Booting and Shutting Down Oracle® Solaris 11.3 Systems

Part No: E54742
October 2017

Booting and Shutting Down Oracle Solaris 11.3 Systems

Part No: E54742

Copyright © 1998, 2017, Oracle and/or its affiliates. All rights reserved.

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

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

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

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Access to Oracle Support

Oracle customers that have purchased support 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.

Référence: E54742

Copyright © 1998, 2017, Oracle et/ou ses affiliés. Tous droits réservés.

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

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

Si ce logiciel, ou la documentation qui l'accompagne, est livré sous licence au Gouvernement des Etats-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

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

Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

Accès aux services de support Oracle

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` ou le site `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` si vous êtes malentendant.

Contents

Using This Documentation

- **Overview** – Describes how to boot and shutdown a system
- **Audience** – Technicians, system administrators, and authorized service providers
- **Required knowledge** – Experience administering an Oracle Solaris system

Product Documentation Library

Documentation and resources for this product and related products are available at `http://www.oracle.com/pls/topic/lookup?ctx=E53394-01`.

Feedback

Provide feedback about this documentation at `http://www.oracle.com/goto/docfeedback`.

♦ ♦ ♦ C H A P T E R 1

Overview of Booting and Shutting Down a System

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 Oracle Solaris system. Any information in this chapter that applies only to SPARC or x86 based systems is identified as such.

Note - This book focuses primarily on booting and shutting down a single Oracle Solaris instance on physical machines. 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. For more information, see the product documentation for your specific hardware or configuration at `https://www.oracle.com/products/oracle-a-z.html`.

This chapter contains the following information:

What's New in Booting and Shutting Down a System

In this release, the `bootadm` command has been expanded to secure the GRUB menu. You can now allow only specific users or users with a specific password to view, edit, or boot from the GRUB menu. See "Administering the GRUB Configuration by Using the `bootadm` Command" on page 30 for more information.

On systems that can boot using firmware-inaccessible storage devices, such as an iSCSI device accessed using IP over Infiniband (IPoIB), the Oracle Solaris boot process has the following enhancements:

- Can access boot archives in a boot pool on firmware-accessible devices. The boot archive includes the set of files needed to boot the Oracle Solaris kernel for the boot environment (BE) with which that boot dataset is associated. The boot pool includes boot loader data files and recovery data. Each dataset in the boot pool is linked to a BE. See Chapter 6, "Managing Systems with Boot Pools" for more information.
- Allows the root pool to reside on a firmware-inaccessible storage device that is not accessible from OpenBoot. When you create a root pool, a boot pool is automatically created on OpenBoot-accessible devices. See "Changes to the Boot Process" on page 143 for more information.
- Can boot from a fallback image stored on the service processor (SP) if no devices in the boot pool are accessible from OpenBoot. The fallback image is available on any domain that has access to a service processor and its associated rKVMS services. See "Booting From a Fallback Image" on page 144 for more information.

Guidelines for Booting a System

Bootstrapping is the process of loading and executing the bootable operating system. Typically, the stand-alone program is the operating system kernel, but any stand-alone program can be booted. After the kernel is loaded, it starts the UNIX system, mounts the necessary file systems, and runs `/usr/sbin/init` to bring the system to the `initdefault` state that is specified in the `/etc/inittab` file.

Keep the following guidelines in mind when booting a system:

- After a SPARC basedsystem is shut down, it is booted by using the `boot` command at the PROM level. After a SPARC based system is turned on, the system firmware (in PROM) executes a power-on self-test (POST). The form and scope of these tests depends on the version of firmware in your system. After the tests have successfully completed, the firmware attempts to auto boot, if the appropriate flag has been set in the non-volatile storage area that is used by the firmware. The name of the file to load, and the device to load it from, can also be manipulated.
- An x86 based system is booted by selecting an operating system in the GRUB menu that is displayed at boot time. If no operating system is selected, the system boots the default operating system that is specified in the `grub.cfg` file.
- A system can also be rebooted by turning the power off and then back on.

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

TABLE 1 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"
Change kernel parameters in the `/etc/system` file.	Reboot the system to a multiuser state (run level 3 with NFS resources shared)	"How to Boot a System to a Multiuser State (Run Level 3)" on page 83
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)	"How to Boot a System to a Single-User State (Run Level S)" on page 85
Repair a system configuration file such as `/etc/system`.	Interactive boot	"How to Boot a System Interactively" on page 89
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" in *Managing Devices in Oracle Solaris 11.3*
Boot a system for recovery purposes due to a lost `root` password, or to fix a file system or a similar problem.	Depending on the error condition or problem, you might need to boot the system from media, mount the boot environment, or both.	"Shutting Down and Booting a System for Recovery Purposes" on page 127
x86 only: Recover from a problem with the GRUB configuration.	Recovery boot from media.	"How to Boot From Media to Resolve a Problem With the GRUB Configuration That Prevents the System From Booting" on page 133
Recover from a hung system by forcing a crash dump.	Recovery boot	"How to Force a Crash Dump and Reboot of the System" on page 135
		"How to Force a Crash Dump and Reboot of the System" on page 137
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 (`kmdb`) Enabled" on page 138
		"How to Boot a System With the Kernel Debugger (`kmdb`) Enabled" on page 139

Overview of the Oracle Solaris Boot Architecture

The Oracle Solaris boot architecture includes the following fundamental characteristics:

- **Uses a boot archive.**

  The boot archive contains a file system image that is mounted by using an in-memory disk. The image is self-describing, specifically containing a file system reader in the boot block (or GRUB boot loader in the case of x86 platforms). On SPARC platforms, the file system reader mounts and opens the RAM disk image, then reads and executes the kernel that is contained within it. By default, this kernel is in `/platform/`uname -m'/kernel/unix`. On x86 platforms, the GRUB boot loader loads the kernel file and the boot archive into memory, then transfers control to the kernel. The default kernel on x86 platforms is `/platform/i86pc/kernel/amd64/unix`.

Note - When booting a SPARC based system from disk, the OBP firmware reads the boot blocks from the partition that is specified as the boot device. This stand-alone booter usually contains a file reader that is capable of reading the Oracle Solaris boot archive. See `boot`(1M).

If you are booting from a ZFS root file system, the path names of both the archive and the kernel file are resolved in the root file system (dataset) that is selected for booting.

- **Uses a boot administration interface to maintain the Oracle Solaris boot archives and to manage GRUB configuration and the GRUB menu on x86 platforms.**

  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.

  On x86 based systems, the `grub.cfg` file and the x86 boot loader are administered with the `bootadm` command. In this release, the `bootadm` commands has been modified, and some new subcommands have been added. to enable you to perform most of the administrative tasks that were previously done by editing the `menu.lst` file. These tasks include managing the GRUB menu, setting kernel arguments for a specific boot entry, and managing the boot loader. See "Administering the GRUB Configuration by Using the `bootadm` Command" on page 30 for instructions.

Note - Some `bootadm` command options do not apply to SPARC platforms.

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

- **Uses a ramdisk image as the root file system during installation.**

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

Note - On SPARC platforms, the OpenBoot PROM continues to be used to access a system's boot device and to transfer the boot archive to the system's memory.

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 access frequently needed installation components from removable media. The ramdisk file system type can be a High Sierra File System (HSFS) or UFS.

- **Supports booting from GPT labeled disks.**

  Oracle Solaris includes support for booting from GPT labeled disks. Booting from a GPT labeled disk is slightly different from booting from a disk that uses the MSDOS partitioning scheme. Installing Oracle Solaris 11.3 on an x86 or SPARC based system with GPT-aware firmware applies a GPT disk label on the root pool disk that uses the entire disk in most cases. See "SPARC: GPT Labeled Disk Support" in *Oracle Solaris 11.3 Release Notes* for more information about applying GPT-aware firmware on supported SPARC based systems. Otherwise, installing Oracle Solaris 11.3 on a SPARC based system applies an SMI (VTOC) label to the root pool disk with a single slice 0.

  On x86 platforms, the introduction of GRUB 2 enables this support. On systems with BIOS firmware, the MBR is still the first chunk of code that the firmware loads to initiate the boot process. There is no longer a VTOC on GPT labeled disks, only discrete partitions. GRUB now has direct support for reading and interpreting the GPT partitioning scheme, which enables the boot loader to locate the Oracle Solaris kernel and the boot archive inside the root pool that is hosted in a ZFS GPT partition.

  On systems with UEFI firmware, the key difference is that the firmware loads the boot application from the (FAT-based) EFI System Partition. After GRUB is loaded on a UEFI system, it performs similar tasks to the BIOS-targeted GRUB.

About Oracle Solaris Boot Archives

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.

The `bootadm` command manages the boot archive on both SPARC and x86 platforms, including 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.

The files that are part of the x86 boot archive are located in the `/platform/i86pc/amd64/archive_cache` directory. The files in the SPARC boot archive are located in the `/platform/`

`uname -m`/archive_cache directory. To list the contents of the boot archive on both the SPARC and x86 platforms, use the `bootadm list-archive` command:

```
$ bootadm list-archive
```

If any files in the boot archive are updated, the archive must be rebuilt. The `bootadm update-archive` command enables you to manually rebuild the boot archive. The command can be used either as a preventative measure or as part of a recovery process.

```
# bootadm update-archive
```

For modifications to take effect, the rebuild of the archive must take place before the next system reboot. For more information, see "Managing the Oracle Solaris Boot Archives" on page 123.

Description of the Boot Process

This section describes the basic boot process on the SPARC and x86 platforms. For more information about boot processes on specific hardware types, including systems that have service processors and system that have multiple physical domains, see the product documentation for your specific hardware at `https://www.oracle.com/products/oracle-a-z.html`.

The process of loading and executing a stand-alone program is called *bootstrapping*. Typically, the stand-alone program is the operating system kernel. However, any stand-alone program can be booted instead of the kernel.

On SPARC platforms, the bootstrapping process consists of the following basic phases:

- After you turn on a system, the system firmware (PROM) executes a power-on self-test (POST).
- After the test has been successfully completed, the firmware attempts to autoboot, if the appropriate flag has been set in the non-volatile storage area that is used by the system's firmware.
- The second-level program is either a file system-specific boot block, when you booting from a disk, or `inetboot` or `wanboot`, when you are booting across the network or using the Automated Installer (AI).

On x86 based systems, the bootstrapping process consists of two conceptually distinct phases, kernel loading and kernel initialization. Kernel loading is implemented by GRUB by using the firmware on the system board and firmware extensions in ROMs on peripheral boards. The system firmware loads GRUB. The loading mechanism differs, depending on the type of system firmware that is shipped on the system board.

- After a PC-compatible system is turned on, the system's firmware executes a power-on self (POST), locates and installs firmware extensions from peripheral board ROMS, and then begins the boot process through a firmware-specific mechanism.
- For systems with BIOS firmware, the first physical sector of a hard disk (known as the boot sector) is loaded into memory and its code is executed. Disks that are partitioned with the GUID Partition Table (GPT) must have boot sector code that behaves differently, loading code from another location, because the GPT scheme does not reserve the first sector of each partition for boot sector code storage. In the case where GRUB is running on BIOS firmware, that other location is a dedicated partition, which is known as the *BIOS Boot Partition*. After the GRUB boot sector code loads the rest of GRUB into memory, the boot process continues.

  The boot program then loads the next stage, which in the case of Oracle Solaris, is GRUB itself. Booting from the network involves a different process on systems with BIOS firmware. See Chapter 5, "Booting a System From the Network".
- For systems with UEFI-based firmware, the boot process differs significantly. The UEFI firmware searches for the EFI System Partition (ESP) on disks that it has enumerated and then loads and executes UEFI boot programs according to a UEFI-specification-defined process, which results in a UEFI boot application being loaded into memory and executed. On Oracle Solaris, that UEFI boot application is GRUB. The version of GRUB in this release is built to run as a UEFI boot application. The boot process then continues as it does on systems with BIOS firmware.

For more information about boot processes on specific hardware types, including systems that have service processors and systems that have multiple physical domains, see `https://www.oracle.com/technical-resources/`.

x86: Differences Between UEFI and BIOS Boot Methods

GRUB 2 is capable of booting systems with both BIOS and UEFI firmware, as well as GPT labeled disks. To support boot on UEFI firmware and BIOS firmware, GRUB 2 is built targeting two different platforms: i386-pc (BIOS) and x86_64-efi (64-bit UEFI 2.1+) and is therefore delivered as two discrete sets of binaries.

When booting an x86 based system, note the following differences between UEFI-targeted and BIOS-targeted systems:

- **Command differences** – Certain commands that are used by the BIOS boot method are not available on UEFI firmware. Likewise, certain UEFI commands are not available on systems that support the BIOS boot method.
- **PXE network boot differences** – Changes have been made to the DHCP server configuration to support booting systems with UEFI firmware from the network. These changes include support for the new UEFI client architecture identifier value (DHCP option 93).

Note - Systems that can be configured to boot by using either UEFI firmware or the BIOS boot method will technically work with Oracle Solaris. GRUB is first installed according to the system firmware type at the time of installation (or image-update). While you can run explicit commands to install GRUB in the boot location that is required by the other firmware type, this method is not supported. Systems with a particular firmware type should *not* be reconfigured to boot by using an alternate firmware type after installing Oracle Solaris.

x86: Creating Boot Partitions That Support Systems With UEFI and BIOS Firmware

A new `-B` option has been added to the `zpool create` command. When a whole disk is passed to the `zpool create` create command, the `-B` option causes the `zpool` command to partition the specified device with two partitions: the first partition is a firmware-specific boot partition, and the second partition is the ZFS data partition. This option also is used to create the required boot partition when adding or attaching a whole disk `vdev` to an existing `rpool`, if necessary. The conditions under which the `bootfs` property is allowed have also been modified. Setting the `bootfs` property to identify the bootable dataset on a pool is allowed, *if* all system and disk labeling requirements are met on the pool. As part of the labeling requirement, the required boot partition must also be present. For more information, see "Managing a ZFS Root Pool" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

Service Management Facility and Booting

The Oracle Solaris Service Management Facility (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.

Milestone and Error Level Options for Booting

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.

- You can choose a specific milestone to boot to by using this command on a SPARC based system.

  ```
  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 Investigate Problems Starting Services at System Boot" in *Managing System Services in Oracle Solaris 11.3* 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 "Specifying the Amount of Startup Messaging" in *Managing System Services in Oracle Solaris 11.3* for specific information about the logging levels.
- To boot an x86 based system to a specific milestone or choose the level of logging for `svc.startd`, edit the GRUB menu at boot time to add the `-m` *smf-options* kernel argument to the end of the `$multiboot` line of the specified boot entry. For example:

  ```
  $multiboot /ROOT/s11.3_18/@/$kern $kern -B $zfs_bootfs -m logging-level
  ```

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. 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..

♦ ♦ ♦ CHAPTER 2

x86: Administering the GRand Unified Bootloader

This chapter provides overview and task-related information about the GRand Unified Bootloader (GRUB). GRUB 2, the descendent of the original GRUB 0.97-based boot loader, is the system boot loader on x86 platforms in this release.

Note - The original GRUB (GRUB Legacy) continues to be the default boot loader on x86 platforms that run Oracle Solaris 10 and Oracle Solaris 11 11/11. If you are running an Oracle Solaris release that supports the legacy version of GRUB, see *Booting and Shutting Down Oracle Solaris on x86 Platforms*.

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

- "Introducing GRUB 2" on page 21
- "Administering the GRUB Configuration by Using the `bootadm` Command" on page 30
- "Adding Kernel Arguments by Editing the GRUB Menu at Boot Time" on page 44
- "Customizing the GRUB Configuration" on page 48
- "Upgrading Your GRUB Legacy System to a Release That Supports GRUB 2" on page 49
- "Advanced GRUB Administration and Troubleshooting" on page 54

x86: Introducing GRUB 2

The following information is provided in this section:

- "Description of the GRUB 2 Configuration" on page 23
- "GRUB 2 Partition and Device Naming Scheme" on page 25
- "GRUB 2 and GRUB Legacy Task Comparison" on page 27

GRUB 2 is a powerful and more modular boot loader that supports a wider range of platforms and firmware types, including booting from Unified Extensible Firmware Interface (UEFI) firmware, and booting from GUID Partition Table (GPT) partitioned disks of any size, on systems with BIOS or UEFI firmware. GRUB 2 also supports the UEFI-specified, GPT partitioning scheme.

Like GRUB Legacy, GRUB 2 uses a two-stage boot process. The key difference between GRUB 2 and GRUB Legacy is that GRUB 2 places many facilities in dynamically loaded modules, which enables the core GRUB 2 (second-stage boot loader) image to be smaller and therefore load faster and be more flexible. As a result, GRUB functionality is loaded on demand at boot time.

GRUB 2 introduces the following key changes:

- **Configuration changes**

  The GRUB 2 configuration differs syntactically from the GRUB Legacy configuration. The `menu.lst` file that is used by GRUB Legacy has been replaced by a new configuration file, `grub.cfg`. Unlike the `menu.lst` file, the `grub.cfg` file is automatically regenerated by boot management commands. Therefore, this file should never be directly edited. as any edits are immediately destroyed when the `grub.cfg` file is regenerated. See "Description of the GRUB 2 Configuration" on page 23.
- **Partition and device naming changes**

  Instead of 0-based indexes, GRUB 2 uses 1-based indexes for partitions and a changed device naming scheme. See "GRUB 2 Partition and Device Naming Scheme" on page 25.
- **Boot loader and GRUB menu administration changes**

  You administer the `grub.cfg` file through the `bootadm` command. Modified subcommands and new subcommands enable you to perform most of the administrative tasks that were previously accomplished by editing the `menu.lst` file. Two examples include setting boot attributes (such as kernel arguments) for an Oracle Solaris boot instance and managing boot loader settings. See "Administering the GRUB Configuration by Using the `bootadm` Command" on page 30.
- **GRUB menu and screen changes**

  The various GRUB menus and some tasks, for example, adding kernel arguments by editing the GRUB menu at boot time, work somewhat differently now. These differences are documented in the various tasks within this document, where appropriate.
- **Other boot loader related command changes**

  The `installgrub` command is deprecated in this release. Do not use this command to install the boot loader on systems that support GRUB 2, as doing so can prevent the system from booting. Instead, if you are running a release that supports GRUB 2, use the `bootadm install-bootloader` command. This command supersedes the functionality of the `installgrub` command on x86 platforms and the `installboot` command on

SPARC platforms. See "Installing GRUB 2 by Using the `bootadm install-bootloader` Command" on page 55.

You can use the `installgrub` command to install GRUB Legacy on a system, but only *after* you have verified that the version of GRUB Legacy you are installing supports the ZFS pool version of your root pool, and that there are no remaining GRUB 2 boot environments on the system. For instructions, see "How to Install GRUB Legacy on a System That Has GRUB 2 Installed" on page 56.

x86: Description of the GRUB 2 Configuration

GRUB 2 uses an entirely different configuration than GRUB Legacy. The GRUB Legacy configuration is managed through the `menu.lst` file, but GRUB 2 does not use a `menu.lst` file. Instead, GRUB 2 uses a configuration file, `grub.cfg`, to store the same type of information. Similar to the `menu.lst` file, the `grub.cfg` file is located at the top-level of the ZFS dataset for the root pool, `/`*pool-name*`/boot/grub`, for example, `/rpool/boot/grub/grub.cfg`.

The syntax of the `grub.cfg` file is based on a subset of bash scripting, which is more complex and powerful than the directive-like language that is used in the `menu.lst` file that is shown in the following example:

