Booting and Shutting Down Oracle® Solaris on x86 Platforms

Part No: 821–2726–11
March 2012

ORACLE®

120305@25097

Contents

Preface

Booting and Shutting Down Oracle Solaris on x86 Platforms is part of a documentation set that provides a significant portion of the Oracle Solaris system administration information. This guide primarily contains information about booting x86 based systems. However, some information pertains to both the x86 and SPARC platforms.

This book assumes you have completed the following tasks:

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

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*. This document cites any implementation differences between the platform types.

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 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
Booting and Shutting Down Oracle Solaris on SPARC Platforms	Booting and shutting down a system, managing boot services, modifying boot behavior, booting from ZFS, managing the boot archive, and troubleshooting booting on SPARC platforms

Book Title	Topics
Booting and Shutting Down Oracle Solaris on x86 Platforms	Booting and shutting down a system, managing boot services, modifying boot behavior, booting from ZFS, managing the boot archive, and troubleshooting booting on x86 platforms
Oracle Solaris Administration: Common Tasks	Using Oracle Solaris commands, booting and shutting down a system, managing user accounts and groups, managing services, hardware faults, system information, system resources, and system performance, managing software, printing, the console and terminals, and troubleshooting system and software problems
Oracle Solaris Administration: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data
Oracle Solaris Administration: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, IP Filter, and IPQoS
Oracle Solaris Administration: Naming and Directory Services	DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP
Oracle Solaris Administration: Network Interfaces and Network Virtualization	Automatic and manual IP interface configuration including WiFi wireless; administration of bridges, VLANs, aggregations, LLDP, and IPMP; virtual NICs and resource management.
Oracle Solaris Administration: Network Services	Web cache servers, time-related services, network file systems (NFS and autofs), mail, SLP, and PPP
Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, 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 Zones, which host Oracle Solaris 10 environments running on the Oracle Solaris 11 kernel
Oracle Solaris Administration: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Cryptographic Framework, Key Management Framework, privileges, RBAC, SASL, Secure Shell and virus scanning.
Oracle Solaris Administration: SMB and Windows Interoperability	SMB service, which enables you to configure an Oracle Solaris system to make SMB shares available to SMB clients; 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 Administration: ZFS File Systems	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

Book Title	Topics
Trusted Extensions Configuration and Administration	System installation, configuration, and administration that is specific to Trusted Extensions
Oracle Solaris 11 Security Guidelines	Securing an Oracle Solaris system, as well as usage scenarios for its security features, such as zones, ZFS, and Trusted Extensions
Transitioning From Oracle Solaris 10 to Oracle Solaris 11	Provides system administration information and examples for transitioning from Oracle Solaris 10 to Oracle Solaris 11 in the areas of installation, device, disk, and file system management, software management, networking, system management, security, virtualization, desktop features, user account management, and user environments

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface	Description	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 `/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

Booting and Shutting Down an x86 Based System (Overview)

Oracle Solaris is designed to run continuously so that enterprise services, such as databases and web services, remain available as much as possible. This chapter provides overview information and guidelines for booting and shutting down an x86 based system.

Note – This guide focuses primarily on booting and shutting down a single Oracle Solaris instance on servers and workstations. Information about booting and shutting down Oracle Solaris on systems that have service processors and systems that have multiple physical domains is not covered in detail in this document. For more information, see the product documentation for your specific hardware at `http://www.oracle.com/technetwork/indexes/documentation/index.html`.

The following is a list of the information that is in this chapter:

For information about booting and shutting down a SPARC based system, see *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

What's New in Booting and Shutting Down a System

The following boot features are new the Oracle Solaris 11 release:

Administratively Provided driver.conf Files

Driver configuration files (`driver.conf`) can be supplemented with local, administrative changes without modifying the original vendor provided files in the `/kernel` and `/platform` directories. This enhancement provides better preservation of local configuration during a system upgrade. You can now provide local changes to driver configuration by adding `driver.conf` files to the new `/etc/driver/drv` directory. At boot time, the system checks for a configuration file in `/etc/driver/drv` for that driver. If found, the system automatically merges the vendor-provided configuration with the administratively provided changes.

To display these merged properties, use the `prtconf` command with the new `-u` option. The `-u` option enables you to display both the original and modified property values for a specified driver. For more information, see the `prtconf`(1M) man page. For instructions, see "How to Display Default and Customized Property Values for a Device" in *Oracle Solaris Administration: Common Tasks*.

Note – Do not edit vendor-provided `driver.conf` files in the `/kernel` and `/platform` directories. If you need to supplement a driver's configuration, the preferred method is to add a corresponding `driver.conf` file to the local `/etc/driver/drv` directory, and then customize that file. For instructions, see Chapter 5, "Managing Devices (Overview/Tasks)," in *Oracle Solaris Administration: Devices and File Systems*.

See also the following additional references:

- `driver.conf`(4)
- `driver`(4)
- *Writing Device Drivers*
- `ddi_prop_exists`(9F)
- `ddi_prop_lookup`(9F)

Bitmapped Console Support

Oracle Solaris 11 supports higher resolution and color depth on x86 based systems than the older Video Graphics Array (VGA) 640-480 16-color console. This support is provided for systems that use traditional BIOS and Video Electronics Standards Association (VESA) option read-only memory (ROM). Note that support is limited to when a graphics card or frame buffer is used as a physical or virtual console. There is no impact on the behavior of serial consoles.

For more information, see "Support for Bitmapped Console" on page 59.

Boot and Shutdown Animation

The progress status indicator that is displayed on a system during the boot process is automatically interrupted in the following instances:

- The kernel debugger is entered.
- A system panic occurs.
- A Service Management Facility (SMF) feature of Oracle Solaris service that requires input interrupts the boot process.
- The GNOME Desktop Manager (GDM) login screen appears.

During the shutdown process, if the `console=graphics` option was specified when booting the system, and the shutdown is triggered by the `X.org` server, a progress status indicator is displayed. You can prevent the progress status indicator from displaying by setting the new `splash-shutdown` property of the `svc:/system/boot-config` SMF service to `false`. For instructions, see "Disabling Shutdown Animation" on page 60.

Fast Reboot

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.

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 `true` on the x86 platform. For task-related information, including how to change the default behavior of Fast Reboot on the SPARC platform, see "Accelerating the Reboot Process on an x86 Based System" on page 44.

x86: Removal of Support for 32–Bit Kernel

In Oracle Solaris 11, 32–bit kernel support on x86 platforms has been removed. As a result, you cannot boot Oracle Solaris 11 on 32-bit x86 platforms. Systems with 32-bit hardware must either be upgraded to 64–bit hardware or continue to run Oracle Solaris 10.

Note – This removal of support does not impact 32–bit applications. Support for 32–bit applications on x86 platforms remains the same.

Booting and Shutting Down an x86 Based System (Topic Map)

Use the following references to find step-by-step instructions on various boot-related topics within this document.

TABLE 1–1 Booting and Shutting Down an x86 Based System: Topic Map

Task	For More Information
Bring an x86 based system to a specified state (run level booting).	Chapter 2, "Booting an x86 Based System to a Specified State (Tasks)"
Shut down an x86 based system.	Chapter 3, "Shutting Down a System (Tasks)"
Reboot an x86 based system.	Chapter 4, "Rebooting an x86 Based System (Tasks)"
Boot an x86 based system from the network.	Chapter 5, "Booting an x86 Based System From the Network (Tasks)"
Change the default boot behavior on an x86 based system.	Chapter 6, "Modifying Boot Parameters on an x86 Based System (Tasks)"
Create, administer and boot from a ZFS BEs, snapshots, or datasets on x86 based systems.	Chapter 7, "Creating, Administering, and Booting From ZFS Boot Environments on x86 Platforms (Tasks)"
Keep an x86 based system bootable by using the boot administration interface (`bootadm`).	Chapter 8, "Keeping an x86 Based System Bootable (Tasks)"
Troubleshoot booting an x86 based system.	Chapter 9, "Troubleshooting Booting an x86 Based System (Tasks)"

Guidelines for Booting an x86 Based System

Keep the following guidelines in mind when booting a system:

- After an x86 based system is shut down, it is booted by selecting an operating system in the GRUB menu. If no operating system is selected, the system boots the default operating system that is specified in the `menu.lst` file.
- 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.

Reasons to Boot a System

The following table lists reasons that you might need to boot an x86 based system. The system administration tasks and the corresponding boot option that is used to complete the task is also described.

TABLE 1–2 Booting a System

Reason for System Reboot	Appropriate Boot Option	For More Information
Turn off system power due to anticipated power outage.	Turn system power back on	Chapter 3, "Shutting Down a System (Tasks)"
Change kernel parameters in the `/etc/system` file.	Reboot the system to a multiuser state (run level 3 with NFS resources shared)	"Booting an x86 Based System to a Multiuser State (Run Level 3)" on page 27
Perform file system maintenance, such as backing up or restoring system data.	Press Control-D from a single-user state (run level S) to bring the system back to a multiuser state (run level 3)	"Booting an x86 Based System to a Single-User State (Run Level S)" on page 27
Repair a system configuration file such as `/etc/system`.	Interactive boot	"Booting an x86 Based System Interactively" on page 29
Add or remove hardware from the system.	Reconfiguration boot (turn on system power after adding or removing devices, if devices are not hot-pluggable)	"Setting Up Disks for ZFS File Systems (Task Map)" in *Oracle Solaris Administration: Devices and File Systems*
Recover from a hung system and force a crash dump.	Recovery boot	"How to Force a Crash Dump and Reboot of the System" on page 87

TABLE 1–2 Booting a System *(Continued)*

Reason for System Reboot	Appropriate Boot Option	For More Information
Boot the system by using the kernel debugger (`kmdb`) to track down a system problem.	Booting `kmdb`	"How to Boot a System With the Kernel Debugger Enabled (`kmdb`)" on page 88

Service Management Facility and Booting

SMF provides an infrastructure that augments the traditional UNIX startup scripts, init run levels, and configuration files. With the introduction of SMF, the boot process creates 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. To generate a message when each service is started during the boot process, use the `-v` option with the `boot` command.

When a system is being booted you can select the milestone to boot to or select the level of error messages to be recorded. For instance:

- You can choose a specific milestone to boot to using this command:

  `ok` **`boot -m milestone=`***`milestone`*

  The default milestone is `all` which starts all enabled services. Another useful milestone is `none` which starts only `init`, `svc.startd` and `svc.configd`. This milestone provides a very useful debugging environment where services can be started manually. See "How to Boot Without Starting Any Services" in *Oracle Solaris Administration: Common Tasks* for instructions on how to use the `none` milestone.

  The run-level equivalents `single-user`, `multi-user`, and `multi-user-server` are also available, but are not commonly used. The `multi-user-server` milestone, in particular does not start any services which are not a dependency of that milestone, so may not include important services.

- You can choose which level of logging for `svc.startd` using the following command:

  `ok` **`boot -m`** *`logging_level`*

  The logging levels that you can select are `quiet`, `verbose` and `debug`. See "SMF Service Error Logging" in *Oracle Solaris Administration: Common Tasks* for specific information about the logging levels.

