, both files will be run.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Rename the script by adding an underscore (_) to the beginning of the new file.**

```
# cd /etc/rcn.d
# mv filename_filename
```

3 **Verify that the script has been renamed.**

```
# ls _*
_filename
```

Example 12–17 Disabling a Run Control Script

The following example shows how to rename the `S99datainit` script.

```
# cd /etc/rc2.d
# mv S99datainit _S99datainit
# ls _*
_S99datainit
```

Troubleshooting the Service Management Facility

▼ Debugging a Service That Is Not Starting

In this procedure, the print service is disabled.

1 **Become an administrator or assume a role that includes the `Service Management` rights profile.**

For more information, see "How to Obtain Administrative Rights" in *System Administration Guide: Security Services*

2 **Request information about the hung service.**

```
# svcs -xv
svc:/application/print/server:default (LP Print Service)
 State: disabled since Wed 13 Oct 2004 02:20:37 PM PDT
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: man -M /usr/share/man -s 1M lpsched
Impact: 2 services are not running:
        svc:/application/print/rfc1179:default
        svc:/application/print/ipp-listener:default
```

The `-x` option provides additional information about the service instances that are impacted.

3 **Enable the service.**

```
# svcadm enable application/print/server
```

▼ How to Repair a Corrupt Repository

This procedure shows how to replace a corrupt repository with a default copy of the repository. When the repository daemon, `svc.configd`, is started, it does an integrity check of the configuration repository. This repository is stored in `/etc/svc/repository.db`. The repository can become corrupted due to one of the following reasons:

- Disk failure
- Hardware bug
- Software bug
- Accidental overwrite of the file

If the integrity check fails, the `svc.configd` daemon writes a message to the console similar to the following:

```
svc.configd: smf(5) database integrity check of:

    /etc/svc/repository.db

  failed.  The database might be damaged or a media error might have
  prevented it from being verified.  Additional information useful to
  your service provider is in:

    /etc/svc/volatile/db_errors

  The system will not be able to boot until you have restored a working
  database.  svc.startd(1M) will provide a sulogin(1M) prompt for recovery
  purposes.  The command:

    /lib/svc/bin/restore_repository

  can be run to restore a backup version of your repository. See
  http://sun.com/msg/SMF-8000-MY for more information.
```

The `svc.startd` daemon then exits and starts `sulogin` to enable you to perform maintenance.

1 **Enter the `root` password at the `sulogin` prompt. `sulogin` enables the `root` user to enter system maintenance mode to repair the system.**

2 **Run the following command:**

```
# /lib/svc/bin/restore_repository
```

Running this command takes you through the necessary steps to restore a non-corrupt backup. SMF automatically takes backups of the repository at key system moments. For more information see "SMF Repository Backups" on page 153.

When started, the `/lib/svc/bin/restore_repository` command displays a message similar to the following:

```
Repository Restore utility
See http://sun.com/msg/SMF-8000-MY for more information on the use of
this script to restore backup copies of the smf(5) repository.

If there are any problems which need human intervention, this script
will give instructions and then exit back to your shell.

Note that upon full completion of this script, the system will be
rebooted using reboot(1M), which will interrupt any active services.
```

If the system that you are recovering is not a local zone, the script explains how to remount the `/` and `/usr` file systems with read and write permissions to recover the databases. The script exits after printing these instructions. Follow the instructions, paying special attention to any errors that might occur.

After the `root` (`/`) file system is mounted with write permissions, or if the system is a local zone, you are prompted to select the repository backup to restore:

```
The following backups of /etc/svc/repository.db exists, from
oldest to newest:
```

```
... list of backups ...
```

Backups are given names, based on type and the time the backup was taken. Backups beginning with `boot` are completed before the first change is made to the repository after system boot. Backups beginning with `manifest_import` are completed after `svc:/system/manifest-import:default` finishes its process. The time of the backup is given in *YYYYMMDD_HHMMSS* format.

3 Enter the appropriate response.

Typically, the most recent backup option is selected.

```
Please enter one of:
        1) boot, for the most recent post-boot backup
        2) manifest_import, for the most recent manifest_import backup.
        3) a specific backup repository from the above list
        4) -seed-, the initial starting repository. (All customizations
           will be lost.)
        5) -quit-, to cancel.

Enter response [boot]:
```

If you press Enter without specifying a backup to restore, the default response, enclosed in `[]` is selected. Selecting `-quit-` exits the `restore_repository` script, returning you to your shell prompt.