```
title title
            bootfs pool-name/ROOT/bootenvironment-name
            kernel$ /platform/i86pc/kernel/$ISADIR/unix -B $ZFS-BOOTFS
            module$ /platform/i86pc/$ISADIR/boot_archive
```

By comparison, the same configuration is stored in the `grub.cfg` file, as follows:

```
menuentry "<title>" {
        insmod part_msdos
        insmod part_sunpc
        insmod part_gpt
        insmod zfs
        search --no-floppy --fs-uuid --set=root f3d8ef099730bafa
        zfs-bootfs /ROOT/<BE name>/@/ zfs_bootfs
        set kern=/platform/i86pc/kernel/amd64/unix
        echo -n "Loading ${root}/ROOT/<BE name>/@$kern: "
        $multiboot /ROOT/<BE name>/@/$kern $kern -B $zfs_bootfs
        set
 gfxpayload="1024x768x32;1024x768x16;800x600x16;640x480x16;640x480x15;640x480x32"
        insmod gzio
        echo -n "Loading ${root}/ROOT/<BE name>/@/platform/i86pc/amd64/boot_archive: "
        $module /ROOT/<BE name>/@/platform/i86pc/amd64/boot_archive
}
```

Another significant difference between the `grub.cfg` file and the `menu.lst` file is that you do *not* edit the `grub.cfg` file. The `menu.lst` file supports user-created menu entries and manual changes to global GRUB configuration file settings and variables, in addition to menu entries that are automatically generated by the system. In contrast, the `grub.cfg` file is administered by using various `bootadm` subcommands, many of which are new in this release. The various `bootadm` subcommands enable you to administer most boot loader administration tasks. See "Administering the GRUB Configuration by Using the `bootadm` Command" on page 30.

Another feature of the `grub.cfg` file is that it is autogenerated whenever you make changes to the boot loader configuration. The file is also autogenerated during some operations and when certain boot administration commands are used. If necessary, you can manually generate a new `grub.cfg` file by running the `bootadm` command with the new `generate-menu` subcommand. Use this subcommand to create a new `grub.cfg` file *only* if the boot configuration becomes corrupted. See "How to Manually Regenerate the GRUB Menu" on page 34.

Note - The autogeneration mechanism for the `grub.cfg` file is intended for installed systems *only*.

For reference, the GRUB 2 configuration is stored in the following files:

- `grub.cfg` – Is the main configuration file that is used by GRUB 2.
- /*pool-name*`/boot/grub/menu.conf` – Is a file that is used by Oracle Solaris to generate the final `grub.cfg` configuration file.

  The `menu.conf` file is a separate GRUB meta configuration file that stores the machine-parsable representation of the GRUB 2 configuration.

  Note - Do *not* attempt to edit this file.

- /*pool-name*`/boot/grub/custom.cfg` – Is an editable file that resides in the same location as the `grub.cfg` and `menu.conf` files. The `custom.cfg` file is created by the administrator (and not on the system by default). This file is reserved for the purpose of adding more complex constructs (menu entries or other scripting information) to the basic GRUB configuration.

  The `custom.cfg` file is referenced in the `grub.cfg` file. If a `custom.cfg` file exists on the system, the commands or directives that are in that file are then processed by the `grub.cfg` file. For more information, see "Customizing the GRUB Configuration" on page 48.

x86: GRUB 2 Partition and Device Naming Scheme

If you are familiar with how device naming for GRUB Legacy works, you need to be aware of the differences between the GRUB Legacy naming scheme and the GRUB 2 naming scheme. While GRUB Legacy implements a 0-based naming scheme for partition indexes, GRUB 2 uses a 1-based naming scheme for partition indexes.

The GRUB 2 device naming scheme uses the following format:

(`hd`*X*, *part-typeY*, *part-typeZ*, ...)

Because partition schemes can be nested, GRUB's device naming scheme has been changed to support arbitrary nesting levels. GRUB accepts either the old-style device naming ("`(hd0,1)`") or the new-style device naming that includes the partition scheme name. For example:

(`hd0`, `gpt1`)

The previous example refers to the first GPT partition on the first disk.

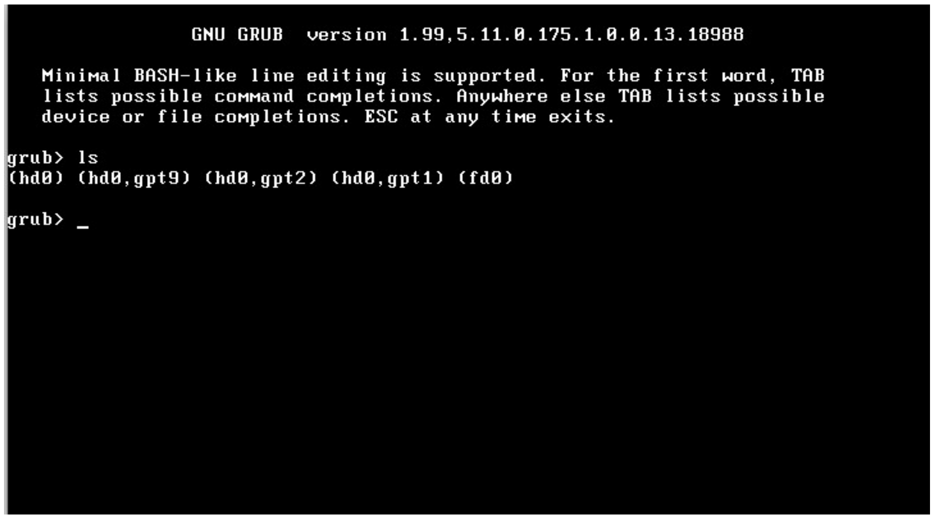
Note - Only the GRUB partition numbering has changed, *not* disk numbering. Disk numbers remain 0-based.

Because GRUB 2 relies on file system UUIDs (or labels) and a built-in search command for automatically locating the proper device or partition name, you are *not* required to manually specify device names. The following table provides examples of the partition indexes and device names that GRUB uses.

TABLE 2 GRUB 2 Partition and Device Naming Scheme

Device Name	Description	Notes
`(hd0, msdos1)`	Specifies the first DOS partition on the first disk.	
`(hd0, gpt2)`	Specifies the second GPT partition on the disk.	This is an example of the prototypical partition where the current release would be installed.
`(hd0, msdos1, sunpc1)`	Specifies the first VTOC slice in the Oracle Solaris partition that is stored in the first DOS partition on the first disk.	This is an example of the prototypical partition where versions of Oracle Solaris prior to this release would be installed.

If you need to determine which partition number refers to a partition that interests you, access the GRUB command-line interpreter by pressing the **C** key (or Control-C, if you are editing a menu entry). Then, run the `ls` command to list all of the partitions that GRUB can identify, which is similar to the following figure.



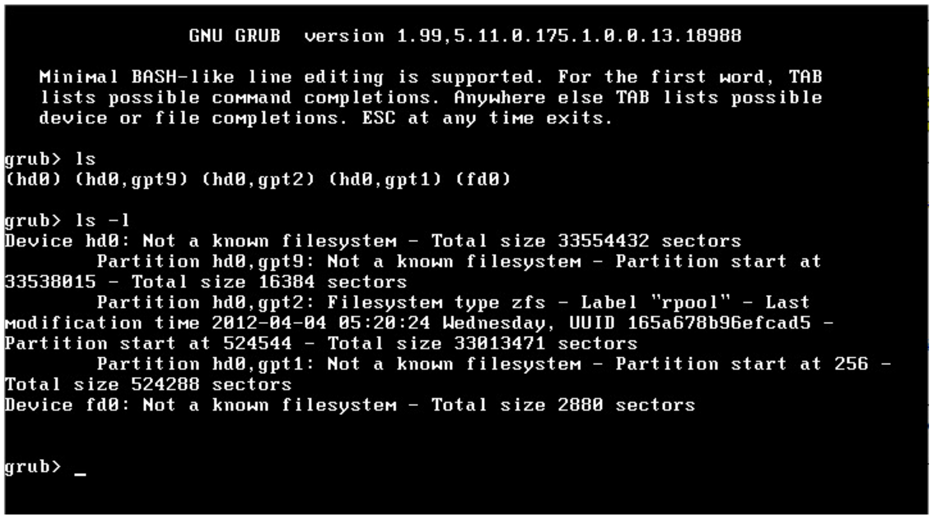
```
                GNU GRUB  version 1.99,5.11.0.175.1.0.0.13.18988

   Minimal BASH-like line editing is supported. For the first word, TAB
   lists possible command completions. Anywhere else TAB lists possible
   device or file completions. ESC at any time exits.

grub> ls
(hd0) (hd0,gpt9) (hd0,gpt2) (hd0,gpt1) (fd0)

grub> _
```

The `-l` option to the `ls` command displays more detailed information about each partition, including file system and file system UUID information, which is similar to the following figure.



```
                GNU GRUB  version 1.99,5.11.0.175.1.0.0.13.18988

   Minimal BASH-like line editing is supported. For the first word, TAB
   lists possible command completions. Anywhere else TAB lists possible
   device or file completions. ESC at any time exits.

grub> ls
(hd0) (hd0,gpt9) (hd0,gpt2) (hd0,gpt1) (fd0)

grub> ls -l
Device hd0: Not a known filesystem - Total size 33554432 sectors
        Partition hd0,gpt9: Not a known filesystem - Partition start at
33538015 - Total size 16384 sectors
        Partition hd0,gpt2: Filesystem type zfs - Label "rpool" - Last
modification time 2012-04-04 05:20:24 Wednesday, UUID 165a678b96efcad5 -
Partition start at 524544 - Total size 33013471 sectors
        Partition hd0,gpt1: Not a known filesystem - Partition start at 256 -
Total size 524288 sectors
Device fd0: Not a known filesystem - Total size 2880 sectors


grub> _
```

Note - GRUB counts the drive numbers from zero, regardless of their type and does not distinguish between Integrated Drive Electronics (IDE) and Small Computer Systems Interface (SCSI) devices.

x86: GRUB 2 and GRUB Legacy Task Comparison

Although GRUB 2 shares several characteristics with GRUB Legacy, because GRUB 2 does not use a `menu.lst` file, many boot administration tasks are performed differently on systems that support GRUB 2. For example, you manage the GRUB menu and perform various boot loader administrative tasks by using new subcommands of the `bootadm` command.

A new `-P` *pool* argument is available for most of the `bootadm` subcommands. This option enables you to view or make changes to the GRUB menu and boot entries for a particular root pool. If you are running an operating system that supports GRUB Legacy, the `-P` option might *not* be available for these `bootadm` subcommands.

For example, you would list the GRUB menu for a specific root pool, as follows:

```
# bootadm list-menu -P pool-name
```

The following table compares some common GRUB 2 tasks and commands to the GRUB Legacy equivalent. For detailed instructions, see the `bootadm`(1M) man page and "Administering the GRUB Configuration by Using the `bootadm` Command" on page 30.

TABLE 3 GRUB 2 Tasks Compared to GRUB Legacy Tasks

Task or Command	GRUB 2 Method	GRUB Legacy Method

List the current boot entries in the GRUB menu.	`bootadm list-menu`
	
	You can also view individual entries by the entry number *or* by title. To view an entry by title:
	
	`bootadm list-menu` *entry-title*
	
	If the title has spaces, quotation marks must be used to protect the title from being parsed as multiple arguments. For example:
	
	`bootadm list-menu `This is a menu entry with a title'`
	
	To view an entry by its entry number:
		`bootadm list-menu`

Task or Command	**GRUB 2 Method**	**GRUB Legacy Method**
	`bootadm list-menu -i` *entry-number*	
Generate a new GRUB configuration file (`grub.cfg`) that contains the default boot loader settings and one menu entry for each Oracle Solaris boot environment on each root pool on the system.	`bootadm generate-menu` If there is an existing `grub.cfg` file on the system, use the `-f` option with the `generate-menu` subcommand. This syntax destroys the existing GRUB 2 configuration and replaces it with the new configuration. If you use the `-P` option to generate a new GRUB 2 configuration file for a specific root pool on the system, note that the `grub.cfg` file that is generated is stored in the top-level ZFS dataset for that root pool.	Manually edit the `menu.lst` file to add the new information.
Add a new entry to the GRUB menu.	To add an entry by specifying its entry number: `bootadm add-entry -i` *entry-number* To add an entry by specifying its title: `bootadm add-entry` *entry-title*	Manually add the entry to the `menu.lst` file.
Change an entry in the GRUB menu.	To change an entry by specifying its entry number: `bootadm change-entry -i` *entry-numberkey=value* To change an entry by specifying its title: `bootadm change-entry` *entry-title key=value* If the title has spaces, quotation marks must be used to protect the title from being parsed as multiple arguments. This subcommand is used to make changes to an individual boot entry, for example, to specify the Oracle Solaris console device as a kernel argument. If the entry title matches multiple menu entries, only the first entry is modified. A boot entry can also be changed by editing the GRUB menu at boot	Manually edit the `menu.lst` file to make persistent changes. As an alternative, edit the GRUB menu at boot time to make changes to the boot entry that persist just until the next time the system is booted.

Task or Command	GRUB 2 Method	GRUB Legacy Method
	time, just as was done in previous releases with a GRUB Legacy entry.	
Remove an entry from the GRUB menu.	To remove an entry by specifying its entry number: `bootadm remove-entry -i` *entry-number* To remove an entry by specifying its title: `bootadm remove-entry` *entry-title* If a title is specified, all of the entries with that title are removed.	Manually remove the entry from the `menu.lst` file.
Manage the GRUB menu. For example, set the default GRUB menu entry from which to boot.	`bootadm set-menu` *key=value*	`bootadm set-menu`
Add custom menu entries to the GRUB menu, for example, a Linux entry.	Add the entry to the `custom.cfg` file, making sure to use the proper GRUB 2 configuration file syntax. See "Customizing the GRUB Configuration" on page 48. **Note -** You must create this file first.	Add the information to the `menu.lst` file after installing Oracle Solaris.
Edit the GRUB menu at boot time to add boot arguments.	1. Interrupt the boot process by using the arrow keys to select the desired menu entry, then type `e`. 2. Add boot arguments to the end of the `$multiboot` line for the specified boot entry. 3. Press Control-X to boot from the modified entry. If the system console is on a serial device, F10 might not be properly recognized on a UEFI system. In that case, use Control-X. **Note -** Pressing the Escape key while editing a menu entry returns you to the menu entry list, and all edits are discarded.	1. Interrupt the boot process by typing `e`. 2. Add the boot arguments to the end of the `kernel$` line for the specified boot entry. 3. Press Return, then type `b` to boot the system.
Install the boot loader program.	`bootadm install-bootloader` This command installs the boot loader on *all* of the devices in a mirrored root pool automatically.	`installgrub` for x86 based systems and `installboot` for SPARC based systems.
Create boot partitions for either UEFI or BIOS firmware.	Use the new `-B` option of the `zpool create` command to automatically create the firmware-appropriate boot partition, and the ZFS data partition	GRUB Legacy supports systems with BIOS firmware *only* and therefore does not require a separate boot partition.

Task or Command	GRUB 2 Method	GRUB Legacy Method
	into which the new ZFS pool will be stored, at the same time. Attaching a disk to a root pool automatically creates the proper boot partitions and installs the boot loader on that disk. See Chapter 6, "Managing the ZFS Root Pool" in *Managing ZFS File Systems in Oracle Solaris 11.3*.	

x86: Administering the GRUB Configuration by Using the `bootadm` Command

This section covers the following topics:

- "Password-Protecting the GRUB Menu" on page 31
- "Granting Users Authorizations to Access the GRUB Menu" on page 32
- "Displaying the GRUB Menu or Specific Menu Entries" on page 33
- "How to Manually Regenerate the GRUB Menu" on page 34
- "How to Change the GRUB Menu" on page 35
- "How to Set Attributes for a Specified Boot Entry in the GRUB Menu" on page 37
- "How to Add a Boot Entry to the GRUB Menu" on page 41
- "How to Remove a Boot Entry From the GRUB Menu" on page 43

On systems that support GRUB Legacy, you primarily manage the GRUB configuration and the GRUB menu by editing the `menu.lst` file. Systems that support GRUB 2 use the `grub.cfg` file. However, instead of manually editing this file, you use the boot administration interface, `bootadm`. You can use the `bootadm` command to administer most of the tasks that you previously did by editing the `menu.lst` file. These tasks include administering boot loader settings and the GRUB menu, as well as the individual attributes of a particular boot entry.

Note - Because changes made to the boot loader by using either the `bootadm` command or the `beadm` command can overwrite the `grub.cfg` file , this file should *never* be directly edited.

The following `bootadm` subcommands support the administration of the GRUB 2 configuration:

`add-entry`	Adds a boot entry to the GRUB menu.

`change-entry`	Changes the attributes of a specified boot entry in the GRUB menu.
`generate-menu`	Generates a new boot loader configuration file.
`install-bootloader`	Installs the system boot loader. This subcommand applies to both x86 and SPARC platforms.
`list-menu`	Displays the current boot entries in the GRUB menu. The `-P` option supports displaying boot entries for a specified root pool. The `-i` option displays information about a specific menu entry identified by an index number. The `-t` option selects the menu entry by the title.
`remove-entry`	Removes a boot entry from the GRUB menu.
`set-menu`	Maintains the GRUB menu. You can use this subcommand to set a particular GRUB menu entry as the default, to add security protection to the GRUB menu, and to set other menu options and boot loader options. The `-P` option supports changing menus on multiple root pools.
`set-menu-password`	Sets a password to prevent the GRUB menu from being seen.
`show-entry`	Shows a boot entry from the GRUB menu. This subcommand is equivalent to `list-menu`.

Note - Because SPARC platforms do not use GRUB, tno boot menu management by using the `bootadm` command. However, you can use the `bootadm` command on SPARC based systems to list the contents of the boot archive, to manually update the boot archive, and to install the boot loader. See "Managing the Oracle Solaris Boot Archives" on page 123.

The following procedures describe how to use the `bootadm` command to manage the GRUB configuration and the GRUB menu. For more complete information, see the `bootadm`(1M) man page.

x86: Password-Protecting the GRUB Menu

In the past, the GRUB menu has been open to anyone who has physical access to the console. Changes in the Oracle Solaris 11.3 release, have added the ability to lock the GRUB menu so that no one can see it without entering a password.

You can control access to the whole GRUB menu by setting a password lock. Any user who has access to the console and knows the password will be able to view entries in the GRUB menu. The options to the `bootadm set-menu-password` command that manage the password lock are:

`-s`	Sets a password needed to view, edit, or boot any entry in the GRUB menu
`-r`	Removes the password needed to access the GRUB menu
`-l`	Lists whether a password lock is in place and which users have access to each menu entry

In addition, you can give individual users the ability to view, edit, or boot all entries in the GRUB menu, or specify entries accessible for each user. For more information, see "Granting Users Authorizations to Access the GRUB Menu" on page 32.

Caution - If you place a password lock on the whole GRUB menu, someone must enter the password before the system will boot.

x86: Granting Users Authorizations to Access the GRUB Menu

You can give specific users authorizations to access the whole GRUB menu, as well as to access specific entries in the GRUB menu. In either situation, you will need to add the user to the list of authenticated users using the `bootadm set-menu adduser=`*username* command. The password used to authenticate access to the GRUB menu is not the same password used by the OS when it is booted.

Once you have added the name to the list of authorized users, you can:

- Add the user to the list of GRUB menu superusers, which will give the user access to all entries
- Define a specific entry to which the user has access

To give individual users the ability to view or edit all entries in the GRUB, add the username to the list of GRUB menu superusers by using the `bootadm set-menu add-superuser` *username* command. Once the username and password are entered at the prompt in the GRUB menu, the user will be able to view, edit or boot all of the entries in the menu.

To give users authorizations to boot and edit specific entries in the GRUB menu, use the `change-entry` subcommand. to select which entries a user can access by either the index number or title. For example, the `bootadm set-menu change-entry -i 3 add-auth=`*username* command gives the named user the ability to edit the index entry `3` menu item.

Caution - If the default boot entry is locked, someone will have to enter the password before the system will boot. If it is important that the system can reboot without manual intervention, make sure that the default entry is not password locked.

x86: Displaying the GRUB Menu or Specific Menu Entries

Use the `list-menu` subcommand of the `bootadm` command to list the GRUB menu entries that are currently on the system. This information is supplied by the `grub.cfg` file. Use either the `list-menu` or `show-entry` subcommand to show information about specific menu entries. You can select specific entries by the entry number using the `-i` option or by title using the `-t` option.

EXAMPLE 1 Listing All Items in the GRUB Menu

You can use the `list-menu` subcommand to see the contents of the current GRUB menu. The subcommand also displays the location of the boot loader configuration files, the default boot entry number, the `autoboot-timeout` value, and the index number and title of each boot entry.