Changes in Boot Behavior When Using SMF

Most of the features that are provided by SMF occur behind the scenes, so users are not typically aware of these features. 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.
- Because services are automatically restarted if possible, it might seem that a process fails to terminate. If the service is defective, the service is placed in maintenance mode, but normally a service is restarted if the process for the service is terminated. 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 by using SMF. Scripts and `inittab` entries that are provided by an ISV or are locally developed will continue to run. The services might not start at exactly the same point in the boot process, but they are not started before the SMF services..

How Run Levels Work

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.

Oracle Solaris 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 1–3 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.
2	Multiuser state	Multiuser	For normal operations. Multiple users can access the system and all file systems. 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.

TABLE 1–3 Oracle Solaris Run Levels *(Continued)*

Run Level	Init State	Type	Purpose
4	Alternative multiuser state	Multiuser	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 a multiuser level with NFS resources shared (or whatever run 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 1–4 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`

What Happens When a System Is Booted to a Multiuser State (Run Level 3)

1. The `init` process is started and reads the properties defined in the `svc:/system/environment:init` SMF service 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 log in to the system.
   b. Passes the startup activities to `svc.startd`.

   For a detailed description of how the `init` process uses the `inittab` file, see the `init(1M)` man page.

When to Use Run Levels or Milestones

In general, changing milestones or run levels is an uncommon procedure. If it is necessary, using the `init` command to change to a run level will change the milestone as well and is the appropriate command to use. The `init` command is also good for shutting down a system.

However, booting a system using the `none` milestone can be very useful for debugging startup problems. There is no equivalent run level to the `none` milestone. For more information, see "How to Boot Without Starting Any Services" in *Oracle Solaris Administration: Common Tasks*.

Overview of the Oracle Solaris Boot Architecture

The Oracle Solaris boot architecture includes the following fundamental 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 "Description of the Oracle Solaris Boot Archives" on page 76.

- **Use of a boot administration interface to maintain the integrity of the Oracle Solaris boot archives**

  The `bootadm` command handles the details of boot archive update and verification. 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 have been updates to the system such as drivers or configuration files, the boot archive is rebuilt to include these changes so that upon reboot, the boot archive and root file system are synchronized. You can use the `bootadm` command to manually update the boot archive. For instructions, see "Maintaining the Integrity of the Boot Archives" on page 78.

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

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

  The ramdisk image is derived from the boot archive and then transferred to the system from the boot device.

  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).

How the x86 Boot Process Works

This section describes the basic boot process on Oracle Solaris x86 platforms. For more information about boot processes on specific hardware types, including systems that have

service processors and systems that have multiple physical domains, see the product documentation for your specific hardware at http://www.oracle.com/technetwork/indexes/documentation/index.html.

When an x86 based system is powered on, the BIOS initializes the CPU, the memory, and the platform hardware. When the BIOS phase completes, the boot loader is loaded from the configured boot device and control of the system is transferred to the boot loader, which in turn starts the boot process. The *boot loader* is the first software program that runs after you turn on a system. This program starts the boot process. In Oracle Solaris the GRand Unified Bootloader, otherwise known as "GRUB", is the default boot loader on x86 based systems.

GRUB-Based Booting

In Oracle Solaris, 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 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 parameters.

The Oracle Solaris kernel is fully compliant with the Multiboot Specification. With GRUB, you can boot the different operating systems that might be installed on a single system. For example, you can individually boot Oracle Solaris, Linux, or Windows by selecting the appropriate boot entry in the GRUB menu when the system boots. Or, you can customize the `menu.lst` file to boot a specific OS instance 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, as well as the drive and the partition where the kernel resides.

GRUB Components

The components of the GRUB boot loader are as follows:

- `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 *Oracle Solaris Administration: 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 file system, where */pool-name*`/boot/grub` is the name of the ZFS storage pool. This file is read by the GRUB `stage2` file. For more information, see the section, "Modifying Boot Entries and Parameters by Editing the `menu.lst` File" on page 61.

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.

Purpose and Function of the GRUB Menu

The menu that is displayed when you boot an x86 based system is the *GRUB menu*. This menu is based on configuration information that is in the GRUB `menu.lst` file. When the boot sequence starts, the GRUB menu is displayed. Unless you interrupt the boot sequence, the default entry (typically the first entry in the `menu.lst` file) is booted by default.

You can edit the GRUB menu at boot time to either boot a different operating system or modify the parameters of the default boot entry. To do so, type `e` as soon as the GRUB menu is displayed. Typing `e` interrupts the boot process and takes you to the *GRUB edit menu*, where you can select another OS to boot or modify default boot parameters for the default boot entry. Note that the modified boot behavior persists only until the next time the system is booted. For instructions, see "Modifying Boot Parameters at Boot Time" on page 56.

GRUB Device-Naming Conventions

GRUB uses device-naming conventions that are slightly different than those used in previous releases. Understanding the device-naming conventions that GRUB uses can assist you in correctly specifying drive and partition information when you configure GRUB on your system.

The following table describes the device-naming conventions that GRUB uses.

TABLE 1–5 Conventions for GRUB Devices

Device Name	Description
(fd0)	First diskette
(fd1)	Second diskette
(nd)	Network device
(hd0,0)	First fdisk partition on first disk
(hd0,1)	Second fdisk partition on first disk
(hd0,0,a),	Slice a on first fdisk partition on first disk
(hd0,0,b)	Slice b on first fdisk partition on first disk

Note – All GRUB device names must be enclosed in parentheses.

Starting with the Solaris 10 10/08 release, the `findroot` command replaces the `root` command previously used by GRUB. The `findroot` command provides enhanced capabilities for discovering a targeted disk, irrespective of the boot device.

x86 and GRUB Boot Terminology

The following basic terminology is used for booting and shutting down an x86 based system:

Basic Input/Output System (BIOS) On x86 based systems, the BIOS is the boot firmware designed to be the first code run by a PC when turned on. The initial function of the BIOS is to identify, test, and initialize system devices such as the video display card, hard disk, floppy disk, and other hardware.

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.

boot loader The first software program that runs after you turn on a system. This program begins the booting process.

GRand Unified Bootloader (GRUB) GRUB is a multiboot boot loader that is used on x86 based systems. The boot loader is the first software program that runs when a system starts. It is responsible for loading and transferring control to the operating system kernel software (Oracle Solaris, Linux, and Windows).

GRUB edit menu	A submenu of the GRUB main menu. GRUB commands are displayed in this submenu. These commands can be edited to change boot behavior.
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.
`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.

◆ ◆ ◆ CHAPTER 2

Booting an x86 Based System to a Specified State (Tasks)

This chapter provides task-related information for booting an x86 based system to various system states, also known as *run levels*.

The following is a list of the information that is in this chapter:

- "Booting an x86 Based System to a Specified State (Task Map)" on page 25
- "Booting an x86 Based System to a Specified State" on page 26

For overview information about booting an x86 based system, see Chapter 1, "Booting and Shutting Down an x86 Based System (Overview)."

For information about booting a SPARC based system to a specified state, see Chapter 2, "Booting a SPARC Based System to a Specified State (Tasks)," in *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

Booting an x86 Based System to a Specified State (Task Map)

TABLE 2–1 Booting an x86 Based System to a Specified State: Task Map

Task	Description	For Instructions
Determine the current run level of a system.	Use the `who` command with the `-r` option to determine a system's current run level.	"Determining a System's Current Run Level" on page 26
Boot an x86 based system to a multiuser state.	Use this boot method to bring the system back to a multiuser state (run level 3) after shutting down or performing a system hardware maintenance task.	"Booting an x86 Based System to a Multiuser State (Run Level 3)" on page 27

TABLE 2–1 Booting an x86 Based System to a Specified State: Task Map *(Continued)*

Task	Description	For Instructions
Boot an x86 based system to a single-user state.	Use this boot method to perform a system maintenance task, such as backing up a file system.	"Booting an x86 Based System to a Single-User State (Run Level S)" on page 27
Boot an x86 based system interactively.	Use this boot method after making temporary changes to a system file or the kernel for testing purposes.	"Booting an x86 Based System Interactively" on page 29

Booting an x86 Based System to a Specified State

The following procedures describe how to boot an x86 based system to a specified state, also known as *run level booting*.

Determining a System's Current Run Level

To determine a system's current run level, use the `who -r` command.

EXAMPLE 2–1 Determining a System's Run Level

The output of the `who -r` command displays information about a system's current run level, as well as 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

Booting an x86 Based System to a Multiuser State (Run Level 3)

If a system is turned off, turning it on, or using the `reboot` command, starts the multiuser boot sequence.

Use the `who -r` command to verify that the system is brought to the specified run level. See "Determining a System's Current Run Level" on page 26.

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

Use this procedure to boot an x86 based 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 main menu is displayed.

2 When the GRUB main menu is displayed, press Enter to boot the default OS instance.

If you do not select 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
   .       run-level 3  Mar  2 09:44     3      0  S
```

Booting an x86 Based System to a Single-User State (Run Level S)

Booting a system to a single-user state is used for system maintenance, such as backing up a file system or troubleshooting other system issues.

How to Boot a System to a Single-User State (Run Level S)

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. **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 select the `kernel$` line.**

   If you cannot use the arrow keys, use the caret (`^`) 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 to a single-user state, type `-s` at the end of the boot entry line, then press Return to go back to the previous screen.**

   Note – 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 the system with the kernel debugger
   - Redirect the console

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

6. **To boot the system to a single-user state, type `b`.**

7. **When prompted, type the `root` password.**

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.**

Booting an x86 Based System Interactively

Booting a system interactively is useful if you need to specify an alternate kernel or the `/etc/system` file during the boot process. Use the following procedure to boot a system interactively. Alternatively, you can resolve a problem with the `/etc/system` file by booting an alternative boot environment. See "Initiating a Reboot of a System to a Newly Activated or Alternate Boot Environment" on page 45.

▼ How to Boot a System Interactively

1. **Make backup copies of the `/etc/system` and the `boot/solaris/filelist.ramdisk` files, then add the `/etc/system.bak` file name to the `/boot/solaris/filelist.ramdisk` file. For example:**

   ```
   # cp /etc/system /etc/system.bak
   # cp /boot/solaris/filelist.ramdisk /boot/solaris/filelist.ramdisk.orig
   # echo "etc/system.bak" >> /boot/solaris/filelist.ramdisk
   ```

2. **Update the boot archive.**

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

3. **Reboot the system.**

   ```
   # reboot
   ```

4. **When the GRUB menu is displayed, select the OS that you want boot interactively, then type `e`.**

5. **Using the arrow keys, select the `kernel$` line, then type `e` to edit the specified boot entry.**

6. **Type `-a` at the end of the line, then press Return.**

7. **Type `b` to boot the system interactively.**

8. **Respond the system prompts as follows:**

   a. **Specify an alternate system file, then press Return.**

      ```
      Name of system file [etc/system]: /etc/system.bak
      ```

   b. **Specify the root file system, then press Return.**

   c. **Specify the physical name of the root device, then press Return.**

   Pressing Return without providing any information accepts the system defaults.

9. **Repair the damaged `/etc/system` file.**

10 Reboot the system to run level 3.