Note – Selecting `-seed-` restores the `seed` repository. This repository is designed for use during initial installation and upgrades. Using the `seed` repository for recovery purposes should be a last resort.

After the backup to restore has been selected, it is validated and its integrity is checked. If there are any problems, the `restore_repository` command prints error messages and prompts you for another selection. Once a valid backup is selected, the following information is printed, and you are prompted for final confirmation.

```
After confirmation, the following steps will be taken:

svc.startd(1M) and svc.configd(1M) will be quiesced, if running.
/etc/svc/repository.db
    -- renamed --> /etc/svc/repository.db_old_YYYYMMDD_HHMMSS
/etc/svc/volatile/db_errors
    -- copied --> /etc/svc/repository.db_old_YYYYMMDD_HHMMSS_errors
repository_to_restore
    -- copied --> /etc/svc/repository.db
and the system will be rebooted with reboot(1M).

Proceed [yes/no]?
```

4 Type yes to remedy the fault.

The system reboots after the `restore_repository` command executes all of the listed actions.

▼ How to Boot Without Starting Any Services

If problems with starting services occur, sometimes a system will hang during the boot. This procedure shows how to troubleshoot this problem.

1. **Boot without starting any services.**

   This command instructs the `svc.startd` daemon to temporarily disable all services and start `sulogin` on the console.

   ```
   ok boot -m milestone=none
   ```

2. **Log in to the system as `root`.**

3. **Enable all services.**

   ```
   # svcadm milestone all
   ```

4. **Determine where the boot process is hanging.**

   When the boot process hangs, determine which services are not running by running `svcs -a`. Look for error messages in the log files in `/var/svc/log`.

5. **After fixing the problems, verify that all services have started.**

   a. **Verify that all needed services are online.**

   ```
   # svcs -x
   ```

   b. **Verify that the `console-login` service dependencies are satisfied.**

   This command verifies that the `login` process on the console will run.

   ```
   # svcs -l system/console-login:default
   ```

6. **Continue the normal booting process.**

▼ How to Force a `sulogin` Prompt If the `system/filesystem/local:default` Service Fails During Boot

Local file systems that are not required to boot the system are mounted by the `svc:/system/filesystem/local:default` service. When any of those file systems are unable to be mounted, the service enters a maintenance state. System startup continues, and any services which do not depend on `filesystem/local` are started. Services which require `filesystem/local` to be online before starting through dependencies are not started.

To change the configuration of the system so that a `sulogin` prompt appears immediately after the service fails instead of allowing system startup to continue, follow the procedure below.

1 Modify the `system/console-login` service.

```
# svccfg -s svc:/system/console-login
svc:/system/console-login> addpg site,filesystem-local dependency
svc:/system/console-login> setprop site,filesystem-local/entities = fmri: svc:/system/filesystem/local

svc:/system/console-login> setprop site,filesystem-local/grouping = astring: require_all

svc:/system/console-login> setprop site,filesystem-local/restart_on = astring: none

svc:/system/console-login> setprop site,filesystem-local/type = astring: service

svc:/system/console-login> end
```

2 Refresh the service.

```
# svcadm refresh console-login
```

Example 12–18 Forcing an `sulogin` Prompt Using JumpStart

Save the following commands into a script and save it as `/etc/rcS.d/S01site-customfs`.

```
#!/bin/sh
#
# This script adds a dependency from console-login -> filesystem/local
# This forces the system to stop the boot process and drop to an sulogin prompt
# if any file system in filesystem/local fails to mount.

PATH=/usr/sbin:/usr/bin
export PATH

    svccfg -s svc:/system/console-login << EOF
addpg site,filesystem-local dependency
setprop site,filesystem-local/entities = fmri: svc:/system/filesystem/local
setprop site,filesystem-local/grouping = astring: require_all
setprop site,filesystem-local/restart_on = astring: none
setprop site,filesystem-local/type = astring: service
EOF

svcadm refresh svc:/system/console-login

[ -f /etc/rcS.d/S01site-customfs ] &&
    rm -f /etc/rcS.d/S01site-customfs
```

Troubleshooting When a failure occurs with the `system/filesystem/local:default` service, the `svcs -vx` command should be used to identify the failure. After the failure has been fixed, the following command clears the error state and allows the system boot to continue: `svcadm clear filesystem/local`.

Index