```
$ bootadm list-menu
the location of the boot loader configuration files is: /rpool/boot/grub
default 0
console graphics
timeout 30
0 Oracle Solaris 11 FCS
1 Oracle Solaris backup-1
2 Oracle Solaris 11 11.2
```

EXAMPLE 2 Listing Information About a Specific Entry Number

If you specify an entry number with the `-i` option when running the `list-menu` subcommand, the output displays the information about the selected entry.

```
$ bootadm list-menu -i 0
     the location of the boot loader configuration files is: /rpool/boot/grub
     title: Oracle Solaris 11 FCS
     kernel: /platform/i86pc/kernel/$ISADIR/unix
     kernel arguments: -B $ZFS-BOOTFS -v
     boot archive: /platform/i86pc/$ISADIR/boot_archive
     ZFS root pool: rpool
```

The same information, without the location of the boot loader configuration files is displayed if you use the `show-entry` subcommand with the `-i` option.

```
$ bootadm show-entry -i 0
     title: Oracle Solaris 11 FCS
     kernel: /platform/i86pc/kernel/$ISADIR/unix
     kernel arguments: -B $ZFS-BOOTFS -v
     boot archive: /platform/i86pc/$ISADIR/boot_archive
     ZFS root pool: rpool
```

▼ x86: How to Manually Regenerate the GRUB Menu

Use the `bootadm generate-menu` command to manually regenerate a `grub.cfg` file that contains the OS instances that are currently installed on a system.

Information from the `/usr/lib/grub2/bios/etc/default/grub` or the `/usr/lib/grub2/uefi64/etc/default/grub` file, combined with information from GRUB meta configuration file, `rpool/boot/grub/menu.conf`, is used to generate the final `grub.cfg` file.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Generate the `grub.cfg` file.**

   ```
   # bootadm generate-menu
   ```

   - **If the `grub.cfg` file already exists, use the `-f` option to overwrite the existing file.**

     ```
     # bootadm generate-menu -f
     ```

   - **Generate a new GRUB menu for a root pool other than the current root pool, as follows:**

     ```
     # bootadm generate-menu -P pool-name
     ```

3. **Verify that the menu has been updated to reflect the changes.**

   ```
   # bootadm list-menu
   ```

Note - If you do not see your changes, check the `grub.cfg` file to verify that the change was made.

▼ x86: How to Change the GRUB Menu

Use the `set-menu` subcommand of the `bootadm` command to maintain the GRUB menu. For example, you can use the command to change the menu timeout and the default boot entry in the GRUB menu.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **(Optional) List the GRUB menu entries.**

   ```
   # bootadm list-menu
   ```

3. **Make the necessary changes to the GRUB menu.**

   ```
   # bootadm set-menu [-P pool] [-R altroot [-p platform]] key=value
   ```

   For more information about each value that can you can specify by using the `set-menu` subcommand, see the `bootadm`(1M) man page. Examples of common ways that you can use the `set-menu` subcommand follow this procedure.

4. **Verify that the changes have been made.**

   ```
   # bootadm list-menu
   ```

Note - If you do not see your changes, check the `grub.cfg` file to verify that the change was made.

Example 3 Changing the Default Boot Entry in the GRUB Menu

Use the `bootadm set-menu` command with the appropriate *key=value* option to set the default entry number (for example, 0, 1, or 2) in the GRUB menu. This number designates which operating system is booted when the timer expires.

The following example sets the default boot entry as `2`, which is `Oracle Solaris 11.3`:

```
# bootadm set-menu default=1
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 1
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11_test
```

In this example, the default menu entry is now `1`. When the system is rebooted, it will automatically boot the new Oracle Solaris entry after the default timer expires.

You can also set the default entry in the GRUB menu by using the `change-entry` subcommand. See "How to Set Attributes for a Specified Boot Entry in the GRUB Menu" on page 37.

Example 4 Changing the Menu Timeout Value in the GRUB Menu

Use the `bootadm set-menu` command with the appropriate *key=value* option to set the menu timeout value.

In the following example, the output of the `bootadm list-menu` command shows a default timeout value of 30 seconds that has been changed to 45 seconds. The change takes effect the next time the system is booted.

```
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11_test
# bootadm set-menu timeout=45
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11_test
```

Example 5 Setting the GRUB Console Type

One value that you can set by using the `set-menu` subcommand of the `bootadm` command is the console type. Changing the console type in this way persists over system reboots.

For example, you would set the console type to `serial` in the `grub.cfg` file, as follows:

```
# bootadm set-menu console=serial
```

You can also set the console type to `text` for plain text console. Choose this option if you are using BIOS serial redirection. Or, you can set the console type to `graphics`. This option provides a more graphical menu, and a background image is used.

When you set the console type to `serial`, you can configure the serial parameters that GRUB 2 uses when initializing the serial port at boot time. If you do not specify a `serial_params` value, the default is to use serial port 0 (`COM1`/`ttya`) and to *not* specify a speed. Note that if a speed is not specified and *only* a port is specified, for example, `serial_params=0`, then the speed that is used is undefined and will be whatever speed the serial port was initialized to before GRUB executes. If you to ensure a specific speed is used, they need to explicitly set it with `serial_params`.

Add the `serial_params` key value to the `bootadm` command line, as follows:

port	Is the port number. Any number from `0` to `3` (usually `0` is used for `ttya` or `COM1`) can be used to specify ports `ttya` through `ttyd`, or `COM1` through `COM4`, respectively.
speed	Is the speed that the serial port uses. If this value is omitted, GRUB 2 uses whatever speed the serial port has been initialized to use. If the serial port has not been initialized, failure to specify the speed might cause unpredictable output. If you are not sure if the serial port has been initialized, and you are not using BIOS console redirection, it is best to specify a speed value.
data bits	Is specified with a value of either `7` or `8`.
parity	Is specified as `e`, `o`, `n` (for even, odd, or none), respectively.
stop bits	Is specified with a value of `0` or `1`.

All of the serial parameters, with the exception of the `port` parameter, are optional.

▼ x86: How to Set Attributes for a Specified Boot Entry in the GRUB Menu

Use the `change-entry` subcommand of the `bootadm` command to set certain boot attributes for a specified boot entry, or a comma-separated list of entries, in the GRUB menu. The entry is specified by either an entry title or an entry number. If multiple entries have the same title, all of the entries are affected.

Note - A special property, `set-default`, sets the default entry to boot from when the timer expires. This subcommand functions the same as the `set-menu default=`*value* subcommand. See Example 3, "Changing the Default Boot Entry in the GRUB Menu," on page 35.

For information about how to set attributes for specific boot entries by editing the GRUB menu at boot time, see "Adding Kernel Arguments by Editing the GRUB Menu at Boot Time" on page 44.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **(Optional) List the GRUB menu entries.**

   ```
   # bootadm list-menu
   ```

3. **Set the boot attributes for the specified entry.**

   ```
   # bootadm change-entry [-P pool] {[entry-title[,entry-title...]}]
    | -i entry-number[,entry-number]...]} { key=value [ key=value ...]
    | set-default }
   ```

   When specifying a value that includes white space, you must enclose the value in single or double quotation marks.

   For more information about each value that you can specify by using the `change-entry` subcommand, see the `bootadm`(1M) man page. Examples of common ways to use the `change-entry` subcommand follow this procedure.

4. **Verify that the changes have been made to the specified entry.**

   ```
   # bootadm list-menu
   ```

Note - If you do not see your changes, check the `grub.cfg` file to verify that the change was made.

Example 6 Setting the Title for a Specified Boot Entry in the GRUB Menu

You can set the title for a specified boot entry by using the `change-entry` subcommand of the `bootadm` command. When setting the title, you can specify either the entry number or the entry title. The following example shows how to set the title for a specified boot entry both ways. If multiple entries have the same title, all of the entries are affected.

Set the title for a boot entry by specifying the entry number, as follows:

```
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 1
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11_test
# bootadm change-entry -i 2 title="Oracle Solaris 11-backup1"
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11.32 Oracle Solaris 11-backup1
```

Set the title for a boot entry by specifying the title, as follows:

```
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 1
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.32 Oracle Solaris 11_test
# bootadm change-entry "Oracle Solaris 11_test" title="Oracle Solaris 11-backup1"
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11.32 Oracle Solaris 11-backup1
```

Example 7 Changing a Boot Entry by Specifying Kernel Arguments

The following examples show how to set kernel boot arguments for a specified boot entry by using the `change-entry` subcommand of the `bootadm` command.

In this example, boot entry number 1 is set to boot in single-user mode:

```
# bootadm list-menu
```

```
The location of the boot loader configuration file is /rpool/boot/grub
default 1
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11_test
# bootadm change-entry -i 1 kargs=-s
# bootadm list-menu -i 1
The location of the boot loader configuration files is: /rpool/boot/grub
     title: Oracle Solaris 11.3
     kernel: /platform/i86pc/kernel/$ISADIR/unix
     kernel arguments: -s
     boot archive: /platform/i86pc/$ISADIR/boot_archive
     ZFS root pool: rpool
```

In this example, multiple kernel arguments are specified for boot entry number 2:

```
# bootadm change-entry -i 2 kargs="-v -s"
# bootadm list-menu -i 2
The location of the boot loader configuration files is: /rpool/boot/grub
     title: Oracle Solaris 11_test
     kernel: /platform/i86pc/kernel/$ISADIR/unix
     kernel arguments: -v -s
     boot archive: /platform/i86pc/$ISADIR/boot_archive
     bootfs: rpool/ROOT/snv_160-nightly-1
```

In this example, the `-v` and `-s` options were specified, which boots the system to a single-user state in verbose mode.

Any time that you set an attribute (or multiple attributes) that include white space, you must enclose the values in single or double quotation marks.

Example 8 Changing a Boot Entry by Using the `-B` Option to Specify Kernel Arguments

The following examples show some of ways that you can set kernel arguments for a specific boot entry by using the `-B` option.

You would disable the `e1000g` network driver and load the kernel debugger at boot time, as follows:

```
# bootadm change-entry -i 0 kargs="-B disable-e1000g=true -k"
```

You can specify multiple `-B` options by using the `bootadm change-entry` command. For example, you would disable the `e1000g` driver *and* ACPI at the same time by using either of the following commands:

```
# bootadm change-entry -i 0 kargs="-B disable-e1000g=true -B acpi-user-options=2"

# bootadm change-entry -i 0 kargs="-B disable-e1000g=true,acpi-user-options=2"
```

You can also use the `-B` option to set certain boot attributes at boot time by editing the specified boot entry. For instructions, see "Adding Kernel Arguments by Editing the GRUB Menu at Boot Time" on page 44.

Example 9 Removing Previously Added Kernel Arguments From a Boot Entry

In the following example, a kernel argument (`-s`) is removed from a specific boot entry:

```
# bootadm list-menu -i 1
the location of the boot loader configuration files is: /rpool/boot/grub
title: s11.3.backup
kernel: /platform/i86pc/kernel/amd64/unix
kernel arguments: -s
boot archive: /platform/i86pc/amd64/boot_archive
bootfs: rpool/ROOT/s11.3.backup
# bootadm change-entry -i 1 kargs=
# bootadm list-menu -i 1
the location of the boot loader configuration files is: /rpool/boot/grub
title: s11.3.backup
kernel: /platform/i86pc/kernel/amd64/unix
kernel arguments:
boot archive: /platform/i86pc/amd64/boot_archive
bootfs: rpool/ROOT/s11.3.backup
```

▼ x86: How to Add a Boot Entry to the GRUB Menu

Use the `add-entry` subcommand of the `bootadm` command to add a new entry to the GRUB menu with the specified entry title. If you specify an entry number, the new entry is inserted at the given position in the GRUB menu. Or, if the entry number is higher than the current number of entries in the menu, the entry is then added as the last entry in the menu.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **(Optional) List the current boot entries in the GRUB menu.**

   ```
   # bootadm list-menu
   ```

3. **Add the new boot entry to the GRUB menu.**

   # bootadm add-entry -P *pool* **-i [***entry-number***]** *entry-title*

4. **Set the `bootfs` property for the newly added entry as follows:**

   # bootadm change-entry -i *new-entry-number* **bootfs='***pool-name***/ROOT/***be-name***'**

   This step ensures that the newly added boot entry does not use the default `bootfs` value that is set in the root pool, which is specified in the `bootfs` *pool-level* property.

5. **Verify that the boot entry was added.**

   ```
   # bootadm list-menu
   ```

   Note - If you do not see your changes, check the `grub.cfg` file to verify that the change was made.

Example 10 x86: Adding a Boot Entry to the GRUB Menu

The following example shows how to add a menu entry to the GRUB menu by using the `bootadm add-entry` command. In this example, entry number 2 is added.

```
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
# bootadm add-entry -i 2  Oracle Solaris 11_test
# bootadm change-entry -i 2 bootfs='rpool/ROOT/test'
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11_test
```

View the contents of the new menu entry by specifying the entry number, as follows:

```
# bootadm list-menu -i 2
     the location of the boot loader configuration files is: /rpool/boot/grub
     title: Oracle Solaris 11_test
```

```
    kernel: /platform/i86pc/kernel/amd64/unix
    kernel arguments: -B $ZFS-BOOTFS
    boot archive: /platform/i86pc/amd64/boot_archive
    ZFS root pool: rpool
```

▼ x86: How to Remove a Boot Entry From the GRUB Menu

Use the `remove-entry` subcommand of the `bootadm` command to remove a given entry, or a comma-separated list of entries, from the GRUB menu. If you specify multiple entries with the same title, all of the entries with that title are removed.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **(Optional) List the current boot entries.**

   ```
   # bootadm list-menu
   ```

3. **Remove the specified entry from the GRUB menu.**

   ```
   # bootadm remove-entry [-P pool] [{entry-title [,entry-title...] |
    -i entry-number[,entry-number...]}
   ```

4. **Verify that the entry has been removed.**

   ```
   # bootadm list-menu
   ```

Note - If you do not see your changes, check the `grub.cfg` file to verify that the change was made.

Example 11 x86: Removing a Boot Entry From the GRUB Menu

The following example shows the removal of entry number 2 from the GRUB menu.