```
# reboot
```

◆ ◆ ◆ CHAPTER 3

Shutting Down a System (Tasks)

This chapter provides overview and task-related information for shutting down a system. The procedures for shutting down an x86 based system are identical to the procedures for shutting down a SPARC based system. However, output for certain examples might vary.

The following is a list of the information that is in this chapter:

For overview information about booting an x86 based system, see Chapter 1, "Booting and Shutting Down an x86 Based System (Overview)."

For information about booting and shutting down a SPARC based system, see *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

Shutting Down a System (Task Map)

TABLE 3–1 Shutting Down a System: Task Map

Task	Description	For Instructions
Determine who is logged in to a system.	If the system is a server, use the `who` command to determine who is logged in to a system.	"How to Determine Who Is Logged in to the System" on page 34

TABLE 3–1 Shutting Down a System: Task Map *(Continued)*

Task	Description	For Instructions
Shut down a system by using the `shutdown` command.	Use the `shutdown` command with the appropriate options to shut down a system. This method is preferred for shutting down a server.	"How to Shut Down a System by Using the `shutdown` Command" on page 34
Shut down a system by using the `init` command.	Use the `init` command and indicate the appropriate run level to shut down a system.	"How to Shut Down a System by Using the `init` Command" on page 38

Overview of Shutting Down a 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 you can safely turn off 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 information about using your system's power management features, see the `poweradm`(1M) man page.

Guidelines for Shutting Down a System

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

- Use either the `shutdown` or the `init` command to shut down a system. Both commands perform a clean system shutdown, which means all system processes and services are terminated normally.
- You need to be the `root` role to use the `shutdown` and `init` commands.
- Both the `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 the multiuser state, with NFS resources shared.
- **Run level 6** – Shuts down the system to run level 0, and then reboots the system to a multiuser level with SMB or NFS resources shared (or whatever run level is the default in the `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 "How Run Levels Work" on page 17.

System Shutdown Commands

The `shutdown` and `init` commands are the primary commands that are used to shut down a system. Both commands perform a *clean shutdown* of the system. As such, all file system changes are written to 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.

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

TABLE 3–2 Shutdown Commands

Command	Description	When to Use
`shutdown`	An executable that calls the `init` program to shut down the system. The system is brought to run level S by default.	Use this command to shut down servers that are operating at run level 3.
`init`	An executable that terminates all active processes and synchronizes the disks before changing run levels.	Because this command provides a faster system shutdown, the command is preferred for shutting down stand-alone systems when other users will not be affected. There is no notification sent for an 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.

TABLE 3–2 Shutdown Commands *(Continued)*

Command	Description	When to Use
`halt`, `poweroff`	An executable that synchronizes the disks and stops the processor.	Not recommended because it does not shut down all processes or 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.

Shutting Down a System

The following procedures and examples describe how to shut down a system by using the `shutdown` and `init` commands.

▼ How to Determine Who Is Logged in to the System

For Oracle Solaris systems that are used as multiuser timesharing systems, you might need to determine if any users are logged into the system before shutting it down. Use the following procedure in these instances.

- **To determine who is logged in to a system, use the `who` command, as follows:**

```
$ 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 fourth column, if present, identifies the host name if the user is logged in from a remote system.

▼ How to Shut Down a System by Using the shutdown Command

1. **Become the `root` role.**

2. **For a server shutdown, find out if any users are logged in to the system.**

```
# who
```

   A list of all logged-in users is displayed.

3 Shut down the system.

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

`-i`*init-state*	Brings the system to an init state that is different from the default of S. The choices are 0, 1, 2, 5, and 6. Run levels 0 and 5 are states reserved for shutting the system down. Run level 6 reboots the system. Run level 2 is available as a multiuser 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 `root` password, if prompted.

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

6 After you have finished performing any 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	x86 Based System Prompt
S (single-user state)	#
0 (power-down state)	#
Run level 3 (multiuser state with remote resources shared)	*hostname* `console login:`

Example 3–1 Bringing a System to a Single-User State (Run Level S) by Using the shutdown Command

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

```
# who
root     console      Apr 15 06:20

# shutdown -g180 -y

Shutdown started.    Fri Apr 15 06:20:45 MDT 2011

Broadcast Message from root (console) on portia Fri Apr 15 06:20:46...
The system portia will be shut down in 3 minutes

showmount: portia: RPC: Program not registered
Broadcast Message from root (console) on portia Fri Apr 15 06:21:46...
The system portia will be shut down in 2 minutes

showmount: portia: RPC: Program not registered
Broadcast Message from root (console) on portia Fri Apr 15 06:22:46...
The system portia will be shut down in 1 minute

showmount: portia: RPC: Program not registered
Broadcast Message from root (console) on portia Fri Apr 15 06:23:16...
The system portia will be shut down in 30 seconds

showmount: portia: RPC: Program not registered
Changing to init state s - please wait
svc.startd: The system is coming down for administration.  Please wait.
root@portia:~# Apr 15 06:24:28 portia svc.startd[9]:

Apr 15 06:24:28 portia syslogd: going down on signal 15
svc.startd: Killing user processes.
Requesting System Maintenance Mode
(See /lib/svc/share/README for more information.)
SINGLE USER MODE

Enter user name for system maintenance (control-d to bypass):xxxxxx
#
```

Example 3–2 Bringing a System to a Shutdown State (Run Level 0) by Using the shutdown Command

In the following example, the `shutdown` command is used to bring a system to run level 0 in five 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.    Fri Apr 15 06:35:48 MDT 2011

Broadcast Message from root (console) on murky Fri Apr 15 06:35:48...
The system pinkytusk will be shut down in 5 minutes

showmount: murkey: RPC: Program not registered
showmount: murkey: RPC: Program not registered
Broadcast Message from root (console) on murkey Fri Apr 15 06:38:48...
The system murkey will be shut down in 2 minutes
```

```
showmount: murkey: RPC: Program not registered
Broadcast Message from root (console) on murkey Fri Apr 15 06:39:48...
The system murkey will be shut down in 1 minute

showmount: murkey: RPC: Program not registered
Broadcast Message from root (console) on murkey Fri Apr 15 06:40:18...
The system murkey  will be shut down in 30 seconds

showmount: murkey: RPC: Program not registered
Broadcast Message from root (console) on murkey Fri Apr 15 06:40:38...
THE SYSTEM murkey IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged

showmount: murkey: RPC: Program not registered
Changing to init state 0 - please wait
root@murkey:~# svc.startd: The system is coming down.  Please wait.
svc.startd: 122 system services are now being stopped.
Apr 15 06:41:49 murkey svc.startd[9]:
Apr 15 06:41:50 murkey syslogd: going down on signal 15
svc.startd: Killing user processes.
Apr 15 06:41:57 The system is down.  Shutdown took 69 seconds.
syncing file systems... done
Press any key to reboot.
Resetting...
```

If you are bringing the system to run level 0 to turn off power to all devices, see "Turning Off Power to System Devices" on page 39.

Example 3–3 Bringing a System to a Multiuser State (Run Level 3) by Using the shutdown Command

In the following example, the `shutdown` command is used to reboot a 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.    Fri Apr 15 06:46:50 MDT 2011

Broadcast Message from root (console) on venus Fri Apr 15 06:46:50...
The system venus will be shut down in 2 minutes

showmount: venus: RPC: Program not registered
showmount: venus: RPC: Program not registered
Broadcast Message from root (console) on venus Fri Apr 15 06:47:50...
The system venus will be shut down in 1 minute

showmount: venus: RPC: Program not registered
showmount: venus: RPC: Program not registered
Broadcast Message from root (console) on venus Fri Apr 15 06:48:20...
The system venus will be shut down in 30 seconds

showmount: venus: RPC: Program not registered
Broadcast Message from root (console) on venus Fri Apr 15 06:48:40...
```

```
THE SYSTEM venus IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged

showmount: venus: RPC: Program not registered
Changing to init state 6 - please wait
root@venus:~# svc.startd: The system is coming down.  Please wait.
svc.startd: 123 system services are now being stopped.
Apr 15 06:49:32 venus svc.startd[9]:
Apr 15 06:49:32 venus syslogd: going down on signal 15
svc.startd: Killing user processes.
Apr 15 06:49:40 The system is down.  Shutdown took 50 seconds.
syncing file systems... done
rebooting...
SunOS Release 5.11 Version 2010-12-10 64-bit
Copyright (c) 1983, 2010, Oracle and/or its affiliates. All rights reserved.
Booting to milestone "milestone/single-user:default".
Hostname: venus
NIS domain name is solaris.us.oracle.com
.
.
.
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 state, see "Booting an x86 Based System to a Multiuser State (Run Level 3)" on page 27.

▼ How to Shut Down a System by Using the init Command

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

1. **Become the `root` role.**

2. **Shut down the system.**

   ```
   # init 5
   ```

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

Example 3–4 Bringing a System to a Shutdown State (Run Level 0) by Using the init Command

In this example, the `init` command is used to bring an x86 based stand-alone system to the run 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
```

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 state, see "Booting an x86 Based System to a Multiuser State (Run Level 3)" on page 27.

Turning Off Power to System 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.

For information about turning off power to devices, see the instructions for the specified hardware in the product documentation at `http://www.oracle.com/technetwork/indexes/documentation/index.html`.

◆ ◆ ◆ CHAPTER 4

Rebooting an x86 Based System (Tasks)

This chapter describes the various methods for rebooting an x86 based system, including information about the Fast Reboot feature of Oracle Solaris.

The following is a list of the information that is in this chapter:

For overview information about booting an x86 based system, see Chapter 1, "Booting and Shutting Down an x86 Based System (Overview)."

For information about rebooting a SPARC based system, see Chapter 4, "Rebooting a SPARC Based System (Tasks)," in *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

Rebooting an x86 Based System (Task Map)

TABLE 4–1 Rebooting an x86 Based System: Task Map

Task	Description	For Instructions
Reboot a system by using the `init` command.	Use the `init` command to initiate a run level transition. When using the `init` command to reboot a system, run levels 2, 3, and 4 are available as multiuser system states.	"How to Reboot a System by Using the `init` Command" on page 43
Reboot a system by using the `reboot` command.	Use the `reboot` command to restart the kernel and bring the system to a multiuser state.	"How to Reboot a System by Using the `reboot` Command" on page 43

TABLE 4–1 Rebooting an x86 Based System: Task Map *(Continued)*

Task	Description	For Instructions
Initiate a reboot of an x86 based system, bypassing the BIOS.	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 Reboot a System Bypassing the BIOS" on page 45
Initiate a reboot of an x86 based system to a newly created boot environment.	Initiate a fast reboot of an x86 based system to a newly created or alternate boot environment by specifying that boot environment with the `reboot` command.	"Initiating a Reboot of a System to a Newly Activated or Alternate Boot Environment" on page 45
Change the default behavior of the Fast Reboot feature on an x86 based system.	On x86 platforms, both Fast Reboot and Panic Fast Reboot are enabled by default and are managed by the `boot-config` service. You can change this default behavior by disabling one or both of these features.	"Changing the Default Behavior of the Fast Reboot Feature" on page 46
Initiate a standard reboot of an x86 based system that has Fast Reboot enabled.	Use the `reboot` command with the `-p` option to perform a standard reboot of a system that has the Fast Reboot feature enabled.	"Initiating a Standard Reboot of a System That Has Fast Reboot Enabled" on page 47

Rebooting an x86 Based System

You can reboot a system by using either the `init` command or the `reboot` command.