```
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 30
```

```
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11_test
bootadm remove-entry -i 2
1 entry removed
# bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
```

x86: Adding Kernel Arguments by Editing the GRUB Menu at Boot Time

On x86 platforms, you can set boot attributes and kernel arguments for a specific boot entry by editing the GRUB menu at boot time. These changes persist until the next time the system is booted.

To permanently set boot attributes for a specific boot entry, use the `bootadm` command with the `change-entry` subcommand. See "How to Set Attributes for a Specified Boot Entry in the GRUB Menu" on page 37.

When you boot an x86 based system the GRUB main menu is displayed. This menu contains a list of all of the boot entries that are currently on the system. To edit a specific boot entry, use the arrow keys to select the entry, then type `e` to edit the entry. In the GRUB edit screen, navigate to the `$multiboot` line, then type the additional boot option or kernel argument at the end of the line.

The `$multiboot` line in the GRUB edit menu might look similar to the following:

```
$multiboot /ROOT/transition/@/$kern $kern -B console=graphics -B $zfs_bootfs
```

For example, to disable the `e1000g` network driver and load `kmdb` at boot time, you would edit the GRUB menu for the specified entry, as follows:

```
$multiboot /ROOT/solaris/@/$kern $kern -B disable-e1000g=true -k -B $zfs_bootfs
```

To exit the GRUB edit menu and boot the entry you just edited, press Control-X. If you have a system with UEFI firmware, and you are not using a serial console, pressing F10 also boots the entry.

Note - If you plan to edit the GRUB menu at boot time, you must reboot the system by using the `-p` option of the `reboot` command to ensure that the GRUB menu is displayed during the boot sequence.

The following kernel arguments and options can be specified when you edit the GRUB menu at boot time:

`unix`	Specifies the kernel to boot.
`-a`	Prompts the user for configuration information.
`-i altinit`	Specifies an alternative executable as the primordial process. `altinit` is a valid path to an executable.
`-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.
`-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.
`-s`	Boots the system to a single-user state.
`-v`	Boots the system with verbose messages enabled.

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

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

x86: Adding `-B` *prop=val* Kernel Arguments at Boot Time by Editing the GRUB Menu

You can specify certain kernel arguments at boot time, for example, setting the Oracle Solaris system console, by specifying `-B` *prop=val* options. The following are the various ways in which you can modify boot parameters on x86 platforms at boot time by adding `-B` *prop=val* options to a specified boot entry:

`-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 "Redirecting the Oracle Solaris Console at Boot Time" on page 47.
`-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.
`-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.
`-B uefirt_disable=1`	Disables the use of UEFI runtime services in Oracle Solaris.

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

EXAMPLE 12 Configuring Text Mode Boot Parameters for the Oracle Solaris System 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, `console=force-text`, directs the system to not use a VGA adapter as a bitmap device and sets the adapter to VGA text mode.

Note that setting the `console=force-text` property for the console *will not* transition the VGA adapter to text mode on systems with UEFI firmware.

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 13 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
```

Redirecting the Oracle Solaris Console at Boot Time

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 UEFI firmware and traditional BIOS firmware with 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.

Oracle Solaris boot entries will attempt a specific set of graphics modes in a particular order. These modes are listed in the `set gfxpayload` line that follows the `$multiboot` line in the `grub.cfg` file. You can alter this line if you desire a mode that is not listed. To make this change persistent, you must copy the entry to the `custom.cfg` file. Otherwise, the next time the `grub.cfg` file is autogenerated, the `gfxpayload` setting is overwritten.

The syntax for the `set gfxpayload` argument is as follows:

Width`x`*Height*`[x`*bit-depth*`]`

The "x" is the actual character, for example:

```
set gfxpayload=1024x768;1280x1024x32
```

This setting means that GRUB will first attempt to locate the 1024x768 mode, in any bit depth (higher bit depths are preferred), then it will attempt to locate 1280x1024, in a 32-bit depth. The special keyword, `text`, chooses the text mode. It should be noted that this keyword *might* not work on UEFI firmware. The `keep` keyword specifies that the mode that GRUB is using, if a graphical console type is in use, should be maintained and used by Oracle Solaris as its frame buffer console resolution.

x86: Customizing the GRUB Configuration

The `grub.cfg` file contains most of the GRUB configuration. An additional, editable file named `custom.cfg` can be used if you want to add more complex constructs, for example, menu entries or other scripting, to the GRUB configuration. This file does not exist on the system by default. You must create the file, and it must reside in the same location as the `grub.cfg` and `menu.conf` files, which is in `/`*pool-name*`/boot/grub/`.

GRUB processes the commands and any customizations that are in the `custom.cfg` file through the following code that is located at the end of the `grub.cfg` file:

```
if [ -f  $prefix/custom.cfg ]; then
   source $prefix/custom.cfg;
fi
```

These instructions direct GRUB to check for the existence of a `custom.cfg` file in the top-level dataset of the root pool, in the `boot/grub` subdirectory. If a `custom.cfg` file exists, GRUB sources the file and processes any commands that are in the file, as if the contents were textually inserted in the `grub.cfg` file.

On a system with 64-bit UEFI firmware, entries in this file might look like the following:

```
menuentry "Windows (64-bit UEFI)" {
   insmod part_gpt
   insmod fat
   insmod search_fs_uuid
   insmod chain
   search --fs-uuid --no-floppy --set=root cafe-f4ee
   chainloader /efi/Microsoft/Boot/bootmgfw.efi
}
```

On a system with BIOS firmware, entries in this file might look like the following:

```
menuentry "Windows" {
   insmod chain
   set root=(hd0,msdos1)
   chainloader --force +1
}
```

x86: Upgrading Your GRUB Legacy System to a Release That Supports GRUB 2

The following information is provided in this section:

- "How to Upgrade Your GRUB Legacy System to a Release That Supports GRUB 2" on page 49
- "How GRUB Legacy Menu Entries Are Migrated to GRUB 2" on page 52
- "Maintaining GRUB 2 and GRUB Legacy Boot Environments on the Same System" on page 53

▼ x86: How to Upgrade Your GRUB Legacy System to a Release That Supports GRUB 2

For fresh installations of an Oracle Solaris release that supports GRUB 2 as the default boot loader, nothing is required before performing the installation.

For upgrades to at least Oracle Solaris 11.1, you must install some prerequisite packages prior to the upgrade. These packages are included in the Oracle Solaris package repositories.

Before You Begin Before upgrading your system to a release that supports GRUB 2, do the following:

- Check for any known issues that might impact the installation or upgrade. See *Oracle Solaris 11.3 Release Notes*.

- Review the information and guidelines in "How GRUB Legacy Menu Entries Are Migrated to GRUB 2" on page 52 and "Maintaining GRUB 2 and GRUB Legacy Boot Environments on the Same System" on page 53.
- Preserve your existing GRUB Legacy configuration.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Install the prerequisite packages.**

   ```
   $ pkg update
   ```

3. **Reboot the system to the new boot environment that was just created in Step 2.**

4. **After your system is running in the new boot environment, update the `pkg` package with the fixes that are required to complete the upgrade by running the following command:**

   ```
   $ pkg update pkg
   ```

   Running this command updates any packages with names that match `*pkg`, which is the package that contains the `pkg` command and its dependencies.

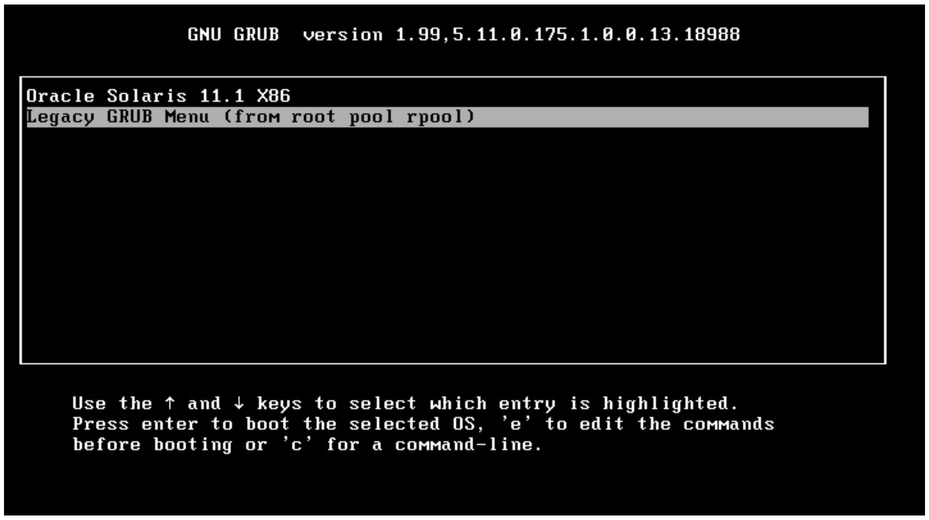
5. **To complete the update to Oracle Solaris 11.3, run the `pkg update` command one more time, as follows:**

   ```
   $ pkg update --accept
   ```

   Note - You must indicate that you agree to and accept the terms of the licenses of the packages that are listed by specifying the `--accept` option.

   The final update installs GRUB 2 as the default system boot loader. The update also creates a `grub.cfg` file that is based on the contents of the GRUB Legacy `menu.lst` file.

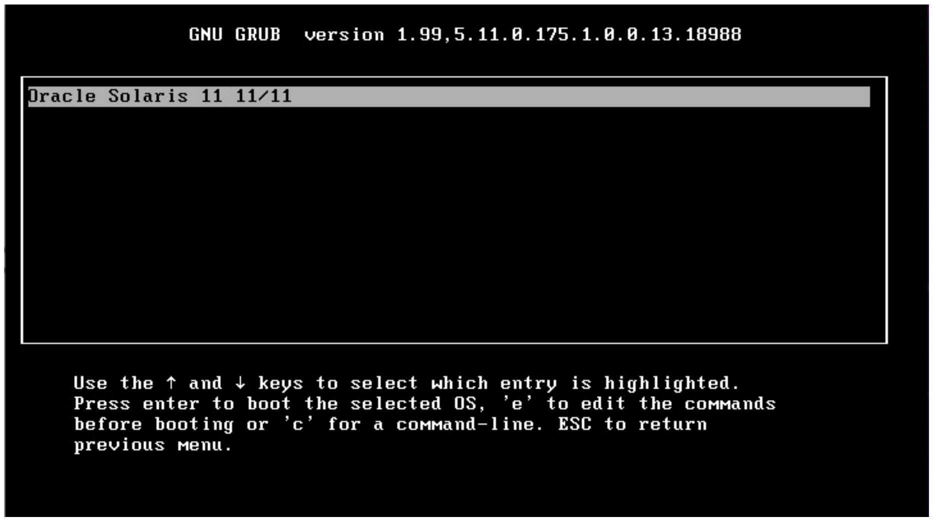
   After the new boot environment is activated, the GRUB Legacy configuration is then migrated to GRUB 2, and GRUB 2 becomes the system's default boot loader. Oracle Solaris boot entries from the `menu.lst` file are copied to the `grub.cfg` file in the order in which they appear. Any *chainloader* entries are also migrated.



```
              GNU GRUB  version 1.99,5.11.0.175.1.0.0.13.18988

 Oracle Solaris 11.1 X86
 Legacy GRUB Menu (from root pool rpool)

      Use the ↑ and ↓ keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the commands
      before booting or 'c' for a command-line.
```



```
              GNU GRUB  version 1.99,5.11.0.175.1.0.0.13.18988

 Oracle Solaris 11 11/11

      Use the ↑ and ↓ keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the commands
      before booting or 'c' for a command-line. ESC to return
      previous menu.
```

x86: How GRUB Legacy Menu Entries Are Migrated to GRUB 2

After upgrading to a version of Oracle Solaris that supports GRUB 2, all of the Oracle Solaris menu entries are automatically migrated from the GRUB Legacy `menu.lst` file to the new `grub.cfg` file. Any chainloader entries are also migrated. When the system reboots, *only* those boot entries that were migrated are displayed in the main GRUB menu. Any other boot entries that you want displayed in the main GRUB menu must be manually converted and added to the `custom.cfg` file. See "Customizing the GRUB Configuration" on page 48.

Note - *All* of the boot entries from the `menu.lst` file are present in the GRUB Legacy submenu for that root pool.

It is also important to note that GRUB 2 can directly boot all supported releases of Oracle Solaris 11, as well as Oracle Solaris 10 releases, starting with the Solaris 10 1/06 release. Previous Oracle Solaris releases can be booted indirectly by using the *chainloading* mechanism. You can add menu entries that use chainloading to the `custom.cfg` file in the same way that other custom entries are added.

Although the principle of chainloading is the same for GRUB 2 as it is for GRUB Legacy, the syntax is slightly different. In the following example, the entry is chainloaded to the master boot record (MBR) on disk 0. This type of chainloading is useful *only* if GRUB 2 is not installed in that location. Note also that chainloading this way only works on systems with BIOS firmware (which includes all Oracle Solaris 10 systems).

```
menuentry "Boot from Hard Disk" {
               set root=(hd0)
               chainloader --force +1
       }
```

In the following example, Oracle Solaris 10 is installed in the second DOS partition. In addition, the Oracle Solaris 10 version of GRUB Legacy is installed into the partition boot record (PBR) of that partition.

```
menuentry "Solaris 10" {
               set root=(hd0,msdos2)
               chainloader --force +1
       }
```

In this example, the entry is chainloaded to the Oracle Solaris 10 GRUB Legacy menu. The result is that there a two levels of menus: one to chainload from GRUB 2 to the Oracle Solaris 10 GRUB Legacy menu, and one to boot the Oracle Solaris 10 kernel from the Oracle Solaris 10 GRUB Legacy menu. To boot the system, you must select the appropriate Oracle Solaris 10 menu entry.

In addition to the Oracle Solaris menu entries that were converted from the `menu.lst` file, there is one submenu for each root pool that contains a GRUB Legacy `menu.lst` file. This submenu includes all of the menu entries in the respective `menu.lst` file and provides access to all `menu.lst` entries for maximum backward compatibility.

When booting back to an Oracle Solaris boot environment that does not contain the prerequisite packages for GRUB 2, changes to the boot configuration, for example, those that are made by using the `beadm` and `bootadm` commands, are *only* made to the `menu.lst` file for the appropriate root pool. If you then reboot the system, the GRUB 2 menu does not reflect those changes. *Only* the Legacy GRUB submenu for the appropriate root pool reflects the changes.

Additionally, these changes do not show up in the *main* GRUB menu until a GRUB 2 aware boot environment is booted, and the `grub.cfg` file is regenerated. Wherever possible, when a system runs a boot environment that uses GRUB 2, the `menu.lst` file is synchronized with the `grub.cfg` file. This synchronization occurs whenever the `beadm` or `bootadm` command is used to make changes to the GRUB 2 configuration.

x86: Maintaining GRUB 2 and GRUB Legacy Boot Environments on the Same System

You can activate GRUB 2 boot environments on a system that has GRUB Legacy boot environments, but *only* if the GRUB Legacy boot environments are GRUB 2 aware. Also, you can activate a GRUB Legacy boot environment from a GRUB 2 boot environment. One caveat for activating GRUB 2 boot environments on systems with GRUB Legacy boot environments is that you *must* install the GRUB 2 prerequisite packages in the current boot environment *before* you invoke the `pkg update` command to install an Oracle Solaris release that supports GRUB 2. See "How to Upgrade Your GRUB Legacy System to a Release That Supports GRUB 2" on page 49.

Boot environments are managed through the `beadm` command. See `beadm`(1M). When the `beadm create` command is used to create a new boot environment, a menu entry is also automatically created for that boot environment. You can display all of the boot environments that are on a system by using the `beadm list` command:

```
$ beadm list
BE                        Active Mountpoint Space Policy Created
--                        ------ ---------- ----- ------ -------
oracle-solaris11-backup -      -          64.0K static 2014-03-29 11:41
oracle-solaris2         -      -          64.0K static 2014-03-29 11:41
solaris11.3             NR     /          3.35G static 2015-05-17 13:22
```

The `beadm` command works with both GRUB 2 and GRUB Legacy configurations. When GRUB 2 boot environments are present in list of boot environments, GRUB 2 is retained as the default boot loader. Oracle Solaris does not attempt to reinstall GRUB Legacy as the default boot loader, even if a GRUB Legacy boot environment is activated. If you remove the last GRUB 2 boot environment from the system, you must manually install GRUB Legacy as the system boot loader. If the system includes the GRUB 2 prerequisite packages, you can use the `bootadm install-bootloader -f` command to manually install the boot loader. See "Installing GRUB 2 by Using the `bootadm install-bootloader` Command" on page 55. Otherwise, you can use the `installgrub` command. See `installgrub`(1M).

Manually reinstalling GRUB Legacy as the default boot loader by using the `bootadm install-bootloader -f` command forcibly installs GRUB Legacy as the system boot loader. To ensure that all boot environments remain bootable, this command must be run from the boot environment that contains the latest GRUB Legacy boot loader version. In addition, prior to reinstalling GRUB Legacy, all GRUB 2 boot environments should be removed from the system by using the `beadm` command. See "How to Install GRUB Legacy on a System That Has GRUB 2 Installed" on page 56.

Note - It is important to note that when using the `bootadm install-bootloader` command with the `-f` option on a system with an older boot loader, the older boot loader must be capable of reading the ZFS version on the boot disk. Otherwise, GRUB might not be able to read the root pool at boot time, rendering the system non-bootable.

If this situation occurs, you must install a newer boot loader by booting from another boot environment or by booting from recovery media and installing the boot loader version that matches your pool version. See "How to Boot From Media to Resolve a Problem With the GRUB Configuration That Prevents the System From Booting" on page 133.

---

x86: Advanced GRUB Administration and Troubleshooting

The following information is provided in this section:

- "Installing GRUB 2 by Using the `bootadm install-bootloader` Command" on page 55
- "How to Install GRUB Legacy on a System That Has GRUB 2 Installed" on page 56

x86: Installing GRUB 2 by Using the `bootadm install-bootloader` Command

If the GRUB 2 boot loader becomes corrupted, and the system can no longer boot, you might be required to boot from media and manually reinstall the boot loader. To reinstall the boot loader, you must boot from the Oracle Solaris installation media (for example, by using the text installer ISO image) and get to a command prompt.

▼ x86: How to Install the Boot Loader

You must import the root pool before you can reinstall GRUB 2. The following procedure describes the steps to follow.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Boot the system from the Oracle Solaris media.**

3. **Import the root pool.**

   ```
   # zpool import -f pool-name
   ```

4. **Install the boot loader.**

   ```
   # bootadm install-bootloader [-f] -P pool-name
   ```

`-f`	Forces the installation of the boot loader an bypasses any versioning checks for not downgrading the version of the boot loader on the system.

   Note - Do *not* used the `-f` option unless you are sure that you want to overwrite the boot loader with the version that is on the media.

`-P`	Specifies the boot configuration for the pool to be used

5. **Export the root pool.**

   ```
   # zpool export pool-name
   ```

6. **Reboot the system.**

▼ x86: How to Install GRUB in a Location Other Than the Default Location

On systems with BIOS firmware, sometimes it is necessary or desirable to install GRUB 2 into the master boot record. The following procedure describes how to do so. After the installation, GRUB 2 is then the default system boot loader, regardless of which DOS partition is marked as the active partition. When DOS partitioning is used on systems with BIOS firmware, and the Solaris partition is a primary partition, the default GRUB 2 installation location is the partition boot record. If the partition is a logical partition, GRUB 2 is always installed in the MBR.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Install the boot loader into the MBR location.**

   ```
   # bootadm install-bootloader -M
   ```

3. **Reboot the system.**

x86: Installing GRUB Legacy on a System That Has GRUB 2 Installed

Because the system does not automatically reinstall the GRUB Legacy boot loader when you destroy the last GRUB 2 boot environment, if you want to reinstall the GRUB Legacy boot loader, you must first boot to the latest boot environment that includes the GRUB Legacy boot loader files (in `/boot/grub/stage1` and `/boot/grub/stage2`).

The `installgrub` command is deprecated in this release and should *only* be used if you are running a release that supports the GRUB Legacy boot loader. See `installgrub`(1M).