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. The `init` command can be used to initiate a run level transition. When using the `init` command to reboot a system, run levels 2, 3, and 4 are available as multiuser system states.

The `reboot` command restarts the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to the loaded kernel. Although the `reboot` command can be used by the `root` user at anytime, in certain cases, as with the reboot of a server, the `shutdown` command is normally used first to warn all users who are logged in to the system of the impending loss of service. For more information, see Chapter 3, "Shutting Down a System (Tasks)."

How to Reboot a System by Using the init Command

The `init` command is an executable shell script that terminates all active processes on a system and then synchronizes the disks before changing run levels.

1. **Become the `root` role.**
2. **Reboot the system.**
   - **To reboot the system to the state that is defined by the `initdefault` entry in the `/etc/inittab` file, type the following command:**

     ```
     # init 6
     ```
   - **To reboot the system to a multiuser state, type the following command:**

     ```
     # init 2
     ```

Example 4–1 Bringing a System to a Single-User State (Run Level S) by Using the init Command

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

```
# 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
#
```

How to Reboot a System by Using the reboot Command

When you reboot an x86 based system by using the `reboot` command, a fast reboot is initiated by default, and the BIOS is bypassed. To reboot a system without bypassing the BIOS, use the `-p` option with the `reboot` command. See "Initiating a Standard Reboot of a System That Has Fast Reboot Enabled" on page 47.

1. **Become the `root` role.**

2 **Reboot the system.**

```
# reboot
```

Accelerating the Reboot Process on an x86 Based System

The Fast Reboot feature of Oracle Solaris 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. Fast Reboot and Panic Fast Reboot (a fast reboot of system after a system panic) are enabled by default, so there is no need to use the `-f` option with the `reboot` command to initiate a fast reboot of an x86 based system.

Fast Reboot support is facilitated by a new `boot-config` service, `svc:/system/boot-config:default`. This service provides a means for setting or changing the default boot configuration properties of a system, if required. When the `config/fastreboot_default` property is set to `true`, the system automatically performs a fast reboot. This property's value is set to `true` on an x86 based system. For more information, see "Changing the Default Behavior of the Fast Reboot Feature" on page 46.

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. 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 drivers implement the `quiesce` function. For troubleshooting instructions, see "Troubleshooting Conditions That Might Prevent Fast Reboot From Working on x86 Platforms" on page 89.

To view a demo that describes the fast reboot process in more detail, go to `http://download.oracle.com/otndocs/tech/OTN_Demos/x86/x86-OTN-Demo/x86-OTN-Demo.html`.

▼ How to Reboot a System Bypassing the BIOS

Note – In this Oracle Solaris release, Fast Reboot is the default operating mode on x86 based systems. Previously, to initiate a fast reboot of an x86 based system, you needed to specify the `-f` option with the `reboot` command to initiate a fast reboot of the system. You no longer need to specify this option.

1. **Become the `root` role.**
2. **To initiate a fast reboot of the system, type either of the following commands:**

```
# reboot

# init 6
```

Initiating a Reboot of a System to a Newly Activated or Alternate Boot Environment

There are several ways that you can perform a fast reboot of an x86 based system to an alternate boot environment. The following examples illustrate some of these methods.

EXAMPLE 4–2 x86: Initiating a Reboot of a System to a Newly ActivatedBoot Environment

The following example shows how to initiate a fast reboot of a system to a newly activated boot environment, `2010-12-10-be`.

```
# bootadm list-menu
the location for the active GRUB menu is: /rpool/boot/grub/menu.lst
default 0
0 oracle solaris 11
1 2010-12-10-be
2 zfsbe2
3 2010-12-10-be-s

# beadm activate 2010-12-10-be
# reboot
```

EXAMPLE 4–3 x86: Initiating a Reboot of a System by Specifying an Alternate Boot Environment

To fast reboot a system to an alternate boot environment, for example `zfsbe2`, you would type the following command:

```
# reboot -- 'rpool/zfsbe2'
```

To initiate a fast reboot of a system to a dataset named `rpool/zfsbe1`, you would type the following command:

EXAMPLE 4–3 x86: Initiating a Reboot of a System by Specifying an Alternate Boot Environment *(Continued)*

```
# reboot -- 'rpool/zfsbe1'
```

To initiate a fast reboot of a system to an alternate ZFS root dataset, you would type the following command:

```
# reboot -- 'rpool/ROOT/zfsroot2'
```

EXAMPLE 4–4 Initiating a Fast Reboot of a System to an Alternate Boot Environment With the Kernel Debugger Enabled

To initiate a fast reboot of a system to the `zfsbe3` boot environment with the kernel debugger enabled, you would type the following command:

```
# reboot -- 'rpool/zfsbe3 /platform/i86pc/kernel/amd64/unix -k'
```

EXAMPLE 4–5 x86: Initiating a Reboot of a System to a New Kernel

To initiate a fast reboot of a system to a new kernel named `my-kernel`, you would type the following command:

```
# reboot -- '/platform/i86pc/my-kernel/amd64/unix -k'
```

EXAMPLE 4–6 x86: Initiating a Reboot of a Mounted Disk or a Mounted Dataset

To initiate a fast reboot of a mounted disk or a mounted dataset, you would type the following command:

```
# reboot -- '/mnt/platform/i86pc/my-kernel/amd64/unix -k'
```

EXAMPLE 4–7 x86: Initiating a Reboot of a System to a Single-User State With the Kernel Debugger Enabled

To initiate a fast reboot of a system to a single-user state with the kernel debugger enabled, you would type the following command:

```
# reboot -- '-ks'
```

Changing the Default Behavior of the Fast Reboot Feature

The Fast Reboot feature is controlled 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 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 `true` on an x86 based system.

The `svc:/system/boot-config:default` service consists of the following properties:

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

EXAMPLE 4–8 x86: Configuring Properties of the boot-config Service

The properties that are part of the `boot-config` service can be configured by using the `svccfg` and `svcadm` commands.

For example, to disable the default behavior of the `fastreboot_onpanic` property on an x86 based system, you would set the property's value to `false`, as shown here:

```
# svccfg -s "system/boot-config:default" setprop config/fastreboot_onpanic=false
# svcadm refresh svc:/system/boot-config:default
```

Note that changing one property's value does not affect the default behavior of the other property.

For information about managing the boot configuration service through SMF, see the `svcadm`(1M) and `svccfg`(1M) man pages.

Initiating a Standard Reboot of a System That Has Fast Reboot Enabled

To reboot an x86 based system that has the Fast Reboot feature enabled, without reconfiguring the `boot-config` service to disable the feature, use the `-p` option with the `reboot` command, as shown here:

```
# reboot -p
```

◆ ◆ ◆ CHAPTER 5

Booting an x86 Based System From the Network (Tasks)

This chapter provides overview, guidelines, and task-related information for booting an x86 based system from the network.

The following is a list of the information that is in this chapter:

For overview information about booting an x86 based system, see Chapter 1, "Booting and Shutting Down an x86 Based System (Overview)."

For information about booting a SPARC based system from the network, see Chapter 5, "Booting a SPARC Based System From the Network (Tasks)," in *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

Booting an x86 Based System From the Network (Task Map)

TABLE 5–1 Booting an x86 Based System From the Network: Task Map

Task	Description	For Instructions
1. Review the requirements for booting an x86 based system from the network.	Review all of the requirements for booting an x86 based system from the network first. Note that some requirements include separate, prerequisite tasks that are to be performed before you can boot the system from the network.	"Requirements for Booting an x86 Based System From the Network" on page 50

TABLE 5–1 Booting an x86 Based System From the Network: Task Map *(Continued)*

Task	Description	For Instructions
2. Boot an x86 based system from the network.	After reviewing all of the requirements and performing any preliminary tasks, you are ready to boot the system from the network. Use the `reboot` command to boot an x86 based system from the network.	"How to Boot an x86 Based System From the Network" on page 51

Booting an x86 Based System from the Network

You might need to boot a system from the network for the following reasons:

- To install Oracle Solaris
- For recovery purposes

The network configuration boot strategy that is used in Oracle Solaris is the Dynamic Host Configuration Protocol (DHCP).

If you booting a system from the network to install Oracle Solaris by using the Automated Installer (AI), perform any additional AI install services. For more information, see *Installing Oracle Solaris 11 Systems*.

x86 Network Boot Processes

Network booting in Oracle Solaris is supported through firmware that is compliant with the Preboot eXecution Environment (PXE), also known as *Pre-Execution Environment*, which is an environment for booting a system by using a network interface independent of data storage devices (like hard disks) or installed operating systems. This firmware is responsible for loading the boot program, which is a special GRUB Stage 2 file named `pxegrub`. The `pxegrub` file includes the basic implementations of the Trivial File Transfer Protocol (TFTP), DHCP, User Datagram Protocol (UDP), Internet Protocol (IP), and a mini-driver that uses the Universal Network Device Interface (UNDI) firmware interfaces to transfer packets across the network.

Requirements for Booting an x86 Based System 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 or to install Oracle Solaris. You can boot an x86 based system directly from a network that supports the PXE network boot protocol. Note that the PXE protocol is available only on devices that implement the Intel Preboot Execution Environment specification.

The default network boot strategy that is used for both PXE and non-PXE devices is DHCP. To perform a network boot of an x86 based system to install Oracle Solaris or for recovery purposes, a DHCP server that is configured for PXE clients is required. A boot server that provides `tftp` service is also required. If no PXE or DHCP server is available, you can load GRUB from a diskette, a CD-ROM, or a local disk.

The DHCP server supplies the information that the client needs to configure its network interface. If you are setting up an Automated Installer (AI) server, that server can also be the DHCP server. Or, you can set up a separate DHCP server. For more information about DHCP, see Part II, "DHCP," in *Oracle Solaris Administration: IP Services*.

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 Oracle Solaris.

▼ How to Boot an x86 Based System From the Network

Before You Begin

- Perform any prerequisite tasks for setting up DHCP configuration. See "Requirements for Booting an x86 Based System From the Network" on page 50.
- If you booting an x86 based system from the network to install Oracle Solaris, you must download the AI client image and create an install service based on that image. For prerequisites and further instructions, see Part III, "Installing Using an Install Server," in *Installing Oracle Solaris 11 Systems*.

1 Reboot the system.

2 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.**

3 **When the GRUB menu is displayed, select the network installation image that you want to install.**

◆ ◆ ◆ CHAPTER 6

Modifying Boot Parameters on an x86 Based System (Tasks)

This chapter provides task-related information about modifying the boot parameters on an x86 based system.

The following is a list of the information that is in this chapter:

If you need to configure x86 boot mode properties on an Oracle Integrated Lights Out Manager (ILOM) service processor, see the hardware documentation at `http://download.oracle.com/docs/cd/E19694-01/E21741-02/index.html`.

For overview information about booting an x86 based system, see Chapter 1, "Booting and Shutting Down an x86 Based System (Overview)."

For information about modifying boot parameters on a SPARC based system, see Chapter 6, "Modifying Boot Parameters on a SPARC Based System (Tasks)," in *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

Modifying Boot Parameters on an x86 Based System (Task Map)

TABLE 6–1 Modifying Boot Parameters on an x86 Based System: Task Map

Task	Description	For Instructions
Display default boot parameters on an x86 based system.	Specify the appropriate parameter of the `eeprom` command to display its value.	"Displaying and Setting Boot Parameters by using the `eeprom` Command" on page 55

TABLE 6–1 Modifying Boot Parameters on an x86 Based System: Task Map *(Continued)*

Task	Description	For Instructions
Modify boot parameters on an x86 based system by using the `eeprom` command.	Modify boot parameters on an x86 based system by using the `eeprom` command. Boot parameters that are set by using the `eeprom` command persist over a system reboot, unless these options are overridden by editing the GRUB menu at boot time.	"How to Modify Boot Parameters by Using the `eeprom` Command" on page 56
Modify boot parameters on an x86 based system at boot time.	Modify boot parameters by editing the GRUB menu at boot time. Boot options that are specified by editing the GRUB menu at boot time only persist until the next time the system is booted.	"x86: How to Modify Boot Parameters at Boot Time" on page 58
Configure console parameters on an x86 based system at boot time.	The Oracle Solaris release supports higher resolution and color depth on x86 based systems than the older Video Graphics Array (VGA) 640–480 16–color console. To modify console settings, specify the appropriate command-line `-B console=`*val* parameter at boot time.	Example 6–2 and Example 6–3
Disable the default shutdown animation behavior.	To prevent the progress status indicator from displaying, during shutdown, set the new `splash-shutdown` property of the `svc:/system/boot-config` SMF service to `false`.	"Disabling Shutdown Animation" on page 60
Modify boot parameters on an x86 based system by editing the `menu.lst` file.	Modify boot parameters by editing the `menu.lst` configuration file to add new OS entries or redirect the console. Changes that you make to the file persist over system reboots.	"Modifying Boot Entries and Parameters by Editing the `menu.lst` File" on page 61
Add a Linux entry to the `menu.lst` file after installing Oracle Solaris.	If you install Linux on one partition first and then install Oracle Solaris on another partition afterwards, you will need to follow special instructions to ensure that the GRUB menu information from the new installation does not erase the GRUB menu information from a previous installation.	"How to Add a Linux Entry to the GRUB Menu After Installing Oracle Solaris" on page 62

TABLE 6–1 Modifying Boot Parameters on an x86 Based System: Task Map *(Continued)*

Task	Description	For Instructions
Locate the active GRUB menu and list menu entries.	Use the `bootadm` command to view the location of the active GRUB menu and display the menu entries.	"How to Locate the Active GRUB Menu and List Current Menu Entries" on page 64
Set the default entry for the active GRUB menu.	Use the `bootadm` command to set the default entry for the active GRUB menu on a system.	"How to Set the Default Boot Entry for the Active GRUB Menu" on page 64

Modifying Boot Parameters on an x86 Based System

The primary methods for modifying boot parameters 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 parameters. 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 parameters 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 parameters (`boot-file`, `boot-arguments`, and `console`) cannot 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 default kernel usage 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 editing the GRUB configuration file

Displaying and Setting Boot Parameters by using the eeprom Command

To determine the default values for a specific boot parameter, use the `eeprom` command, as follows:

```
$ eeprom parameter
```

For example, to display the default value for the `boot-device` parameter, you would type:

```
$ eeprom boot-device
```

Note – You do not need to be the `root` user to display boot parameters. However, to change boot parameters or any other parameters by using the `eeprom` command, you must become the `root` user.

▼ How to Modify Boot Parameters by Using the eeprom Command

1. **Become the `root` role.**
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 6–1 Setting the auto-boot Parameter by Using the eeprom Command

The following example shows how to set the `auto-boot` boot parameter to true.

```
# eeprom auto-boot?=true
```

When the `eeprom` command is run in user mode, any parameters that have a trailing question mark (`?`) need to be enclosed in double quotation marks to prevent the shell from interpreting the question mark. Preceding the question mark with an escape character (`\`) also prevents the shell from interpreting the question mark. For example:

```
# eeprom "auto-boot?"=true
```

Modifying Boot Parameters at Boot Time

To modify a system's boot behavior at boot time, for example to boot a system to a single-user state or with the kernel debugger enabled, when the GRUB menu is displayed, interrupt the boot process by typing `e` to edit the boot entry in the GRUB menu.

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 to a single-user state.
`-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 parameters.

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

`-B acpi-enum=off`	Disables the Advanced Configuration and Power Interface (ACPI) enumeration of devices.
`-B acpi-user-options=0x2`	Disables ACPI entirely.
`-B console=force-text`	Specifies to use VGA text mode for booting. See "Support for Bitmapped Console" on page 59.
`-B console=graphics`	Specifies that the console use graphics mode for booting, which enables a high-resolution state.
`-B console=text`	Specifies that the console use text mode for booting, which enables a high-resolution state.

`-B screen-#columns=`*value*`, screen-#rows=` *value*	Specifies the number of rows and columns of the frame buffer console. The most appropriate font for the selected number of rows or columns is automatically detected by the system. This option is used to optimize the frame buffer console size. See "Support for Bitmapped Console" on page 59.
`-B console=ttya`	Redirects the console to `ttya`.
`-B console=ttya,acpi-enum=off`	Redirects the console to `ttya` and disables the ACPI enumeration of devices.

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

x86: How to Modify Boot Parameters at Boot Time

When you modify the default kernel usage by editing the GRUB menu at boot time, the changes do not persist over a system reboot. The default boot parameters are 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$` line in the 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 effect when the system is booted.

Support for Bitmapped Console

Oracle Solaris 11 supports higher resolution and color depth on x86 based systems than the older Video Graphics Array (VGA) 640-480 16-color console. This support is provided for systems that use traditional BIOS and Video Electronics Standards Association (VESA) option read-only memory (ROM). Note that support is limited to when a graphics card or frame buffer is used as a physical or virtual console. There is no impact on the behavior of serial consoles.

To support this feature, two command-line `-B` *option=val* parameters are available:

`-B console=force-text`	Specifies to use VGA text mode for booting.
`-B screen-#columns=`*value*`, screen-#rows=`*value*	Specifies the number of rows and columns of the frame buffer console. The most appropriate font for the selected number of rows or columns is automatically detected by the system. This option is used to optimize the frame buffer console size.

By default, GRUB detects a resolution and color depth that works with the installed video card and monitor. However, a different resolution can be specified, for example, a higher resolution and different color depth.

GRUB supports the following two methods for specifying video mode:

`vbeset` *hexmode*

Specifies the hex code of the desired VESA mode. To obtain a list of all of the modes that are supported by the card and monitor, use the `vbeprobe` command at the GRUB command prompt, which displays a list similar to the following:

```
0x117: Direct Color, 1024x768x16
0x118: Direct Color, 1024x768x32
0x11a: Direct Color, 1280x1024x16
0x11b: Direct Color, 1280x1024x32
[...]
```

A `vbeset` entry that specifies a 1024x768x32 configuration is shown as follows:

```
vbset 0x118
```

The `vbeset` entry must be specified after the `kernel$` and `module$` entries in the GRUB menu.

`vbematch` *xres yres depth*

Directs GRUB to search for the specified configuration, for example 1024x768x32. If found, GRUB sets the configuration that is specified.

When used instead of a `vbeset` entry, a `vbematch` entry for a 1024x768x32 configuration looks like the following:

```
vbematch 1024 768 32
```

A `vbematch` entry must be specified after the `kernel$` and `module$` entries in the GRUB menu.

EXAMPLE 6–2 x86: Configuring Text Mode Boot Parameters for the Console

In text mode, the console output is sent to the frame buffer, and input is received from the keyboard. A variant of text mode, the graphics mode displays an image with an animation until either a key is pressed or console interaction is required by the console `login`, `sulogin`, or `kmdb` command. A new property of text, `force-text`, directs the system to not use a VGA adapter as a bitmap device and sets the adapter to VGA text mode.

When this property is not present, the console device reverts to the device that is specified by the `input-device` and `output-device` property pair. When neither the console property, nor the `input-device` and `output-device` property pair are present, the console defaults to the frame buffer and keyboard.

The following example shows how to specify the `-B console=force-text` property on the kernel command line at boot time:

```
-B console=force-text
```

EXAMPLE 6–3 x86: Enabling a Graphical Display and Configuring Console Text Mode Parameters

By default, the console text mode is 80 columns by 24 rows. To reconfigure this parameter, use the `-B` option with the `screen-#columns=`*value* and `screen-#rows=`*value* parameters.

For example, the following parameters can be specified on the kernel command line to enable a graphical display and allocate a console terminal of 100 columns by 60 rows:

```
-B console=graphics, screen-#columns=100,screen-#rows=60
```

Disabling Shutdown Animation

During the shutdown process, if the `console=graphics` option was used to boot the system, and the shutdown is triggered by the `Xorg` server, a progress status indicator is displayed. To prevent the progress status indicator from displaying, set the new `splash-shutdown` property of the `svc:/system/boot-config` SMF service to `false`, as follows:

```
# svccfg -s svc:/system/boot-config:default setprop config/splash_shutdown = false
# svcadm refresh svc:/system/boot-config:default
```

Modifying Boot Entries and Parameters by Editing the menu.lst File

The GRUB menu, which is based on entries in the `menu.lst` configuration file, can be customized. Solaris automatically manages GRUB `menu.lst` entries for Oracle Solaris boot environments (BEs). As boot environments are created (either by the packaging system or explicitly with the `beadm` command), GRUB entries are added to the `menu.lst` file. When boot environments are removed by using the `beadm destroy` command with the destroy subcommand, the corresponding entries are removed from the GRUB `menu.lst` file. Oracle Solaris does not automatically add `menu.lst` entries for other operating systems that you have installed on your system. You must manually add menu entries for those operating systems. For more information, see "How to Add a Linux Entry to the GRUB Menu After Installing Oracle Solaris" on page 62.

A typical `menu.lst` file for Oracle Solaris 11 could include the following information:

```
#---------- ADDED BY BOOTADM - DO NOT EDIT ----------
title Oracle Solaris 11     1
findroot (pool_rpool,0,a)     2
bootfs rpool/ROOT/solaris     3
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS,console=graphics     4
module$ /platform/i86pc/$ISADIR/boot_archive     5
#---------------------END BOOTADM--------------------
```

1. Specifies the title of the operating system for the menu entry.
2. Searches all partitions for the *SIGNATURE* file name. In this example, the file name is `pool_rpool`).