▼ x86: How to Install GRUB Legacy on a System That Has GRUB 2 Installed

The following procedure applies if you have upgraded your system from a release that supports GRUB Legacy to Oracle Solaris 11.3.

If you decide to revert your system to the older GRUB Legacy boot loader, use the following procedure.

Caution - Be sure to perform these steps from the boot environment that contains the Oracle Solaris release or a Support Repository Update (SRU) that you used to update to Oracle Solaris 11.3. Additionally, if you have upgraded the ZFS pool's capabilities by using the `zpool upgrade` command past version 33, you will *not* be able to downgrade to GRUB Legacy or complete Step 2 of this procedure. Forcibly downgrading to GRUB Legacy after the root pool has been upgraded past version 33 results in an unbootable system.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Boot from the boot environment that was upgraded to the Oracle Solaris 11.3 release.**

3. **Remove all of the GRUB 2 boot environments from the system by using the `beadm destroy` command. See "Destroying a Boot Environment" in *Creating and Administering Oracle Solaris 11.3 Boot Environments*.**

   Performing this step ensures that you do not accidentally activate and install GRUB 2, as activating any boot environments that include the Oracle Solaris 11.3 release will replace the Legacy GRUB boot loader with GRUB 2.

4. **On the boot environment that contains the latest GRUB Legacy version, forcibly reinstall GRUB Legacy on the system, as follows:**

   ```
   # bootadm install-bootloader -f
   ```

   Note - You do not need to reboot after these steps. On the next full reboot, the GRUB Legacy boot loader will execute.

♦ ♦ ♦ CHAPTER 3

Shutting Down a System

This chapter provides overview and task-related information for shutting down an Oracle Solaris system. Any information in this chapter that applies only to SPARC or x86 based systems is identified as such.

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

For overview information about booting a system, see Chapter 1, "Overview of Booting and Shutting Down a System".

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 80.

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. System services managed by SMF are shutdown in reverse dependency order.

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 4 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 systems 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.
`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" on page 61
- "How to Shut Down a System by Using the `shutdown` Command" on page 62
- "How to Shut Down a Stand-Alone System by Using the `init` Command" on page 66

For information about shutting down a system for recovery purposes, including using the `halt` command, see "How to Stop a System for Recovery Purposes" on page 128.

▼ 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. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **For a shutdown of a system with multiple users, 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	SPARC Based System Prompt
S (single-user state)	#	#
0 (power-down state)	#	`ok` or `>`
Run level 3 (multiuser state with remote resources shared)	*hostname* `console login:`	*hostname* `console login:`

Example 14 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 2015

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 15 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 2015

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 67.

Example 16 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 2015

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

Broadcast Message from root (console) on venus Fri Apr 15 06:47:50...
The system venus will be shut down in 1 minute

Broadcast Message from root (console) on venus Fri Apr 15 06:48:20...
The system venus will be shut down in 30 seconds

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 11.3 64-bit
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.
Booting to milestone "milestone/single-user:default".
Hostname: venus
NIS domain name is solaris.example.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 Chapter 4, “Booting a System”.

▼ How to Shut Down a Stand-Alone System by Using the `init` Command

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

1. **Assume the `root` role.**

   See “Using Your Assigned Administrative Rights” in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Shut down the system.**

   ```
   # init 5
   ```

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

Example 17 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 a 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.

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.

Note - You can shut down an x86 based system by pressing the power button. Shutting the system off this way causes an ACPI event to be sent to the system, alerting the system that the user has requested a shutdown. Turning the power off this way is equivalent to running the `shutdown -i0` or `init 0` commands.

For information about turning off power to devices, see the instructions for the specified hardware in the product documentation.

♦ ♦ ♦ C H A P T E R 4

Booting a System

This chapter provides task-related information for booting and rebooting an Oracle Solaris System. Any information in this chapter that applies only to SPARC or x86 based systems is identified as such.

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

- "Displaying and Setting Boot Attributes" on page 69
- "Booting a System" on page 80
- "Booting From an Alternate Operating System or Boot Environment" on page 93
- "Rebooting a System" on page 97

For overview information about booting a system, see Chapter 1, "Overview of Booting and Shutting Down a System".

Displaying and Setting Boot Attributes

The following information describes the various ways in which you can display and set boot attributes on SPARC and x86 platforms. For specific information about setting boot attributes on x86 based systems, either at boot time or by using the `bootadm` command, see "Adding Kernel Arguments by Editing the GRUB Menu at Boot Time" on page 44.

The following procedures are provided in this section:

- "Displaying and Setting Boot Attributes by Using the OpenBoot PROM" on page 70
- "Working With EEPROM Parameters" on page 75
- "Managing Shutdown Animation Through SMF" on page 80

SPARC: Displaying and Setting Boot Attributes by Using the OpenBoot PROM

The boot PROM is used to boot a SPARC based system and to modify boot parameters. For example, you might want to reset the device from which to boot, change the default boot file or kernel, or run hardware diagnostics before bringing the system to a multiuser state.

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

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

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

▼ SPARC: How to Identify the PROM Revision Number of a System

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

```
# init 0
```

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

```
ok banner
```

▼ SPARC: How to Identify Devices on a System

You might need to identify the devices on a system to determine the appropriate devices from which to boot.

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

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

```
ok setenv auto-boot? false
```

- Issue the `reset-all` command to clear system registers.

```
ok reset-all
```

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

```
ok sifting probe
```

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

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

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

   ```
   ok probe-device
   ```

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

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

3. **Boot the system to a multiuser state.**

   ```
   ok reset-all
   ```

Example 18 SPARC: Identifying the Devices on a System

The following example shows how to identify the devices connected to a system.

```
ok setenv auto-boot? false
auto-boot? =          false
ok reset-all
SC Alert: Host System has Reset



Sun Fire T200, No Keyboard
.
.
.
Ethernet address 0:14:4f:1d:e8:da, Host ID: 841de8da.
ok probe-ide
   Device 0  ( Primary Master )
         Removable ATAPI Model: MATSHITACD-RW  CW-8124

  Device 1  ( Primary Slave )
         Not Present
```

```
  Device 2  ( Secondary Master )
         Not Present

  Device 3  ( Secondary Slave )
         Not Present

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

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

```
ok devalias
ttya                     /pci@7c0/pci@0/pci@1/pci@0/isa@2/serial@0,3f8
nvram                    /virtual-devices/nvram@3
net3                     /pci@7c0/pci@0/pci@2/network@0,1
net2                     /pci@7c0/pci@0/pci@2/network@0
net1                     /pci@780/pci@0/pci@1/network@0,1
net0                     /pci@780/pci@0/pci@1/network@0
net                      /pci@780/pci@0/pci@1/network@0
ide                      /pci@7c0/pci@0/pci@1/pci@0/ide@8
cdrom                    /pci@7c0/pci@0/pci@1/pci@0/ide@8/cdrom@0,0:f
disk3                    /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@3
disk2                    /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@2
disk1                    /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@1
disk0                    /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0
disk                     /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0
scsi                     /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2
virtual-console          /virtual-devices/console@1
name                     aliases
```

▼ SPARC: How to Determine the Default Boot Device

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

   ```
   # init 0
   ```

2. **Determine the default boot device.**

   ```
   ok printenv boot-device
   ```

`boot-device`	Identifies the value for setting the device from which to boot.

   For more information, see the `printenv(1B)` man page.

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

```
boot-device =  /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a
```

If the `boot-device` value specifies a network boot device, the output is similar to the following:

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

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

Before You Begin You might need to identify the devices on the system before you can change the default boot device to some other device. For information about identifying devices on the system, see "How to Identify Devices on a System" on page 70.

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

```
# init 0
```

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

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

`device[`*n*`]`	Identifies the `boot-device` value, such as `disk` or `network`. The *n* can be specified as a disk number. Use one of the `probe` commands if you need help identifying the disk number.

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

```
ok printenv boot-device
```

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

```
ok reset-all
```

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

Example 19 SPARC: Changing the Default Boot Device by Using the Boot PROM

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

```
# init 0
#
INIT: New run level: 0
```

```
.
.
.
The system is down.
syncing file systems... done
Program terminated
ok **setenv boot-device /pci@1f,4000/scsi@3/disk@1,0**
boot-device =         /pci@1f,4000/scsi@3/disk@1,0
ok **printenv boot-device**
boot-device           /pci@1f,4000/scsi@3/disk@1,0
ok **boot**
Resetting ...

screen not found.
Can't open input device.
Keyboard not present.  Using ttya for input and output.
.
.
.
Rebooting with command: boot disk1
Boot device: /pci@1f,4000/scsi@3/disk@1,0  File and args:
```

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

```
# **init 0**
#
INIT: New run level: 0
.
.
.
The system is down.
syncing file systems... done
Program terminated
ok **setenv boot-device net**
boot-device =         net
ok **printenv boot-device**
boot-device           net                   disk
ok **reset**
.
.
.
Boot device: net  File and args:

pluto console login:
```

Working With EEPROM Parameters

You can display and modify the value of parameters in the EEPROM by using the `eeprom` command. You do not need any special privileges to display EEPROM parameters. However, to modify these parameters, you must assume the `root` role.

Note the following additional information about how EEPROM properties are set and stored on x86 platforms.

- On x86 platforms, the setting of EEPROM properties is simulated by:
  - Storing Oracle Solaris specific properties in the `/boot/solaris/bootenv.rc` file.
  - Manipulating the GRUB menu to simulate the effect of setting certain EEPROM properties.
  - Implementing NVRAM storage for variables specific to the UEFI environment.
- Setting the `boot-args` or `boot-file` properties causes a special GRUB menu entry to be created and manipulated, as this is the only way to simulate the effect on x86 platforms. The title of the special GRUB menu entry is `Solaris bootenv rc`. This special entry is marked as the default entry when it is created.
- Properties that are set by using the `eeprom` command can be overridden by setting their property names to different values on the kernel command line, for instance, by editing the GRUB menu at boot time. One example would be to set the console property to `graphics` by using the `eeprom` command, then by adding `B console=text` to the kernel command line at boot time. In this case, the console type is set to `text`, even though the `bootenv.rc` file specifies a value of `graphics`.

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

EEPROM Parameters on UEFI Systems

For UEFI enabled systems, the parameters are stored in two places. Oracle Solaris specific variables are stored in `bootenv.rc` file. UEFI specific variables are set in the NVRAM store. Unlike SPARC with OBP, Oracle Solaris variables are not consumed by UEFI firmware. To make the UEFI specific variables available, use the `-u` option with the `eeprom` command.

Most UEFI variables are in a binary format and are translated to a readable format. When translation is not possible, a hexdump is printed.

Viewing EEPROM Parameters

EEPROM parameters vary by platform. For example, `boot-device` is a parameter on SPARC platforms, but not on x86 platforms. To view the available EEPROM parameters for your system type, use the `eeprom` command with no arguments.

EXAMPLE 20 Viewing All EEPROM Parameters

The example below shows the output of the `eeprom` command on an x86 based system:

```
$ eeprom
keyboard-layout=Unknown
ata-dma-enabled=1
atapi-cd-dma-enabled=1
ttyb-rts-dtr-off=false
ttyb-ignore-cd=true
ttya-rts-dtr-off=false
ttya-ignore-cd=true
ttyb-mode=9600,8,n,1,-
ttya-mode=9600,8,n,1,-
lba-access-ok=1
console=ttya
```

EXAMPLE 21 Viewing a Specific EEPROM Parameter

To display the value for a specific EEPROM parameter add the parameter name to the `eeprom` command as follows:

```
$ /usr/sbin/eeprom console
console=ttya
```

EXAMPLE 22 Viewing All UEFI EEPROM Parameters

The example below shows how to display all of the UEFI parameters on a system in UEFI mode. You must assume the `root` role to use this command.

```
# eeprom -u
MonotonicCounter=0x1f2
OsaBootOptNum=0xffff
ConOut=/PciRoot(0x0)/Pci(0x1c,0x7)/Pci(0x0,0x0)/Pci(0x0,0x0)/AcpiAdr(2147549440)
 /PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
```

```
ConIn=/PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
 /PciRoot(0x0)/Pci(0x1d,0x0)/USB(0x1,0x0)/USB(0x8,0x0)
BootOrder=Boot0000 Boot0001 Boot0002 Boot0003 Boot0004 Boot0005 Boot0006
Lang=eng
PlatformLang=en-US
Timeout=0x1
Boot0001=description:string=[UEFI]USB:USBIN:USB USB Hard Drive , flags:int=1,
 device_path: \
 string=/PciRoot(0x0)/Pci(0x1a,0x0)/USB(0x1,0x0)/USB(0x2,0x0)/
HD(1,MBR,0x004D5353,0x800,0x3b5800), \
 optional_data:string=AMBO
Boot0002=description:string=[UEFI]PXE:NET0:Intel(R) Ethernet Controller 10 Gigabit X540-
AT2, \
 flags:int=1, device_path:string=/PciRoot(0x0)/Pci(0x2,0x0)/Pci(0x0,0x0)/
MAC(002128e77478), \
 optional_data:string=AMBO
Boot0003=description:string=[UEFI]PXE:NET1:Intel(R) Ethernet Controller 10 Gigabit X540-
AT2, \
 flags:int=1, device_path:string=/PciRoot(0x0)/Pci(0x2,0x0)/Pci(0x0,0x1)/
MAC(002128e77479), \
 optional_data:string=AMBO
Boot0004=description:string=[UEFI]PXE:NET2:Intel(R) Ethernet Controller 10 Gigabit X540-
AT2, \
 flags:int=1, device_path:string=/PciRoot(0x1)/Pci(0x1,0x0)/Pci(0x0,0x0)/
MAC(002128e7747a), \
 optional_data:string=AMBO
Boot0005=description:string=[UEFI]PXE:NET3:Intel(R) Ethernet Controller 10 Gigabit X540-
AT2, \
 flags:int=1, device_path:string=/PciRoot(0x1)/Pci(0x1,0x0)/Pci(0x0,0x1)/
MAC(002128e7747b), \
 optional_data:string=AMBO
Boot0006=description:string=[UEFI]SAS:PCIE3:ATA     HITACHI HDS7225SA81A, flags:int=1, \
 device_path:string=/PciRoot(0x0)/Pci(0x3,0x0)/Pci(0x0,0x0) \
 /MessagingPath(10,2c00b .... 12010100) \
 /HD(1,GPT,BCB01265-4665-F1CA-8BF5-9C4FB95962FA,0x100,0x80000),
 optional_data:string=AMBO
Boot0000=description:string=Oracle Solaris s12_13, flags:int=1, device_path: \
 string=HD(1,GPT,C7398875-60D2-A9E0-83EE-94DAA21B0383,0x100,0x80000),
 file_path:string=/EFI/Oracle/grubx64.efi
USB_POINT=5139417f00000000
ConOutDev=/PciRoot(0x0)/Pci(0x1c,0x7)/Pci(0x0,0x0)/Pci(0x0,0x0)/AcpiAdr(2147549440)
 /PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
ConInDev=/PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
 /PciRoot(0x0)/Pci(0x1d,0x0)/USB(0x1,0x0)/USB(0x8,0x0)
BootOptionSupport=0x1
```

```
ErrOutDev=/PciRoot(0x0)/Pci(0x1c,0x7)/Pci(0x0,0x0)/Pci(0x0,0x0)/AcpiAdr(2147549440)
 /PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
ErrOut=/PciRoot(0x0)/Pci(0x1c,0x7)/Pci(0x0,0x0)/Pci(0x0,0x0)/AcpiAdr(2147549440)
 /PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
PlatformLangCodes=en-US
S3PerfAdd=hexdump:989fd6aa00000000
LangCodes=eng
BootCurrent=Boot0000
```

EXAMPLE 23 Viewing a Specific UEFI Parameter

```
# eeprom -u Boot0000
Boot0000=description:string=Oracle Solaris s12_13, flags:int=1, device_path: \
 string=HD(1,GPT,C7398875-60D2-A9E0-83EE-94DAA21B0383,0x100,0x80000), \
 file_path:string=/EFI/Oracle/grubx64.efi
```

▼ SPARC: How to Set a Boot Attribute

The following procedure describes how to set the default boot device on a SPARC based system. On x86 platforms, the boot device is set through the setup utility for your firmware type, for example, UEFI Boot Manager.

Note - On x86 platforms, the boot device is set through the setup utility for your firmware type, for example, UEFI Boot Manager.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Specify the boot attribute.**

   ```
   # eeprom attribute=value
   ```

3. **Verify that the attribute has been set.**

   ```
   # eeprom attribute
   ```

   The output should display the new `eeprom` value for the changed attribute.

Example 24 Setting the `auto_boot` Parameter

To set the `auto_boot` parameter to false, type the following command using the `root` role:

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

Example 25 Setting Kernel Boot Arguments

You can set kernel boot arguments by specifying a value for the `boot-args` parameter. For example, type the following command to specify that the system boot the kernel debugger:

```
# eeprom boot-args=-k
```

Example 26 Setting Parameters for the Console Device

To switch the Oracle Solaris console setting to graphic mode, use the following command:

```
# eeprom console=graphics
```

Example 27 Setting a Parameter on a UEFI Enabled System

This example shows how to change the boot order on a UEFI enabled system:

```
# eeprom -u BootOrder="Boot0005 Boot0001 Boot0002 Boot0003 Boot0004 Boot0000"
```

▼ How to Delete a UEFI EEPROM Parameter

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Delete the UEFI EEPROM parameter.**

   In this example, a custom parameter named `attribute` is deleted.

   ```
   # eeprom -u -d attribute
   ```

3. **Verify that the attribute has been deleted.**

   ```
   # eeprom -u attribute
   ```

```
eeprom: read: attribute doesn't exist
```

x86: Managing Shutdown Animation Through SMF

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
```

Booting a System

The following procedures describe how to boot a system to various states, also known as *run level booting*.

The following procedures are provided in this section:

- "How Run Levels Work" on page 80
- "How to Boot a System to a Multiuser State (Run Level 3)" on page 83
- "How to Boot a System to a Single-User State (Run Level S)" on page 85
- "How to Boot a System Interactively" on page 89

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 5 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.

Run Level	Init State	Type	Purpose
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.
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 stop the operating system and reboot to the state that is defined by the `initdefault` entry in the `/etc/inittab` file. The SMF service, `svc:/system/boot-config:default`, is enabled by default. When the `config/fastreboot_default` property is set to true, `init 6` bypasses certain firmware initialization and test steps, depending on the specific capabilities of the system. See "Accelerating the Reboot Process" on page 100.

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 6 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 Investigate Problems Starting Services at System Boot" in *Managing System Services in Oracle Solaris 11.3*.

Determining a System's Current Run Level

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

EXAMPLE 28 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

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

Use this procedure to boot a system that is currently at run level 0 to run level 3. Any information in this procedure that applies to either the SPARC or x86 platforms is noted accordingly.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Depending on the platform, do one of the following:**

   - **For SPARC platforms:**

     a. **Bring the system to the `ok` PROM prompt.**

     ```
     # init 0
     ```

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

     ```
     ok boot
     ```

   - **For x86 platforms, reboot the system to run level 3.**

     ```
     # reboot
     ```

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

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

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

   ```
   hostname console login:
   ```

Example 29 SPARC: Booting a System to a Multiuser State (Run Level 3)

The following example shows the messages from booting a SPARC based system to run level 3 after the boot process has started.

```
ok boot
Probing system devices
Probing memory
```

```
ChassisSerialNumber FN62030249
Probing I/O buses

.
.
.
.
OpenBoot 4.30.4.a, 8192 MB memory installed, Serial #51944031.
Ethernet address 0:3:ba:18:9a:5f, Host ID: 83189a5f.
Rebooting with command: boot
Boot device: /pci@1c,600000/scsi@2/disk@0,0:a  File and args:
SunOS Release 5.11 Version 11.3 64-bit
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.
misc/forthdebug (455673 bytes) loaded
Hardware watchdog enabled
Hostname: portia-123
NIS domain name is solaris.example.com

portia-123 console login: NIS domain name is solaris.example.com
```

Example 30 x86: Booting a System to a Multiuser State (Run Level 3)

The following example shows the messages when booting an x86 based system to run level 3 after the boot process has started. Because the Fast Reboot feature is the default in this release (on x86 platforms), booting the system with the `reboot` command initiates a fast reboot of the system, meaning the BIOS or UEFI firmware is bypassed. Also, the GRUB menu is not displayed during the system boot. If you need to access the system's firmware or edit the GRUB menu at boot time, use the `reboot` command with the `-p` option. See "Initiating a Standard Reboot of a System That Has Fast Reboot Enabled" on page 104.

```
~# reboot
Apr 23 13:30:29 system-04 reboot: initiated by ... on /dev/console
Terminated
system-04% updating /platform/i86pc/boot_archive
updating /platform/i86pc/amd64/boot_archive

system-04 console login: syncing file systems... done
SunOS Release 5.11 Version 11.3 64-bit
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.
Hostname: system-04

system-04 console login: <username>
Password: xxxxxx
Last login: Mon Apr 23 11:06:05 on console
Oracle Corporation      SunOS 5.11      11.3    May 2015
# who -r
  run-level 3  Apr 23 13:31     3      0  S
```

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

You boot a system to a single-user state for the purpose of performing system maintenance, such as backing up a file system or to troubleshoot other system issues.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Depending on the platform, do one of the following:**

   - **For SPARC platforms:**

     a. **Bring the system to the `ok` PROM prompt.**

     ```
     # init 0
     ```

     b. **Boot the system to a single-user state.**

     ```
     ok boot -s
     ```

     c. **Type the `root` password when the following message is displayed:**

     ```
     SINGLE USER MODE

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

   - **For x86 platforms:**

     a. **Perform a standard reboot of the system.**

     ```
     # reboot -p
     ```

     Because the Fast Reboot feature is enabled by default, you must specify the `-p` option when rebooting the system, which enables the GRUB menu to be displayed at boot time. To disable the Fast Reboot feature so that the `-p` option does not need to be specified, see "Changing the Default Fast Reboot Behavior" on page 103.

     - **If the system displays the Press Any Key to Reboot prompt, press any key to reboot the system. Or, you can also use the Reset button at this prompt.**

- **If the system is shut down, turn the system on with the power switch.**

b. **When the GRUB menu is displayed, select the boot entry that you want to modify, then type `e` to edit that entry.**

c. **Using the arrow keys, navigate to the `$multiboot` line, then type `-s` at the end of the line.**

d. **To exit the GRUB edit menu and boot the entry you just edited, press Control-X. If you are not using a serial console on a system with UEFI firmware, pressing F10 also boots the entry.**

   See "Adding Kernel Arguments by Editing the GRUB Menu at Boot Time" on page 44 for more information about editing the GRUB menu at boot time.

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

```
# who -r
```

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

5. **Reboot the system.**

Example 31 SPARC: Booting a System to a Single-User State (Run Level S)

The following example shows the messages from booting a SPARC based system to run level S after the boot process has started.

```
# init 0
# svc.startd: The system is coming down.  Please wait.
svc.startd: 122 system services are now being stopped.
Mar  5 10:30:33 system1 syslogd: going down on signal 15
svc.startd: Killing user processes.
umount: /ws busy
umount: /home busy
Mar  5 17:30:50 The system is down.  Shutdown took 70 seconds.
syncing file systems... done
Program terminated
{1c} ok boot -s

SC Alert: Host System has Reset
NOV 17 21:46:59 ERROR: System memory downgraded to 2-channel mode from 4-channel mode
NOV 17 21:47:00 ERROR: Available system memory is less than physically installed memory
NOV 17 21:47:00 ERROR: System DRAM  Available: 008192 MB  Physical: 016384 MB
```

```
Sun Fire T200, No Keyboard
.
.
.
Ethernet address 0:14:4f:1d:e8:da, Host ID: 841de8da.



ERROR: The following devices are disabled:
    MB/CMP0/CH2/R0/D0

Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a
File and args: -s

SunOS Release 5.11 Version 11.3 64-bit
Copyright (c) 1983, 2015 Oracle and/or its affiliates. All rights reserved.
NOTICE: Hypervisor does not support CPU power management
Booting to milestone "milestone/single-user:default".
Hostname: system1
Requesting System Maintenance Mode
SINGLE USER MODE
Enter root password (control-d to bypass): xxxxxx
single-user privilege assigned to root on /dev/console.
Entering System Maintenance Mode

Mar  5 10:36:14 su: 'su root' succeeded for root on /dev/console
Oracle Corporation      SunOS 5.11      11.3    ay 2015
root@system1:~# who -r
run-level S  Mar  5 10:35     S      0  0
root@tsystem1:~#
```

Example 32 x86: Booting a System to a Single-User State (Run Level S)

The following example shows the messages from booting an x86 based system to run level S after the boot process has started.

```
root@system-04:~# init 0
root@system-04:~# svc.startd: The system is coming down.  Please wait.
svc.startd: 129 system services are now being stopped.
Apr 23 13:51:28 system-04 syslogd: going down on signal 15
svc.startd: Killing user processes.
umount: /home busy
Apr 23 13:51:36 The system is down.  Shutdown took 26 seconds.
syncing file systems... done
Press any key to reboot.
.
.
.LSI Corporation MPT SAS BIOS
```

```
MPTBIOS-6.26.00.00 (2008.10.14)
Copyright 2000-2008 LSI Corporation.

Initializing..|Press F2 to runS POPUP  (CTRL+P on Remote Keyboard)
Press F12 to boot from the network (CTRL+N on Remote Keyboard)
System Memory : 8.0 GB , Inc.
Auto-Detecting Pri Master..ATAPI CDROM                              0078
            Ultra DMA Mode-2
.
.
.
GNU GRUB  version 1.99,5.11.0.175.1.0.0.14.0

 ****************************************************************************
 *Oracle Solaris 11.3                                                      *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 ****************************************************************************

      Use the * and * keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the commands
      before booting or 'c' for a command-line.

 GNU GRUB  version 1.99,5.11.0.175.1.0.0.14.0

 ****************************************************************************
 * setparams 'Oracle Solaris 11.3'                                          *
 *                                                                          *
 * insmod part_msdos                                                        *
 * insmod part_sunpc                                                        *
 * insmod part_gpt                                                          *
 * insmod zfs                                                               *
 * search --no-floppy --fs-uuid --set=root cd03199c4187a7d7                 *
 * zfs-bootfs /ROOT/s11.3/@/ zfs_bootfs                                     *
 * set kern=/platform/i86pc/kernel/amd64/unix                               *
 * echo -n "Loading ${root}/ROOT/s11.3   /@$kern: "                         *
 * $multiboot /ROOT/s11.3@/$kern $kern -B $zfs_bootfs -s
 * set gfxpayload="1024x768x32;1024x768x16;800x600x16;640x480x16;640x480x1\ *
 * 5;640x480x32"                                                            **
 ****************************************************************************
```

```
      Minimum Emacs-like screen editing is supported. TAB lists
      completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for
      a command-line or ESC to discard edits and return to the GRUB menu.

Booting a command list

Loading hd0,msdos1,sunpc1/ROOT/s11.3/@/platform/i86pc/kernel/amd64/unix: 0
%...done.
Loading hd0,msdos1,sunpc1/ROOT/s113/@/platform/i86pc/amd64/boot_archive:
0%...
.
.
.
SunOS Release 5.11 Version 11.3 64-bit
Copyright (c) 1983, 201, Oracle and/or its affiliates. All rights reserved.
NOTICE: kmem_io_2G arena created
Booting to milestone "milestone/single-user:default".
Hostname: system-04
Requesting System Maintenance Mode
SINGLE USER MODE


Enter user name for system maintenance (control-d to bypass): root
Enter root password (control-d to bypass): xxxxxxx
single-user privilege assigned to root on /dev/console.
Entering System Maintenance Mode

May  8 11:13:44 su: 'su root' succeeded for root on /dev/console
Oracle Corporation      SunOS 5.11      11.3    May 2015
You have new mail.

root@system-04:~# who -r
   .       run-level S  Apr 23 14:49     S      0  0
```

▼ How to Boot a 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 because the original file is damaged or the system will not boot. Use the following procedure to boot a system interactively.

The following procedure describes how to specify an alternate `/etc/system` file during an interactive boot of a system that has only one boot environment. Alternatively, you can boot an alternative boot environment.

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

   ```
   # 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. **Depending on the platform, do one of the following:**

   - **For SPARC platforms:**

     a. **Bring the system to the `ok` PROM prompt.**

        ```
        # init 0
        ```

     b. **Boot the system interactively.**

        ```
        ok boot -a
        ```

   - **For x86 platforms:**

     a. **Perform a standard reboot of the system.**

        ```
        # reboot -p
        ```

     b. **When the GRUB menu is displayed, select the boot entry that you want to boot interactively, then type `e` to edit the entry.**

     c. **Type `-a` at the end of the `$multiboot` line.**

     d. **To exit the GRUB edit menu and boot the entry you just edited, press Control-X. If you have a system with UEFI firmware, and you are not using a serial console, pressing F10 also boots the entry.**

3. **When prompted for the alternate file system, specify the backup file that you created, the press Return. For example:**

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

   Pressing Return without providing any information accepts the system defaults.

4. **At the Retire store prompt, press Return or specify `/dev/null` to bypass.**

Note - The `/etc/devices/retire_store` file is the backing store for devices that are retired by the Fault Management Architecture (FMA). The system no longer uses these devices. You can provide an alternate file for `/etc/devices/retire_store`, if necessary. However, for recovery purposes, specifying `/dev/null` is the most useful choice to boot the system without respecting the contents of the `/etc/devices/retire_store` file.

---

5. **After the system has booted, correct the problem with the `/etc/system` file.**

6. **Reboot the system.**

```
# reboot
```

Example 33 SPARC: Booting a System Interactively

In the following example, the system defaults (shown in square brackets `[]`) are accepted.

```
# init 0
# svc.startd: The system is coming down.  Please wait.
svc.startd: 121 system services are now being stopped.
Apr 22 00:34:25 system-28 syslogd: going down on signal 15
svc.startd: Killing user processes.
umount: /home busy
Apr 22 06:34:37 The system is down.  Shutdown took 18 seconds.
syncing file systems... done
Program terminated
{11} ok boot -a

SC Alert: Host System has Reset

Sun Fire T200, No Keyboard
Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved.
OpenBoot 4.30.4.d, 16256 MB memory available, Serial #74139288.
Ethernet address 0:14:4f:6b:46:98, Host ID: 846b4698.

Boot device: /pci@780/pci@0/pci@9/scsi@0/disk@0,0:a  File and args: -a
Name of system file [/etc/system]:  /etc/system.bak
SunOS Release 5.11 Version 11.3 64-bit
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.
Retire store [/etc/devices/retire_store] (/dev/null to bypass): Press Return


system-28 console login:
```

Example 34 x86: Booting a System Interactively

In the following example, an x86 based system is booted interactively.

```
root@system-04:~# reboot -p
Apr 23 15:37:04 system-04 reboot: initiated by user1 on /dev/consoleTerminated
system-04% syncing file systems... done
rebooting...
.
.
.LSI Corporation MPT SAS BIOS
MPTBIOS-6.26.00.00 (2008.10.14)
Copyright 2000-2008 LSI Corporation.

Initializing..|Press F2 to runS POPUP  (CTRL+P on Remote Keyboard)
Press F12 to boot from the network (CTRL+N on Remote Keyboard)
System Memory : 8.0 GB , Inc.
Auto-Detecting Pri Master..ATAPI CDROM                              0078
            Ultra DMA Mode-2
GNU GRUB  version 1.99,5.11.0.175.1.0.0.14.0

 ****************************************************************************
 *Oracle Solaris 11.3                                                       *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 *                                                                          *
 ****************************************************************************

      Use the * and * keys to select which entry is highlighted.
      Press enter to boot the selected OS, 'e' to edit the commands
      before booting or 'c' for a command-line.

 GNU GRUB  version 1.99,5.11.0.175.1.0.0.15.1

 +--------------------------------------------------------------------------+
 | setparams 'Oracle Solaris 11.3'                                          |
 |                                                                          |
 | insmod part_msdos                                                        |
 | insmod part_sunpc                                                        |
 | insmod part_gpt                                                          |
 | insmod zfs                                                            |
 | search --no-floppy --fs-uuid --set=root cd03199c4187a7d7              |
 | zfs-bootfs /ROOT/s11.3/@/ zfs_bootfs                                  |
 | set kern=/platform/i86pc/kernel/amd64/unix                            |
 | echo -n "Loading ${root}/ROOT/s11./@$kern: "                         |
 | $multiboot /ROOT/s11.3/@/$kern $kern -B $zfs_bootfs -a                |
 | set gfxpayload="1024x768x32;1024x768x16;800x600x16;640x480x16;640x480x1\ |
 +-----------------------------------------------------------------------+

      Minimum Emacs-like screen editing is supported. TAB lists
      completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for
      a command-line or ESC to discard edits and return to the GRUB menu.

 Booting a command list

Loading hd0,msdos1,sunpc1/ROOT/s11.3/@/platform/i86pc/kernel/amd64/unix: 0
%...done.
Loading hd0,msdos1,sunpc1/ROOT/s11.3/@/platform/i86pc/amd64/boot_archive:
0%...
.
.
.
Name of system file [/etc/system]:  /etc/system.bak
SunOS Release 5.11 Version 11.3 64-bit
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.: 0
Retire store [/etc/devices/retire_store] (/dev/null to bypass): Press Return

NOTICE: kmem_io_2G arena created
Hostname: system-04

system-04 console login:
```

Booting From an Alternate Operating System or Boot Environment

The following procedures are provided in this section:

- SPARC: How to Boot From an Alternate Operating System or Boot Environment
- x86: How to Boot From an Alternate Operating System or Boot Environment

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. For more information about the `beadm` utility, see the `beadm`(1M) man page. For more information about managing boot environments, including using the utility in a global or non-global zone, see *Creating and Administering Oracle Solaris 11.3 Boot Environments*.

x86 only: If the device that is identified by GRUB as the boot device contains a ZFS storage pool, the `grub.cfg` file that is used to create the GRUB menu can be found in the pool's top level dataset. This is the dataset that has the same name as the pool. There is always exactly one such dataset in a pool. This dataset is well-suited for pool-wide data, such as the GRUB configuration files and data. After the system is booted, this dataset is mounted at /*pool-name* in the root file system.

x86 only: There can be multiple bootable datasets (that is, root file systems) within a pool. The default root file system in a pool is identified by the pool' s `bootfs` property. If a specific `bootfs` is not specified with the `zfs-bootfs` command in a GRUB menu entry located in the `grub.cfg` file, the default `bootfs` root file system is used. Each GRUB menu entry can specify a different `zfs-bootfs` command to use, which enables you to choose any bootable Oracle Solaris instance within a pool. For more information, see the `boot`(1M) man page.

▼ SPARC: How to Boot From an Alternate Operating System or Boot Environment

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

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

   ```
   # init 0
   ```

3. **(Optional) Display a list of available boot environments by using the `boot` command with the `-L` option.**

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

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

5. **To boot the system, follow the instructions that are displayed on the screen.**

   ```
   To boot the selected entry, invoke:
   ```

```
boot [<root-device>] -Z rpool/ROOT/boot-environment
```

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

For example:

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

6. **After the system has booted, verify the active boot environment.**

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

7. **(Optional) To display the boot path for the active boot environment, type the following command:**

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

8. **(Optional) To determine whether the correct boot environment was booted, type the following command:**

   ```
   # df -lk
   ```

Example 35 SPARC: Booting From an Alternate Boot Environment

This example shows how to use the `boot -Z` command to boot from an alternate boot environment on a SPARC based system.

```
# init 0
root@system-28:~# svc.startd: The system is coming down.  Please wait.
svc.startd: 126 system services are now being stopped.
Jul  3 22:11:33 system-28 syslogd: going down on signal 15
svc.startd: Killing user processes.
umount: /home busy
Jul  3 22:11:50 The system is down.  Shutdown took 23 seconds.
syncing file systems... done
Program terminated
{1c} ok boot -L

SC Alert: Host System has Reset

Sun Fire T200, No Keyboard
Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved.
OpenBoot 4.30.4.d, 16256 MB memory available, Serial #74139288.
Ethernet address 0:14:4f:6b:46:98, Host ID: 846b4698.

Boot device: /pci@780/pci@0/pci@9/scsi@0/disk@0,0:a  File and args: -L
1 Oracle Solaris 11.3 SPARC
2 s11.3_backup
```

```
3 s11.3_backup2
Select environment to boot: [ 1 - 3 ]: 3

To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/s11.3_backup2

Program terminated
{0} ok boot -Z rpool/ROOT/s11.3_backup2

SC Alert: Host System has Reset

Sun Fire T200, No Keyboard
Copyright (c) 1998, 2015, Oracle and/or its affiliates. All rights reserved.
OpenBoot 4.30.4.d, 16256 MB memory available, Serial #74139288.
Ethernet address 0:14:4f:6b:46:98, Host ID: 846b4698.