   GRUB searches only in the `/boot/grub/bootsign` directory for the file name and then stops as soon as the first instance of the file is found. To be useful, the name of the signature file must be unique across all partitions. After locating the signature file, GRUB invokes the `root`command on that partition. To optimize the search, an optional partition and slice can be specified.
3. Sets the current ZFS boot file system to the specified value. In this example of the `menu.lst` file, the property sets the current ZFS boot file system to `rpool/ROOT/solaris`.
4. Loads the primary boot image from the specified path. The rest of this line is passed verbatim as the kernel command line. The dollar sign (`$`) is an expansion of the `$ISADIR` entry.
5. Loads the boot archive for the given kernel. The dollar sign (`$`) is an expansion of the `$ISADIR` entry.

Note – To learn more about specific GRUB commands, type `help` *command* from the command line.

A configurable timeout is available to boot the default operating system. The default entry that is booted is configurable through the `default` command. The installation software typically sets the command to boot one of the valid boot entries. To boot a different version of Oracle Solaris (if applicable), or to boot another operating system besides Oracle Solaris, use the arrow keys to select that boot entry, then press Enter to boot that operating system. Note that if the `default` command is not set, the first boot entry in the GRUB menu will boot.

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.

How to Add a Linux Entry to the GRUB Menu After Installing Oracle Solaris

If you are setting up a boot environment in such a way that you install Linux on one partition first and Oracle Solaris on another partition afterwards, you will need to follow special instructions to ensure that the GRUB menu information from the new installation does not erase the GRUB menu information from a previous installation. The following procedure describes how to manually update the `menu.lst` file to include a Linux entry from a previous installation. These instructions assume that you have already installed Linux on your system and then installed Oracle Solaris afterwards.

1 **At the completion of the Linux installation, copy the active `menu.lst` file to a USB drive, so you can reuse the information after you have completed the Oracle Solaris installation.**

Typically, this file is `/boot/grub/menu.lst`.

- **If you are unsure about the location of the active `menu.lst` file, use the `bootadm` command to locate the file:**

```
# bootadm list-menu
```

- **If you are unsure about the location of the USB drive, use the `mount` command, with no options, to determine where the USB drive is mounted. Then, copy the `menu.list` file to that location.**

2 After the installation has completed, edit the active `menu.lst` file, as follows:

a. Open a terminal window and become the `root` role.

```
$ su root
Password:
```

b. Using a text editor, edit the `menu.lst` file.

For example:

```
# vi /pool-name/boot/grub/menu.lst
```

where *pool-name* is the name of the ZFS storage pool.

c. Using the USB drive that you copied the `menu.lst` file to in Step 1, copy the Linux `menu.lst` information from the original Linux installation to the end of new `menu.lst` file.

For example, the `menu.lst` file from an Ubuntu installation would look similar to the following:

```
title        Ubuntu 8.04, kernel 2.6.24-18-generic
    root        (hd0,4)
    kernel        /vmlinuz-2.6.24-18-generic \
root=UUID=1ed7fa17-6d77-4b49-be1a-22481310fd1b ro quiet splash
    initrd        /initrd.img-2.6.24-18-generic
    quiet

    title        Ubuntu 8.04, kernel 2.6.24-18-generic (recovery mode)
    root        (hd0,4)
    kernel        /vmlinuz-2.6.24-18-generic \
root=UUID=1ed7fa17-6d77-4b49-be1a-22481310fd1b ro single
    initrd        /initrd.img-2.6.24-18-generic
```

Caution – Do not directly edit the original contents of the `menu.lst` file. Always add new information to the end of the file, or make changes by duplicating the existing content, then modify that content.

d. Save and exit the file.

3 Reboot the system.

When the system reboots, the GRUB menu should include entries for both the Linux and Oracle Solaris operating systems.

Displaying and Setting Parameters for Boot Entries by Using the bootadm Command

▼ 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` role.**
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` default entry number, and the title of each entry, is included in the output.

Example 6–4 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 2010-12-10-be
1 Oracle Solaris 11
2 Linux
```

▼ How to Set the Default Boot Entry for the Active GRUB Menu

1. **Become the `root` role.**
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`.

   menu-entry Specifies the GRUB menu entry to set as the default.

3. **To verify that the default menu entry has been changed, type:**

   ```
   # bootadm list-menu
   ```

   The new default menu entry should be displayed.

Example 6–5 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 Linux, menu entry 2.

```
# bootadm set-menu default=2
```

◆ ◆ ◆ CHAPTER 7

Creating, Administering, and Booting From ZFS Boot Environments on x86 Platforms (Tasks)

This chapter describes how to create, administer, and boot from a ZFS boot environment, also called a *BE*, on an x86 based system.

The following is a list of the information that is in this chapter:

For overview information about booting an x86 based system, see Chapter 1, "Booting and Shutting Down an x86 Based System (Overview)."

For information about booting a from a ZFS boot environment on SPARC platforms, see Chapter 7, "Creating, Administering, and Booting From ZFS Boot Environments on SPARC Platforms (Tasks)," in *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

For detailed information about managing boot environments, see *Creating and Administering Oracle Solaris 11 Boot Environments*.

Creating, Administering, and Booting From ZFS Boot Environments (Task Map)

TABLE 7–1 Creating, Administering, and Booting From ZFS Boot Environments: Task Map

Task	Description	For Instructions
Create a new boot environment.	Create a new boot environment by using the `beadm create` command.	"How to Create a New Boot Environment" on page 69

TABLE 7–1 Creating, Administering, and Booting From ZFS Boot Environments: Task Map *(Continued)*

Task	Description	For Instructions
Create a snapshot of a boot environment.	Create a snapshot of an existing boot environment by using the `beadm create` *beName*`@`*snapshot* command.	"How to Create a Snapshot of a Boot Environment" on page 71
Create a boot environment from an existing snapshot.	Create a new boot environment from an existing snapshot by using the `beadm` command.	"How to Create a Boot Environment From an Existing Snapshot" on page 71
Activate a newly created boot environment.	Activate a newly created boot environment by using the `beadm activate` command.	"How to Activate a Newly Created Boot Environment" on page 71
Display a list of boot environments, snapshots, and datasets.	To display a list of boot environments, snapshots, and datasets, use the `beadm list` command.	"How to Display a List of Available Boot Environments, Snapshots, and Datasets" on page 72
Destroy a boot environment.	Destroy a boot environment by using the `beadm destroy` command.	"How to Destroy a Boot Environment" on page 73
Boot from a specified boot environment, dataset, or root file system on an x86 based 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 boot environment contains the `-B $ZFS-BOOTFS` boot argument. The system therefore automatically boots from a ZFS root. **Note –** This option is supported *only* for boot devices that contain a ZFS pool.	"Booting From a ZFS Boot Environment or Root File System on x86 Platforms" on page 74

Creating and Administering Boot Environments

The following tasks describe how to create and administer boot environments, snapshots, and datasets by using the `beadm` utility.

- A *boot environment* (BE) is a ZFS file system that is designated for booting. A boot environment is essentially a bootable instance of the Oracle Solaris OS image, plus any other software packages that are installed into that image. You can maintain multiple boot environments on a single system. Each boot environment can have different OS versions installed. When you install Oracle Solaris, a new boot environment is automatically created during the installation.
- A *snapshot* is a read-only image of a dataset or boot environment that is taken at a given point in time. Note that a snapshot is not bootable. However, you can create a boot environment that is based on a particular snapshot and then activate that new boot environment so that it becomes the default boot environment upon the next system reboot.
- A *dataset* is a generic term that is used to identify a ZFS file system, clone, snapshot, or volume.
- *Shared datasets* are user-defined directories, such as `/export`, that contain the same mount point in both the active and inactive boot environments. Shared datasets are located outside the root dataset area of each boot environment.
- A boot environment's *critical datasets* are included within the root dataset area for that environment.

For more information about the `beadm` utility, see the `beadm(1M)` man page. For more information about managing boot environments, see *Creating and Administering Oracle Solaris 11 Boot Environments*. For specific information about using the `beadm` utility in a global or non-global zones environment, see Chapter 2, "beadm Zones Support," in *Creating and Administering Oracle Solaris 11 Boot Environments*.

▼ How to Create a New Boot Environment

1 Become the `root` role.

2 Create a boot environment by using the `beadm create` command.

```
# beadm create beName
```

where *beName* is a variable for the name of the new boot environment. This new boot environment is inactive.

Note – The `beadm create` command does not create a partial boot environment. Either a new, full boot environment is successfully created, or the command fails.

3 (Optional) Mount the new boot environment.

`# beadm mount` *beName mountpoint*

If the directory for the mount point does not exist, the `beadm` command creates the directory, then mounts the boot environment on that directory. If the boot environment is already mounted, the `beadm mount` command fails and does not remount the boot environment at the new location.

The boot environment is mounted, but remains inactive. Note that you can upgrade a mounted, inactive boot environment. Also, remember to unmount the boot environment before rebooting your system.

4 (Optional) To boot from the new boot environment, first activate the boot environment.

`# beadm activate` *beName*

where *beName* is a variable for the name of the boot environment to be activated. Upon reboot, the newly active boot environment becomes the default boot entry that is listed in the GRUB menu.

Example 7–1 Creating a Cloned Boot Environment With Shared Datasets

The following example shows the datasets in a newly created boot environment named `BE2`. The original boot environment in this example is `BE1`. The new boot environment, `BE2`, contains separate datasets that were cloned from `BE1`. If `BE1` contains separate datasets for traditional file systems, such as `/opt`, then those datasets are also cloned.

```
# beadm create BE2
# beadm list -a BE2
BE/Dataset/Snapshot Active Mountpoint Space Policy Created
------------------- ------ ---------- ----- ------ -------
BE2
   rpool/ROOT/BE2   -      -          42.0K static 2011-04-07 10:56
```

As shown in the previous output, the name of the storage pool is `rpool`. The pool already exists on the system, as it was previously set up by the initial installation or an upgrade. `ROOT` is a special dataset that was also created previously by the initial installation or upgrade. `ROOT` is reserved exclusively for use by boot environment roots.

How to Create a Snapshot of a Boot Environment

1. **Become the `root` role.**

2. **Create the snapshot of the boot environment.**

   ```
   # beadm create beName@snapshot
   ```

   Example snapshot names include the following:

   - `BE@0312200.12:15pm`
   - `BE2@backup`
   - `BE1@march132008`

How to Create a Boot Environment From an Existing Snapshot

1. **Become the `root` role.**

2. **Create a new boot environment from a snapshot by typing the following command:**

   ```
   # beadm create -e BEname@snapshotdescription beName
   ```

   Replace *BEname@snapshotdescription* with the name of an existing snapshot and *beName* with a custom name for the new boot environment.

   For example:

   ```
   # beadm create -e BE1@now BE2
   ```

   This command creates a new boot environment named `BE2` from the existing snapshot named `BE1@now`. You can then active the boot environment. For instructions, see "How to Activate a Newly Created Boot Environment" on page 71.

How to Activate a Newly Created Boot Environment

You can activate a newly created boot environment so that upon reboot it is the default boot environment that is booted. Note that only one boot environment can be active at any given time.

1. **Become the `root` role.**

2. **Activate an existing, inactive boot environment by using the following command:**

   ```
   # beadm activate beName
   ```

   where *beName* is a variable for the boot environment to be activated.

Note the following:

- The `beadm activate` *beName* command activates the boot environment by setting the `bootfs` bootable pool property to the value of the `ROOT` dataset of the boot environment that is being activated.
- The `beadm activate` command sets the newly activated boot environment as the default in the `menu.lst` file.

3 Reboot the system.

The newly activated boot environment is now the default entry in the x86 GRUB menu.

Note – If the boot environment fails to boot, reboot and select the previous boot environment from the GRUB menu or the boot menu.

▼ How to Display a List of Available Boot Environments, Snapshots, and Datasets

To display available boot environments, snapshots, and datasets that were created by using the `beadm` command, use the `beadm list` command.

1 Become the `root` role.

2 To list all of the available datasets on the system, type the following command:

`# beadm list` *option*

`-a` Lists all available information about the boot environment. This option includes subordinate snapshots and datasets.

`-d` Lists information about a boot environment's datasets.

`-s` Lists information about a boot environment's snapshots. This option is used in conjunction with the `-d` option.

`-H` Omits the header information from the display. Choosing this option results in a display that can be more easily parsed for scripts or other programs.

3 To list the available datasets for a specific boot environment, include the boot environment name in the `beadm list` command syntax.

For example, to list all of the available datasets in the `oracle-solaris` boot environment, you would type the following command:

```
# beadm list -a oracle-solaris
 BE/Dataset/Snapshot   Active Mountpoint Space  Policy Created
-------------------   ------ ---------- -----  ------ -------
oracle-solaris
   rpool/ROOT/solaris -      -          14.33M static 2011-01-20 07:45
```

Example 7–2 Viewing Snapshot Specifications

The following `beadm list` example includes the `-s` option, which displays information for any snapshots that exist on the current image.

In the following sample results, each snapshot title includes a time stamp, indicating when that snapshot was taken.

```
# beadm list -s test-2
```

The sample results are displayed.

```
BE/Snapshot      Space Policy Created
---------- ----- ------ -------
test-2
test-2@2010-04-12-22:29:27 264.02M static 2010-04-12 16:29
test-2@2010-06-02-20:28:51 32.50M static 2010-06-02 14:28
test-2@2010-06-03-16:51:01 16.66M static 2010-06-03 10:51
test-2@2010-07-13-22:01:56 25.93M static 2010-07-13 16:01
test-2@2010-07-21-17:15:15 26.00M static 2010-07-21 11:15
test-2@2010-07-25-19:07:03 13.75M static 2010-07-25 13:07
test-2@2010-07-25-20:33:41 12.32M static 2010-07-25 14:33
test-2@2010-07-25-20:41:23 30.60M static 2010-07-25 14:41
test-2@2010-08-06-15:53:15 8.92M static 2010-08-06 09:53
test-2@2010-08-06-16:00:37 8.92M static 2010-08-06 10:00
test-2@2010-08-09-16:06:11 193.72M static 2010-08-09 10:06
test-2@2010-08-09-20:28:59 102.69M static 2010-08-09 14:28
test-2@install 205.10M static 2010-03-16 19:04
```

▼ How to Destroy a Boot Environment

If you want to make more disk space available on your system, you can use the `beadm` command to destroy (remove) an existing boot environment.

Note the following:

- You cannot destroy the boot environment that is currently booted.
- The `beadm destroy` command automatically removes the destroyed boot environment's entry from the x86 GRUB menu.
- The `beadm destroy` command destroys only the critical or nonshared datasets of the boot environment. Shared datasets are located outside of the boot environment root dataset area and are not affected when a boot environment is destroyed.

1 Become the `root` role.

2 To destroy a boot environment, type the following command:

```
# beadm destroy beName
```

You are prompted for confirmation before destroying the boot environment.

`beadm destroy` *beName*	Destroys the boot environment that is specified by *beName*.
`-F`	Forces the destruction of the boot environment without a confirmation request.
`-f`	Forces the destruction of the boot environment, even if it is mounted.

Booting From a ZFS Boot Environment or Root File System on x86 Platforms

The following entries are added to the */pool-name*`/boot/grub/menu.lst` file during the installation process or during the `beadm activate` operation to boot ZFS automatically:

```
title 2010-12-10-be-s
findroot (pool_rpool,0,a)
bootfs rpool/ROOT/2010-12-10-be_152
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS -s
module$ /platform/i86pc/$ISADIR/boot_archive
```

If the device that is identified by GRUB as the boot device contains a ZFS storage pool, the `menu.lst` file is used to create the GRUB menu. On an x86 based system with multiple ZFS boot environments, you can select a boot environment from the GRUB menu during boot time. If the root file system that corresponds to this menu entry is a ZFS dataset, the following option is added:

```
-B $ZFS-BOOTFS
```

The `$ZFS-BOOTFS` keyword enables you to boot from an Oracle Solaris ZFS root file system on an x86 based system. This option identifies which boot environment or 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, contains this information by default.

EXAMPLE 7–3 Booting From a ZFS boot environment, Dataset, or File System

When booting from a ZFS file system, the root device is specified by the `-B $ZFS-BOOTFS` boot parameter on the `kernel$` line in the GRUB menu. This value, similar to all parameters specified by the `-B` option, is passed from GRUB to the kernel. For example:

```
title Oracle Solaris 11 Express snv_152
findroot (pool_rpool,0,a)
bootfs rpool/ROOT/solaris
kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/$ISADIR/boot_archive
```

◆ ◆ ◆ CHAPTER 8

Keeping an x86 Based System Bootable (Tasks)

This chapter describes how the keep an x86 based system bootable by using the boot administration interface (`bootadm`). Procedures for displaying information about the boot archive and for maintaining the integrity of boot archive, as well as troubleshooting boot archive issues, are described.

The following is a list of the information that is in this chapter:

For overview information about booting an x86 based system, see Chapter 1, "Booting and Shutting Down an x86 Based System (Overview)."

For information about keeping a SPARC based system bootable, see Chapter 8, "Keeping a SPARC Based System Bootable (Tasks)," in *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

Keeping an x86 Based System Bootable (Task Map)

TABLE 8–1 Keeping an x86 Based System Bootable: Task Map

Task	Description	For Instructions
List the contents of the boot archive by using the `bootadm` command.	Use the `bootadm list-archive` command to list the contents of the boot archive.	"How to List the Contents of the Boot Archive" on page 77

TABLE 8–1 Keeping an x86 Based System Bootable: Task Map *(Continued)*

Task	Description	For Instructions
Manage the `boot-archive` service.	The `boot-archive` service is controlled by the SMF. Use the `svcs` command to verify whether the `boot-archive` service is running. Use the `svcadm` command to enable or disable the service.	"Managing the Boot Archive SMF Service" on page 77
Clear a boot archive update failure on an x86 based system 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`.	"How to Clear a Failed Automatic Boot Archive Update by Using the `auto-reboot-safe` Property" on page 78
Clear a boot archive update failure on an x86 based system by using the `bootadm` command.	Use this procedure to manually clear boot archive update failures on an x86 based system, if the `auto-reboot-safe` property is enabled.	"How to Clear a Failed Automatic Boot Archive Update by Manually Updating the Boot Archive" on page 79

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.

In addition, the `bootadm` command handles the details of boot archive update and verification. During the process of a normal system shutdown, the shutdown process compares the boot archive's contents with the root file system. If there have been updates to the system such as drivers or configuration files, the boot archive is rebuilt to include these changes so that upon reboot, the boot archive and root file system are synchronized.

Obtaining Information About the Location and Contents of the x86 Boot Archive

The files in the x86 boot archive are located in the `/platform/i86pc/amd64/boot_archive` directory. You can list the contents of the boot archive by using the `bootadm list-archive` command, as described in the following procedure. Whenever any files in the boot archive are updated, the archive must be rebuilt. For modifications to take effect, the rebuild of the archive must take place before the next system reboot.

▼ How to List the Contents of the Boot Archive

1. **Become the `root` role.**
2. **To list the files and directories that are included in the boot archive, type:**

```
# bootadm list-archive
```

Managing the Boot Archive SMF Service

The `boot-archive` service is controlled by SMF. The service instance is `svc:/system/boot-archive:default`. The `svcadm` command is used to enable and disable services.

Determining Whether the boot-archive SMF Service Is Running

If the `boot-archive` service is disabled, automatic recovery of the boot archive upon a system reboot might not occur. As a result, the boot archive could become unsynchronized or corrupted, which would prevent the system from booting.

To determine whether the `boot-archive` service is running, use the `svcs` command, as follows:

```
$ svcs boot-archive
STATE          STIME    FMRI
online         10:35:14 svc:/system/boot-archive:default
```

In this example, the output of the `svcs` command indicates that the `boot-archive` service is online.

For more information, see the `svcadm`(1M) and `svcs`(1) man pages.

▼ How to Enable or Disable the boot-archive SMF Service

1. **Become an administrator.**

   For more information, see "How to Obtain Administrative Rights" in *Oracle Solaris Administration: 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 that the service is offline.

Troubleshooting For information about troubleshooting boot archive update failures, see "Maintaining the Integrity of the Boot Archives" on page 78.

Maintaining the Integrity of the Boot Archives

The boot administration interface, `bootadm`, enables you to perform the following tasks for maintaining the Oracle Solaris boot archive:

- List the files and directories that are included in a system's boot archive.
- Manually update the boot archive.

The syntax of the command is as follows:

`bootadm` **`[`***subcommand***`]`** **`[-`***option***`]`** **`[-R`** *altroot***`]`**

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

▼ How to Clear a Failed Automatic Boot Archive Update by Using the auto-reboot-safe Property

Boot archive recovery on x86 platforms is automated through the Fast Reboot feature. However, during the process of booting the system, if a warning similar to the following is displayed:

```
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. As a result, the automatic update of the boot archive fails. To correct the problem, follow the steps in this procedure.

1 **Become the `root` role.**

2. **Reboot the system.**

```
# reboot
```

3. **If the active BIOS boot device and the GRUB menu entries point to the current boot instance, follow these steps to prevent a boot archive update failure:**

   a. **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
```

   b. **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 a Failed Automatic Boot Archive Update by Manually Updating the Boot Archive

During the process of booting the system, if a warning message that is similar to the following is displayed, and as a result, the automatic update of the boot archive fails.

```
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.

Note – The same procedure can be used to manually update the boot archive on an x86 based system.

---

1. **Become the `root` role.**

2. **To update the boot archive, type the following command:**

```
# bootadm update-archive
```

Note – 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.

Caution – 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
```

◆ ◆ ◆ CHAPTER 9

Troubleshooting Booting an x86 Based System (Tasks)

The following are procedures for troubleshooting booting an x86 based system.

The following is a list of the information that is in this chapter:

For information about stopping and starting Oracle Solaris for recovery purposes, if you are running a service processor, as well as instructions on controlling Oracle ILOM service processors, see the hardware documentation at `http://download.oracle.com/docs/cd/E19694-01/E21741-02/index.html`.

For information about how to resolve problems with the Oracle Solaris boot archives, see "Maintaining the Integrity of the Boot Archives" on page 78.

For information about troubleshooting booting on a SPARC based system, see Chapter 9, "Troubleshooting Booting a SPARC Based System (Tasks)," in *Booting and Shutting Down Oracle Solaris on SPARC Platforms*.

Troubleshooting Booting an x86 Based System (Task Map)

TABLE 9–1 Troubleshooting Booting an x86 Based System: Task Map

Task	Description	For Instructions
Stop an x86 based system for recovery purposes.	If a damaged file is preventing an x86 based system from booting, first stop the system to attempt recovery.	"How to Stop a System for Recovery Purposes" on page 83

TABLE 9–1 Troubleshooting Booting an x86 Based System: Task Map *(Continued)*

Task	Description	For Instructions
Boot an x86 based system in single-user mode to resolve a minor booting problem, such as a bad `root` shell or incorrect password entry.	Boot a system in single-user mode to resolve an unknown `root` password or similar problem.	"How to Boot in Single-User Mode to Resolve a Bad `root` Shell or Password Problem" on page 83
Boot an x86 based system from media to resolve an unknown `root` password problem.	Boot a system from media, then import and mount the root pool to correct the problem.	"How to Boot From Media to Resolve an Unknown `root` Password" on page 84
Boot an x86 based system from media to resolve a problem with the `menu.lst` file that prevents the system from booting.	Boot a system from media, then import the root pool to analyze and correct a problem with the `menu.lst` file.	"How to Boot From Media to Resolve a Problem With the `menu.lst` File That Prevents the System From Booting" on page 86
Force a crash dump and reboot of an x86 based system.	Force a crash dump and reboot of an x86 based system as a troubleshooting measure.	"How to Force a Crash Dump and Reboot of the System" on page 87
Boot an x86 based system with the kernel debugger (`kmdb`) enabled.	Boot an x86 based system with the kernel debugger enabled to interact with the kernel and troubleshoot system problems.	"How to Boot a System With the Kernel Debugger Enabled (`kmdb`)" on page 88
Boot an x86 based system for recovery purposes in a ZFS root environment.	Use this procedure, if you need to boot the system, so that you can recover from a lost `root` password or similar problem.	"How to Boot From Media to Resolve a Problem With the `menu.lst` File That Prevents the System From Booting" on page 86
Troubleshoot issues with the Fast Reboot feature on an x86 based system.	Troubleshoot issues that can prevent an x86 based system from initiating a fast reboot.	"Troubleshooting Issues With Fast Reboot on the x86 Platform" on page 89

Shutting Down and Booting an x86 Based System for Recovery Purposes

In the following instances, you must first shut down a system to analyze or troubleshoot booting and other system problems.

- 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.

The procedures that follow describe how to safely shut down and then boot an x86 based system for recovery purposes.

Stopping and Booting a System for Recovery Purposes

You might need to boot the system for recovery purposes.

The following are some of the more common error and recovery scenarios:

- Boot a system in single-user mode to resolve a minor problem, such as correcting the `root` shell entry in the `/etc/passwd` file or changing a NIS server.
- Boot from the installation media or from an install server on the network to recover from a problem that is preventing the system from booting or to recover from a lost `root` password. This method requires you to mount the boot environment after importing the root pool.
- Resolve a boot configuration problem by importing the root pool. If a problem with the `menu.lst` file exists, you do *not* have to mount the boot environment, just import the root pool, which automatically mounts the `rpool` file system that contains the boot-related components.

▼ How to Stop a System for Recovery Purposes

1 Stop the system.

- First, become the `root` role, then type `init 0` if the keyboard and mouse are functional.
- If the `Press any key to reboot` prompt is displayed, press any key to reboot the system.
- To reboot the system, type `init 6`.

2 If the system does not respond to any input from the mouse, do one of the following:

- **Press the Reset key to reboot the system.**
- **Use the power switch to reboot the system.**

▼ How to Boot in Single-User Mode to Resolve a Bad root Shell or Password Problem

1 Stop the system.

```
# init 0
```

2 Reboot the system.

```
# reboot
```

3. **When the GRUB menu is displayed, do the following:**

   a. **Select the appropriate boot entry, then type `e` to edit that entry.**

   b. **Using the arrow keys, select the `kernel$` line.**

      If you cannot use the arrow keys, use the caret (^) key to scroll up and the letter v key to scroll down.

   c. **Type `-s` at the end of the `$kernel` line, then press Return to save your changes and return to the previous screen.**

   d. **Type `b` to boot the system in single-user mode.**

4. **Correct the shell entry in the `/etc/passwd` file.**

   ```
   # vi /etc/password
   ```

5. **Press `control-d` to reboot the system.**

How to Boot From Media to Resolve an Unknown root Password

Use the following procedure if you need to boot the system to correct a problem an unknown `root` password or similar problem. Note that this procedure requires you to mount the boot environment after importing the root pool. If you need to recover a root pool or root pool snapshot, see "How to Replace a Disk in a ZFS Root Pool" in *Oracle Solaris Administration: ZFS File Systems*.

1. **Boot from the Oracle Solaris media.**

   - **Live Media – Boot from the installation media and use a GNOME terminal for the recovery procedure.**

   - **Text installation – From the GRUB menu, select the `Text Installer and command line` boot entry, then select the option `3 Shell` from the text installation screen.**

   - **Automated installation – Booting from an install server on the network requires a PXE boot. Select the `Text Installer and command line` entry from the GRUB menu. Then, select the option `3 Shell` from the text installation screen.**

     For example:

     ```
     1  Install Oracle Solaris
             2  Install Additional Drivers
             3  Shell
             4  Terminal type (currently xterm)
             5  Reboot

     Please enter a number [1]: 3
     To return to the main menu, exit the shell
     ```

2. **Import the root pool.**

   ```
   zpool import -f rpool
   ```

3. **Create a mount point for the boot environment.**

   ```
   # mkdir /a
   ```

4. **Mount the boot environment on `/a`**

   ```
   # beadm mount solaris-instance|bename /a
   ```

   For example:

   ```
   # beadm mount solaris-2 /a
   ```

5. **If a password or shadow entry is preventing a console login, correct the problem.**

   a. **Set the `TERM` type.**

      ```
      # TERM=vt100
      # export TERM
      ```

   b. **Edit the `shadow` file.**

      ```
      # cd /a/etc
      # vi shadow
      # cd /
      ```

6. **Update the boot archive.**

   ```
   # bootadm update-archive /R /a
   ```

7. **Unmount the boot environment.**

   ```
   # beadm umount be-name
   ```

8. **Halt the system.**

   ```
   # halt
   ```

9. **Reboot the system in single-user mode, as described in "How to Boot in Single-User Mode to Resolve a Bad `root` Shell or Password Problem" on page 83, and when prompted for the `root` password, press Return.**

10. **Reset the `root` password.**

    ```
    root@system:~# passwd -r files root
    New Password: xxxxxx
    Re-enter new Password: xxxxxx
    passwd: password successfully changed for root
    ```

11. **Press `control-d` to reboot the system.**

How to Boot From Media to Resolve a Problem With the menu.lst File That Prevents the System From Booting

Use the following procedure if you need to boot the system to correct a problem with the default `menu.lst` file. Note that this procedure does *not* require you to mount the boot environment. If you need to recover a root pool or root pool snapshot, see "How to Replace a Disk in a ZFS Root Pool" in *Oracle Solaris Administration: ZFS File Systems*.

1. **Boot from the Oracle Solaris media.**

   - **Live Media – Boot from the installation media and use a GNOME terminal for the recovery procedure.**
   - **Text installation – From the GRUB menu, select the `Text Installer and command line` boot entry, then select the option `3 Shell` from the text installation screen.**
   - **Automated installation – Booting from an install server on the network requires a PXE boot. Select the `Text Installer and command line` entry from the GRUB menu. Then, select the option `3 Shell` from the text installation screen.**

     For example:

     ```
     1  Install Oracle Solaris
             2  Install Additional Drivers
             3  Shell
             4  Terminal type (currently xterm)
             5  Reboot

     Please enter a number [1]: 3
     To return to the main menu, exit the shell
     ```

2. **Import the root pool.**

   ```
   zpool import -f rpool
   ```

3. **Examine the entries in the `menu.lst` file and make corrections as needed.**

   ```
   # cd /rpool/boot/grub
   # vi menu.lst
   ```

4. **Update the boot archive.**

   ```
   # bootadm update-archive -R /a
   ```

5. **Exit the shell and reboot the system.**

   ```
   exit
           1  Install Oracle Solaris
           2  Install Additional Drivers
           3  Shell
           4  Terminal type (currently sun-color)
           5  Reboot

   Please enter a number [1]: 5
   ```

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 "Managing System Crash Dump Information" in *Oracle Solaris Administration: Common Tasks*.

▼ 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 time 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 force 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 9–1 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 by using the `halt -d` and `boot` commands.

```
# halt -d
4ay 30 15:35:15 wacked.<domain>.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
```

How to Boot a System With the Kernel Debugger Enabled (kmdb)

This procedure shows the basics for loading the kernel debugger (`kmdb`). The `savecore` feature is enabled by default.

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 caret (^) key to scroll up and the letter 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. **In the GRUB edit menu, type `-kmdb` or `-k` at the end of the `kernel$` line.**

   ```
   grub edit> kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS -s -k
   ```

6. **Press enter to go back one screen, then type `b` to boot the system with the kernel debugger enabled.**

   Typing `-kmdb` or `-k` loads the debugger, then directly boots the operating system.

7. **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 that is appropriate for that type of serial console.

To access the kernel debugger before the system fully boots, use the `-kd` option.

Using the `-kd` option loads the debugger and then gives you an opportunity to interact with the debugger before booting the operating system.

A welcome message is displayed when you access the kernel debugger for the first time.

See Also For more detailed information about interacting with the system by using `kmdb` and the execution control facilities that are provided by `kmdb`, see the `kmdb(1)` man page.

Troubleshooting Issues With Fast Reboot on the x86 Platform

The following sections describe how to identify and resolve some common issues that you might encounter with the Fast Reboot feature of Oracle Solaris on x86 platforms.

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" >> /etc/system
# echo "fastreboot_onpanic/W" | mdb -kw
```

Troubleshooting Conditions That Might Prevent Fast Reboot From Working on x86 Platforms

The following are possible conditions under which the Fast Reboot feature might not work:

- The GRUB menu cannot be processed.
- The 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.

- There is insufficient memory.

  If there is not enough memory on the system, 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
  ```

- The environment is unsupported.

  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)

Index

G

H

I

K

L

M

N

O

P

R

S

T

W

X