Boot device: /pci@780/pci@0/pci@9/scsi@0/disk@0,0:a  \
File and args: -Z rpool/ROOT/s11.3_backup2
SunOS Release 5.11 Version 11.3 64-bit
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.
WARNING: consconfig: cannot find driver for
screen device /pci@780/pci@0/pci@8/pci@0/TSI,mko@0
Loading smf(5) service descriptions: Loading smf(5)
service descriptions: Hostname: system-28
.


system-28 console login: Jul  3 22:39:05 system-28
```

▼ x86: How to Boot From an Alternate Operating System or Boot Environment

1. **Assume the `root` role.**

   See “Using Your Assigned Administrative Rights” in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Perform a standard reboot the system.**

   ```
   # reboot -p
   ```

3. **When the GRUB menu is displayed, navigate to the alternate boot environment or operating system that you want to boot.**

4. **To boot from the alternate operating system, press Control-X.**
   If you have a system with UEFI firmware, and you are not using a serial console, pressing F10 also boots the alternate operating system.

Example 36 Booting From an Alternate Boot Environment by Using the `reboot` Command

You can boot an alternate boot entry by using the `reboot` command specifying the boot entry number, as shown in the following example:

```
# bootadm list-menu
the location of the boot loader configuration files is: /rpool/boot/grub
default 1
timeout 30
0 s11.s.backup
1 Oracle Solaris 11.s B14
# reboot 1
Apr 23 16:27:34 system-04 reboot: initiated by userx on /dev/consoleTerminated
system-04% syncing file systems... done
SunOS Release 5.11 Version 11.s 64-bit
Copyright (c) 1983, 2015, Oracle and/or its affiliates. All rights reserved.

Hostname: system-04

system-04 console login:
```

Rebooting a System

The following procedures are provided in this section:

- "How to Reboot a System by Using the `init` Command" on page 98
- "How to Reboot a System by Using the `reboot` Command" on page 99
- "Accelerating the Reboot Process" on page 100

Normally, the system reboots at power-up or after a system crash. You can reboot a system by using either the `init` command or the `reboot` command. The `init 6` command asks for stop methods (either SMF or `rc.d`). Whereas, the `reboot` command does not, thereby making the `reboot` command a more reliable way of rebooting a system. See `init`(1M) and `reboot`(1M) for details.

The `reboot` performs the following actions:

- Restarts the kernel
- Performs a `sync` operation on the disks
- Initiates a multi-user boot.

Although the `reboot` command can be used by the `root` user at any time, in certain cases, as with the reboot of a system with multiple users, the `shutdown` command is 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".

▼ How to Reboot a System by Using the `init` 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. See "How Run Levels Work" on page 80.

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. The `init 6` command stops the operating system and reboots to the state that is defined by the `initdefault` entry in the `/etc/inittab` file.

Note - Starting with the Oracle Solaris 11 release, the SMF service, `svc:/system/boot-config:default`, is enabled by default. When the `config/fastreboot_default` property is set to true (which is the case for all x86 based systems), the `init 6` command bypasses certain firmware initialization and test steps, depending on the specific capabilities of the system. On SPARC based systems, this property is set to `false` by default, but the property can be manually enabled. See "Accelerating the Reboot Process" on page 100.

---

1. **Assume the `root` role.**
   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

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 37 Rebooting a System to a Single-User State (Run Level S) by Using the `init` Command

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

```
~# init s
~# svc.startd: The system is coming down for administration.  Please wait.
Jul 20 16:59:37 system-04 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): root
Enter root password (control-d to bypass): xxxxxx
single-user privilege assigned to root on /dev/console.
Entering System Maintenance Mode

Jul 20 17:11:24 su: 'su root' succeeded for root on /dev/console
Oracle Corporation      SunOS 5.11      11.3    May 2015
You have new mail.
~# who -r
   .       run-level S  Jul 20 17:11     S      1  3
```

▼ How to Reboot a System by Using the `reboot` Command

Use this procedure to reboot a running system to a multiuser state (run level 3).

Note - On x86 platforms, using the `reboot` command initiates a fast reboot of the system, bypassing the BIOS or UEFI firmware and certain boot processes. To perform a standard reboot of an x86 based system that has the Fast Reboot feature enabled, use the `-p` option with the `reboot` command. See "Initiating a Standard Reboot of a System That Has Fast Reboot Enabled" on page 104.

---

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Reboot the system.**

   ```
   # reboot
   ```

Accelerating the Reboot Process

The Fast Reboot feature of Oracle Solaris is supported on both SPARC and x86 platforms. The Fast Reboot feature 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.

Support for the Fast Reboot feature 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. By default, this property is set to `true` on an x86 based system and `false` on a SPARC based system.

On an x86 based system, a fast reboot of the system bypasses the system firmware (BIOS or UEFI) and the boot loader processes. Fast Reboot and Panic Fast Reboot (a fast reboot of system after a system panic) are enabled by default on x86 platforms, so there is no need to use the `-f` option with the `reboot` command to initiate a fast reboot of an x86 based system.

The Fast Reboot feature works differently on SPARC based systems than it does on an x86 based systems. Note the following additional information about Fast Reboot support on SPARC platforms:

- Fast Reboot is *not supported* on `sun4u` systems.
- Fast Reboot is supported on `sun4v` systems. However, a fast reboot of a SPARC based system is *not* the same as a fast reboot of an x86 based system. On SPARC `sun4v` systems, a fast reboot is a minimal hypervisor initiated restart that delivers that same basic performance as a fast reboot of an x86 based system.
- Fast Reboot behavior on SPARC based systems is not enabled by default. To initiate a fast reboot of a SPARC based system, you must use the `-f` option with the `reboot` command. Or, to make a fast reboot the default behavior, you can set the `config/fastreboot_default` property to `true`. For instructions, see "Changing the Default Fast Reboot Behavior" on page 103.
- On SPARC based systems the `boot-config` service also requires the `solaris.system.shutdown` authorization as the `action_authorization` and `value_authorization`.

x86: About the `quiesce` Function

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

Note - Not all device drivers implement the `quiesce` function. For troubleshooting instructions, see "Conditions Under Which Fast Reboot Might Not Work" on page 141 and "How to Clear a Failed Automatic Boot Archive Update on a System That Does Not Support Fast Reboot" on page 126.

▼ How to Initiate a Fast Reboot of a System

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Depending on the system's platform, do one of the following:**

   - **On a SPARC based system, type the following command:**

     ```
     # reboot -f
     ```

   - **On an x86 based system, type either of the following commands:**

     ```
     # reboot
     ```

     ```
     # init 6
     ```

     Running either of these commands reboots the system to the default entry in the `grub.cfg` file.

x86: Initiating a Fast Reboot of a System to a Newly Activated 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 38 x86: Initiating a Fast Reboot of a System to a Newly Activated Boot Environment

The following example shows activate a boot environment named `2013-06-10-be`, so that it will be fast rebooted.

```
# beadm activate 2013-06-10-be
# reboot
```

EXAMPLE 39 x86: Initiating a Fast Reboot of a System While 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:

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

For example, you would initiate a fast reboot of a system to an alternate ZFS root dataset as follows:

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

EXAMPLE 40 x86: Initiating a Fast Reboot of a System to an Alternate Boot Environment With the Kernel Debugger Enabled

Initiate a fast reboot of a system to the `zfsbe3` boot environment as follows:

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

EXAMPLE 41 x86: Initiating a Fast Reboot of a System to a New Kernel

Initiate a fast reboot of a system to a new kernel named `my-kernel` as follows:

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

EXAMPLE 42 x86: Initiating a Fast Reboot of a Mounted Disk or a Mounted Dataset

Initiate a fast reboot of a mounted disk or a mounted dataset as follows:

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

EXAMPLE 43 x86: Initiating a Fast Reboot of a System to a Single-User State With the Kernel Debugger Enabled

Initiate a fast reboot of a system to a single-user state, with the kernel debugger enabled, as follows:

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

Changing the Default Fast Reboot Behavior

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 and `false` on a SPARC based system.

EXAMPLE 44 x86: Configuring Properties of the `boot-config` Service

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

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

These properties 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
```

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.

EXAMPLE 45 SPARC: Configuring Properties of the `boot-config` Service

The following example shows how to make a fast reboot the default behavior on a SPARC based system by setting the `boot-config` SMF service property to `true`.

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

Setting the property's value to `true` accelerates the reboot process, which enables systems that support the Fast Reboot feature to bypass certain POST tests. When the property is set to `true`, you can perform a fast reboot of the system without having to use the `-f` option with the `reboot` command.

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

To reboot a 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
```

♦ ♦ ♦ C H A P T E R 5

Booting a System From the Network

This chapter provides overview, guidelines, and task-related information for booting SPARC and x86 based systems from the network. Any information in this chapter that applies only to SPARC or x86 based systems is identified as such.

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

- SPARC: Booting a System From the Network
- "Booting a System From the Network" on page 110

For overview information about booting a system, see Chapter 1, "Overview of Booting and Shutting Down a System".

For information about booting a system from the network for the purpose of installing Oracle Solaris, see *Installing Oracle Solaris 11.3 Systems*.

SPARC: Booting a System From the Network

The following procedures are provided in this section:

- "Network Boot Processes" on page 106
- "Requirements for Booting a System From the Network" on page 106
- "Setting Network Boot Arguments in the OpenBoot PROM" on page 106
- "Setting Up an NVRAM Alias to Automatically Boot by Using DHCP" on page 108
- "How to Boot a System From the Network" on page 109

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).

For general information about how DHCP works in this Oracle Solaris release and specific information about setting up a DHCP server, see *Working With DHCP in Oracle Solaris 11.3*.

SPARC: Network Boot Processes

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

When booting over a LAN, the firmware uses DHCP to discover either the boot server or the install server. The Trivial File Transfer Protocol (TFTP) is then used to download the booter, which is `inetboot` in this case.

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

SPARC: Requirements for Booting a 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, if the system cannot boot from the local disk.

- To perform a network boot of a SPARC based system to install Oracle Solaris for recovery purposes, a DHCP server is required.

  The DHCP server supplies the information that the boot 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, see *Working With DHCP in Oracle Solaris 11.3*.
- A boot server that provides `tftp` service is also required.

SPARC: Setting Network Boot Arguments in the OpenBoot PROM

The `network-boot-arguments` parameter of the `eeprom` utility enables you to set configuration parameters to be used by the PROM when you perform a WAN boot. Setting network boot arguments in the PROM takes precedence over any default values. If you are using DHCP, these arguments also take precedence over configuration information that is provided by the DHCP server for the given parameter.

If you are manually configuring an Oracle Solaris system to boot from the network, you must provide thesystem with all of the necessary information for the system to boot.

Information that is required by the PROM includes the following:

- IP address of the booting client

Note - WAN boot does not include support for IPv6 addresses.

- Name of the boot file
- IP address of the stem that is providing the boot file image

In addition, you might be required to provide the subnet mask and IP address of the default router to be used.

The syntax to use for network booting is as follows:

`[`*protocol*`,] [`*key*=*value*`,]*`

protocol	Specifies the address discovery protocol that is to be used.
key=*value*	Specifies configuration parameters as attribute pairs.

The following table lists the configuration parameters that you can specify for the `network-boot-arguments` parameter.

Parameter	Description
`tftp-server`	IP address of the TFTP server
`file`	File to download by using TFTP or URL for WAN boot
`host-ip`	IP address of the boot client (in dotted-decimal notation)
`router-ip`	IP address of the default router (in dotted-decimal notation)
`subnet-mask`	Subnet mask (in dotted-decimal notation)
`client-id`	DHCP client identifier: this can be set to any unique value that the DHCP server allows. For AI clients, this value should be set to the hexidecimal hardware address of the client, preceded by the string `01` to indicate an ethernet network. For example, an Oracle Solaris client with the hexadecimal Ethernet address `8:0:20:94:12:1e` uses the client ID `0108002094121E`.
`hostname`	Host name to use in the DHCP transaction
`http-proxy`	HTTP proxy server specification (*IPADDR*[:*PORT*])

Parameter	Description
`tftp-retries`	Maximum number of TFTP retries
`dhcp-retries`	Maximum number of DHCP retries

▼ SPARC: How to Specify Network Boot Arguments in the OpenBoot PROM

Before You Begin Complete any preliminary tasks that are required for booting a system from the network. For more information, see "Requirements for Booting a System From the Network" on page 106.

1. **On the system that is to be booted from the network, Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Specify the appropriate values for the `network-boot-arguments` parameter.**

   ```
   # eeprom network-boot-arguments="protocol,hostname=hostname"
   ```

   For example, to use DHCP as the boot protocol and a host name of `mysystem.example.com`, you would set the values for the `network-boot-arguments` parameter as follows:

   ```
   # eeprom network-boot-arguments="DHCP,hostname=mysystem.example.com"
   ```

3. **Bring the system to the `ok` PROM prompt.**

   ```
   # init 0
   ```

4. **Boot the system from the network.**

   ```
   ok boot net
   ```

   Note - When you specify the `network-boot-arguments` parameter in this way, there is no need to specify the arguments from the PROM command line. Doing so will ignore any other values set for the `network-boot-arguments` parameter that you have might have specified.

SPARC: Setting Up an NVRAM Alias to Automatically Boot by Using DHCP

In Oracle Solaris 11, DHCP is the network configuration boot strategy that is used when booting from the network to install Oracle Solaris. To boot a system from the network with DHCP, a DHCP boot server must be available on your network.

You can specify that a SPARC based system boot by using the DHCP protocol when you run the `boot` command. Or, you can save the information across system reboots at the PROM level by setting up an NVRAM alias.

The following example uses the `nvalias` command to set up a network device alias for booting with DHCP by default:

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

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

Caution - Do not use the `nvalias` command to modify the `NVRAMRC` file, unless you are very familiar with the syntax of this command and also the `nvunalias` command.

SPARC: How to Boot a System From the Network

Before You Begin

- Perform any prerequisite tasks for setting up DHCP configuration. See "Requirements for Booting a System From the Network" on page 106.
- If you booting the system over the network to install Oracle Solaris, first download the AI client image and create an install service based on that image. For instructions, see Part 3, "Installing Using an Install Server," in *Installing Oracle Solaris 11.3 Systems*.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **If necessary, bring the system to the `ok` PROM prompt.**

   ```
   # init 0
   ```

3. **Boot the system from the network without using the "install "flag.**

   ```
   ok boot net:dhcp
   ```

   Note - If you have changed the PROM setting to boot with DHCP by default, you only have to specify `boot net`, as shown here:

   ```
   ok boot net
   ```

x86: Booting a System From the Network

The following information is provided in this section:

- "Requirements for Booting a System From the Network" on page 110
- "Where the GRUB 2 PXE Boot Image Is Installed" on page 112
- "Booting Systems With UEFI and BIOS Firmware From the Network" on page 112
- "How to Boot a System From the Network" on page 113

You might need to boot a system from the network for recovery purposes or to install Oracle Solaris. Any system can boot from the network, if a boot server is available. Any x86 based system whose network adapter firmware supports the Preboot eXecution Environment (PXE) specification can be used to boot Oracle Solaris. GRUB 2 is the PXE Network Bootstrap Program (NBP) that is then used to load the Oracle Solaris kernel and to proceed with the boot process.

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.

The DHCP server supplies the information that the boot client needs to configure its network interface. If you are setting up an AI server, that sstem can also be the DHCP server. Or, you can set up a separate DHCP server. For more information about DHCP, see *Working With DHCP in Oracle Solaris 11.3*.

x86: Requirements for Booting a System From the Network

Keep the following information in mind when booting an x86 based system from the network:

- The network configuration boot strategy that is used in Oracle Solaris is the Dynamic Host Configuration Protocol (DHCP).
- Network booting of Oracle Solaris uses PXE firmware interfaces, which provides a mechanism to load a boot program over the network, independent of data storage devices (like hard disks) and installed operating systems. This firmware is responsible for loading the boot program, which is a specially constructed GRUB 2 image named `pxegrub2` for systems with BIOS firmware and `grub2netx86.efi` for systems with 64-bit UEFI firmware. These files include the basic implementations of the Trivial File Transfer Protocol (TFTP), DHCP, User Datagram Protocol (UDP), Internet Protocol (IP), and a mini-driver that uses either the Universal Network Device Interface (UNDI) firmware interfaces (on BIOS

systems) or the Simple Network Protocol (SNP) interface (on UEFI systems), to transfer packets across the network.

- GRUB 2 uses a similar mechanism to GRUB Legacy PXE based network booting. The GRUB 2 PXE boot image contains the code and modules that are necessary for initializing GRUB, the file system modules that are required to boot from ZFS, as well a number of useful GRUB commands. Because loading modules over the network can add an unnecessary burden on network resources and can also expose the PXE boot process to failure where essential commands would not be available, modules that implement GRUB commands are built into the GRUB 2 PXE image, instead of remaining on the TFTP server.
- The GRUB 2 boot image includes an embedded `grub.cfg` file that implements the same search algorithm that is present in GRUB Legacy. This algorithm searches in several places on the TFTP server for the `grub.cfg` file to use for booting the operating system.
- Similar to GRUB Legacy, the GRUB 2 PXE boot image is installed in the TFTP server's root directory. The name of the PXE boot image depends on how the Automated Installer (AI) was configured. The appropriate DHCP `BootFile` macro contains the name of the PXE boot image, in accordance with the AI documentation.
- The `installadm` command has been modified to unconditionally copy the BIOS and UEFI PXE images to the proper location on the TFTP server. In addition, the DHCP server must also be able to return the appropriate `BootFile` macro when the appropriate system architecture tag is sent by the boot client so that systems that are running UEFI firmware are given the correct GRUB 2 (UEFI) `BootFile` option during the PXE boot. This information is provided when the DHCP server sends the `DHCPOFFER`.

  On an installed Oracle Solaris instance, the PXE boot images are stored in the `/boot/grub/pxegrub2` file (for a BIOS-targeted image), and in the `/boot/grub/grub2netx64.efi` file (for a 64-bit UEFI-targeted image).

  If you are booting a system from the network to install Oracle Solaris by using AI, see *Installing Oracle Solaris 11.3 Systems* for more information.

The DHCP server must be able to respond to the DHCP classes, `PXEClient` with the following information:

- IP address of the file server
- Name of the boot file, which is `pxegrub2` for systems with BIOS firmware and `grub2netx64.efi` for systems with UEFI firmware.

The sequence for performing a PXE boot from the network is as follows:

1. The firmware is configured to boot from a network interface.
2. The firmware sends a DHCP request.
3. The DHCP server replies with the server address and the name of the boot file.
4. The firmware downloads `pxegrub2` (or `grub2netx64.efi`) by using TFTP and then executes the GRUB 2 image.

5. The system downloads a GRUB configuration 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.

x86: Where the GRUB 2 PXE Boot Image Is Installed

Similar to GRUB Legacy, the GRUB 2 PXE boot image is installed in the root directory of the TFTP server. The name of the boot image depends on how AI was configured. The appropriate DHCP `BootFile` option contains the name of the PXE boot image. Both BIOS and UEFI firmware types are supported automatically, if the AI image is GRUB 2 based. No special arguments are required.

On an installed Oracle Solaris instance, the PXE boot images for both a BIOS-targeted and UEFI-targeted images are stored in boot/grub, in the root directory of the AI image, for example, `/export/auto_install/my_ai_service/boot/grub`.

This directory contains the following contents:

```
bash-4.1$ cd grub/
bash-4.1$ ls
grub_cfg_net i386-pc splash.jpg x86_64-efi
grub2netx64.efi pxegrub2 unicode.pf2
```

There are firmware-specific subdirectories for GRUB 2 modules that are in the `i386-pc` directory for systems with BIOS firmware, and the `x64_64-efi` directory for 64-bit UEFI systems. However, files in these directories are not used during a network boot (modules are built into the GRUB 2 images and are not transferred over TFTP).

Note - If you are using a DHCP server that is not managed by the `installadm` command, you will need to configure the DHCP server in accordance with how the `installadm` command normally configures an accessible DHCP server, which is to set up the `BootFile` based on the client architecture identifier. As an aid to the administrator, the `installadm` command prints out the client arch boot file paths that should be set for manually configured DHCP servers.

x86: Booting Systems With UEFI and BIOS Firmware From the Network

Bootable network adapters include firmware that complies with the PXE specification. When activated, the PXE firmware performs a DHCP exchange on the network and downloads the

BootFile macro that the DHCP server included in the DHCP response from the TFTP server that is also in the DHCP response. For Oracle Solaris, this BootFile macro, pxegrub2 (for systems with BIOS firmware), or grub2netx64.efi (for systems with 64-bit UEFI firmware), is GRUB 2. GRUB then proceeds to download the unix kernel, and the boot archive then loads both into memory. At which point, control is transferred to the Oracle Solaris kernel.

The network boot process on a system with UEFI firmware is very similar to the process on a system with BIOS firmware, with the exception that systems with UEFI firmware make a slightly different DHCP request, which provides the DHCP server with enough information to customize the BootFile macro that is returned for the UEFI system. Systems with UEFI firmware require UEFI boot applications, not BIOS-targeted boot programs, which would otherwise be returned as the BootFile macro from the DHCP server. After the UEFI boot application (GRUB) that is specified in the BootFile macro (grub2netx64.efi or the equivalent) is downloaded to the UEFI client, the boot loader (GRUB) is then executed. As with the BIOS network boot process, GRUB downloads the unix kernel and boot archive from the DHCP-specified TFTP server, then loads both into memory, and finally transfers control to the unix kernel.

▼ x86: How to Boot a System From the Network

Before You Begin

- Perform any prerequisite tasks for setting up DHCP configuration. See "Requirements for Booting a System From the Network" on page 110.
- 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 3, "Installing Using an Install Server," in *Installing Oracle Solaris 11.3 Systems*.

1. **Assume the root role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Perform a reboot of the system through the BIOS.**

   ```
   # reboot -p
   ```

   On systems that have the Fast Reboot feature enabled by default, the firmware is bypassed during a reboot unless the -p option is specified. Specifying this option enables a standard (slow reboot) of the, so you can access the system's firmware utility to specify a PXE boot and installation. For more information about Fast Reboot, see "Accelerating the Reboot Process" on page 100.

3. **Instruct the BIOS or UEFI firmware to boot from the network.**

- **If your system uses a specific keystroke sequence to boot from the network, type that sequence as soon as the BIOS or UEFI firmware screen is displayed.**

  For example, press F12 on a system with BIOS firmware to enter the setup utility.

- **If you need to manually modify the firmware settings to boot from the network, type the keystroke sequence to access the firmware setup utility. Then, modify the boot priority to boot from the network.**

4. **When the GRUB menu is displayed, select the network installation image that you want to install, the press Return to boot and install that image.**

   The system will proceed to boot and install the selected Oracle Solaris installation image from the network. The installation can take several minutes to complete. For information about performing AI installations, see Part 3, "Installing Using an Install Server," in *Installing Oracle Solaris 11.3 Systems*.

♦ ♦ ♦ C H A P T E R 6

Managing Systems with Boot Pools

This is a list of the topics related to managing a system that boots from boot pools.

Overview of Booting From Firmware-Inaccessible Storage Devices

Boot pools are created on systems, such as the SPARC M7 series servers, that can boot from firmware-inaccessible storage devices. These are devices that can not be identified by the firmware without additional step to the boot process. For instance, booting from an iSCSI device using IPoIB uses firmware-inaccessible storage. See "Changes to the Boot Process" on page 143 for more information.

Managing a Boot Pool, Boot Pool Datasets and Fallback Images

On a system that boots from a hardware-inaccessible storage device, you can manage the boot pool with the `bootadm boot-pool` command. Each option can include a `-P` *rpool* option to specify a specific root pool. For more information see the `bootadm`(1M) man page. The following options are available:

`add`	Adds new devices to the boot pool.

`list`	Displays information about configuration settings for the boot pools.
`remove`	Immediately removes a device from the boot pool.
`resync`	Resynchronizes the boot pool, and creates bootable datasets for particular boot environments.
`set`	Changes a boot pool parameter. Currently, you can change only the `eviction_algorithm` parameter.

By default, the boot pool datasets associated with the most recently booted BEs will remain in the boot pool. Datasets booted less recently will be removed if the boot pool runs out of space. If you want to make sure that a boot pool dataset is not automatically deleted, see "How to Retain the Ability to Boot a BE" on page 116. You can also change the default behavior of the system for all boot pool datasets. See "Changing the Eviction Behavior for All Boot Pool Datasets" on page 117 for more information.

Note - Only the datasets associated with a BE are removed by this process. The BE is not effected.

▼ How to Retain the Ability to Boot a BE

By default, datasets that are associated with the most recently booted BEs are retained if the boot pool runs out of memory. This procedure retains a boot pool dataset for a BE so that the BE will remain bootable.

1. **Identify the BE to retain.**

```
# beadm list
BE                 Flags Mountpoint Space   Policy Created
--                 ----- ---------- -----   ------ -------
BE1                -   -          6.13M   static 2014-10-09 17:21
BE2                -   -          52.86M  static 2015-01-03 16:22
BE3                NR   /          313.1M  static 2015-02-04 17:36
```

2. **Change policy for the BE.**

   In this example, the boot environment named `BE2` is being retained.

```
# beadm set-policy -p noevict BE2
```

3. **Verify that policy has changed.**

   The addition of the `noevict` policy to the `BE2` entry indicates that `BE2` will be retained.

```
# beadm list
BE               Flags Mountpoint Space   Policy         Created
--               ----- ---------- -----   ------         -------
BE1                    -          6.13M   static         2014-10-09 17:21
BE2              -     -          52.86M  noevict,static 2015-01-03 16:22
BE3              NR    /          313.1M  static         2015-02-04 17:36
```

Changing the Eviction Behavior for All Boot Pool Datasets

BE datasets are retained in order of the last boot, so datasets associated with recently booted BEs would be retained. By default, datasets that have not been booted recently will be removed if the boot pool gets full. To change this behavior, type the following command:

```
# bootadm boot-pool set eviction_algorithm=none
```

This will prevent a dataset from being removed from the boot pool. However, if the boot pool gets full, activities that add information to the boot pool will fail. The activities that add information to a boot pool include:

- Creating a new BE, which is often done by `pkg` operations
- Activating a BE whose dataset is not in the boot pool
- Changing the policy on a BE to `noevict`

▼ How to Make a BE Bootable

If the system is not booted from a firmware-inaccessible storage device, all BEs are bootable. However, when booting from an iSCSI boot device using IPoIB, for example on a SPARC M7 series server, it is possible that the dataset associated with a BE has been evicted from the boot pool. In this case the BE is no longer bootable. You can use the steps below to restore the dataset and make the BE bootable.

1. **Identify unbootable BE.**

   In this example, `BE1` includes a `!` in the list of flags, which means that `BE1` is not bootable. Note that the `N` flag indicates that BE3 is booted now, and the `R` flag indicates that `BE3` will be used during the next reboot.

```
# beadm list
BE               Flags Mountpoint Space   Policy Created
--               ----- ---------- -----   ------ -------
```

```
BE1                 !-    -          6.13M   static 2014-10-09 17:21
BE2                 -     -          52.86M  static 2015-01-03 16:22
BE3                 NR    /          313.1M  static 2015-02-04 17:36
```

2. **Activate the BE.**

   This command sets the BE to be active on reboot, and configures the boot pool dataset for booting.

```
# beadm activate BE1
```

3. **Verify that the BE is bootable.**

   In this example, `BE1` is now bootable because the ! is missing from the list of flags for that BE.

```
# beadm list
BE                  Flags Mountpoint Space   Policy         Created
--                  ----- ---------- -----   ------         -------
BE1                 R     -          6.13M   static         2014-10-09 17:21
BE2                 -     -          52.86M  noevict       2015-01-03 16:22
BE3                 N     /          313.1M  static         2015-02-04 17:36
```

4. **(Optional) Reset the BE to be booted on the next reboot.**

   If you do not want to boot from BE1 the next time the OS reboots, reset the active BE.

```
# beadm activate BE3
# beadm list
BE                  Flags Mountpoint Space   Policy         Created
--                  ----- ---------- -----   ------         -------
BE1                 -     -          6.13M   static         2014-10-09 17:21
BE2                 -     -          52.86M  noevict        2015-01-03 16:22
BE3                 NR    /          313.1M  static         2015-02-04 17:36
```

▼ How to Update the Fallback Image

This procedure applies to those systems that have service processors (SPs) with the fallback miniroot installed. Whenever you update to an SRU version on the host, you must also update the fallback image.

Note - The fallback image's `Readme` file also provides the same instructions for updating the fallback image. Make sure to refer to that document for additional information related to the image.

---

1. **Access your MOS account at `https://support.oracle.com`.**

On the support web page, search for Doc ID 2045311.1 (*Oracle Solaris 11.3 Support Repository Updates (SRU) Index*). This index lists the different SRU versions with links to their corresponding fallback boot images.

2. **Download the SRU version's fallback image.**

3. **At the ILOM prompt, check the version of the current image to verify that you do not have the latest version installed already.**

```
-> show /SP/firmware/host/miniroot version

  /SP/firmware/host/miniroot
    Properties:
        version = fallback-5.11-0.175.2.9.0.5.0
```

4. **At the ILOM prompt, update the fallback image.**

```
-> load -source  http://webserver.example.com/fallback/fallback.pkg
```

See Also To boot from the fallback image, see "Booting From a Fallback Image" on page 144.

OpenBoot Properties in Oracle Solaris 11.3

New OpenBoot properties provide information about the devices that can be used while the system is booting. These properties are automatically maintained.

These properties are read-only and are visible through `/chosen`.

`boot-pool-list`	Lists device paths to OpenBoot accessible storage devices that comprise a boot pool and are the devices that Oracle Solaris will use when booting
`tboot-list`	Lists storage devices that include fallback images

`os-root-device` Variable

The `os-root-device` NVRAM variable defines devices and root file systems for root pools. To view this variable use the `printenv` command at the OpenBoot prompt or the `eeprom` command at a shell prompt. This variable is automatically maintained and normally should not need manual intervention. You can use the following keywords to define root pools stored on a firmware-inaccessible storage device:

- `osroot-path`
- `osroot-type`
- `osroot-iscsi-initiator-id`
- `osroot-iscsi-target-ip`
- `osroot-iscsi-port`
- `osroot-iscsi-partition`
- `osroot-iscsi-lun`
- `osroot-iscsi-target-name`
- `osroot-subnet-mask`
- `osroot-host-ip`

EXAMPLE 46 Showing `os-root-device` Variable Keyword Data

This example shows data associated with the `os-root-device` ariable. The output is displayed with one keyword pair per line to make the text more readable. Normally the output is one long string.

```
# eeprom os-root-device
os-root-device=osroot-type=ZFS/iSCSI/IPv4/IPoIB;
osroot-iscsi-port=3260;
osroot-iscsi-target-ip=168.168.1.2;
osroot-iscsi-partition=a;
osroot-iscsi-initiator-id=iqn.1986-03.com.sun:01:0010e05db261.550b268b;
osroot-iscsi-lun=5;
osroot-iscsi-target-name=iqn.1986-03.com.sun:02:fb3685b9-883d-460a-b817-8ea0d8c023dc;
osroot-subnet-mask=255.255.255.0;osroot-host-ip=168.168.1.156;
osroot-path=/pci@314/pci@1/pciex15b3,1003@0:port=1,pkey=FFFF,protocol=ip
```

▼ How to Enable Normal Booting

After a system has been configured to boot from a firmware-inaccessible storage device, the stem will continue to boot in that configuration. If you want to boot using another device, change the `os-root-device` variable.

1. **Check the `os-root-device` property.**

   ```
   # eeprom os-root-device
   ```

2. **Clear the `os-root-device` property.**

   ```
   # eeprom os-root-device=
   ```

3. **Verify that the property has been changed.**

   ```
   # eeprom os-root-device
   ```

Next Steps If you need to reboot the system from a firmware-inaccessible storage device, boot the system from the boot pool. The `os-root-device` variable will automatically be updated during the boot process so that from then on the system will continue to boot using the iSCSI device.

♦ ♦ ♦ CHAPTER 7

Troubleshooting Booting a System

This chapter describes how to troubleshoot issues that prevent the system from booting or that require you to shut down and reboot the system for recovery purposes. Any information in this chapter that applies only to SPARC or x86 based systems is identified as such.

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

- "Managing the Oracle Solaris Boot Archives" on page 123
- "Shutting Down and Booting a System for Recovery Purposes" on page 127
- "Forcing a Crash Dump and Reboot of the System" on page 135
- "Booting a System With the Kernel Debugger (`kmdb`) Enabled" on page 138
- "Troubleshooting Issues With Fast Reboot" on page 140
- "Troubleshooting Issues With Booting and the Service Management Facility" on page 142

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

Managing the Oracle Solaris Boot Archives

The following information is provided in this section:

- "How to List Contents of the Boot Archive" on page 124
- "Managing the `boot-archive` SMF Service" on page 124
- "How to Clear a Failed Automatic Boot Archive Update by Manually Updating the Boot Archive" on page 125
- "How to Clear a Failed Automatic Boot Archive Update on a System That Does Not Support Fast Reboot" on page 126

For an overview of the Oracle Solaris boot archives, see "About Oracle Solaris Boot Archives" on page 15.

In addition to administering the boot loader on x86 platforms, the `bootadm` command is also used to perform the following tasks to maintain both the SPARC and x86 Oracle Solaris boot archives:

- 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 List Contents of the Boot Archive

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **To list the files and directories that are included in the boot archive, type:**

```
# bootadm list-archive
```

`list-archive`	Lists the files and directories that are included in the boot archive or archives.

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.

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 "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

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.

▼ How to Clear a Failed Automatic Boot Archive Update by Manually Updating the Boot Archive

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

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
```

x86: How to Clear a Failed Automatic Boot Archive Update on a System That Does Not Support Fast Reboot

During the process of rebooting a system, if the system does not support the Fast Reboot feature, the automatic update of the boot archive could fail. This problem might result in the system's inability to reboot from the same boot environment.

In this case, a warning similar to the following is displayed, and the system enters system maintenance mode:

```
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 `svc:/system/boot-config:default` SMF service contains the `auto-reboot-safe` property, which is set to `false` by default. Setting the property to `true` communicates that both the system's firmware and the default GRUB menu entry are set to boot from the current boot device. The value of this property can be changed so that a failed automatic boot archive update can be cleared, as described in the following procedure.

1. **Assume the `root` role.**

See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Reboot the system.**

   ```
   # reboot
   ```

3. **If the active BIOS or UEFI 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
      ```

Shutting Down and Booting a System for Recovery Purposes

The following procedures are provided in this section:

- "How to Stop a System for Recovery Purposes" on page 128
- "How to Stop and Reboot a System for Recovery Purposes" on page 130
- "How to Boot to a Single-User State to Resolve a Bad `root` Shell or Password Problem" on page 130
- "How to Boot From Media to Resolve an Unknown `root` Password" on page 131
- "How to Boot From Media to Resolve a Problem With the GRUB Configuration That Prevents the System From Booting" on page 133

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.

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 to a single-user state 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 AI 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.
- **x86 only:** Resolve a boot configuration problem by importing the root pool. If a problem with the 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.

▼ SPARC: How to Stop a System for Recovery Purposes

1. **Bring the system to `ok` PROM prompt by using the `shutdown` or `init 0` command.**

2. **Synchronize the file systems.**

```
ok sync
```

3. **Type the appropriate `boot` command to start the boot process.**

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

4. **Verify that the system was booted to the specified run level.**

```
# who -r
 .       run-level s  May  2 07:39     3      0  S
```

5. **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.**

Example 47 Powering Off a System

If you are running Oracle Solaris 11 on a sstem that uses a service processor, after shutting down the system, you must switch from the system console prompt to the service processor prompt. From there, you can stop the service processor, as shown in this example:

```
# shutdown -g0 -i0 -y
# svc.startd: The system is coming down. Please wait.
svc.startd: 91 system services are now being stopped.
Jun 12 19:46:57 wgs41-58 syslogd: going down on signal 15
svc.stard: The system is down.
syncing file systems...done
Program terminated
r)eboot o)k prompt, h)alt?
# o

ok #.
->

-> stop /SYS
Are you sure you want to stop /SYS (y/n)? y
Stopping /SYS

->
```

If you need to perform an immediate shutdown, use the `stop -force -script /SYS` command. Before you type this command, ensure that all data is saved.

Example 48 Powering On a System

The following example shows how to power on the system that uses a service processor. You must first be logged in to Oracle ILOM. See `https://docs.oracle.com/cd/E19166-01/E20792/z40002fe1296006.html`.

If you have a modular system, make sure that you are logged into the desired server module.

```
-> start /SYS
Are you sure you want to start /SYS (y/n) ? y
Starting /SYS

->
```

If you do not want to be prompted for a confirmation, use the `start -script /SYS` command.

▼ x86: How to Stop and Reboot a System for Recovery Purposes

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **If the keyboard and mouse are functional, type `init 0` to stop the system.**

   ```
   # init 0
   ```

3. **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 to a Single-User State to Resolve a Bad `root` Shell or Password Problem

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Depending on the platform, do one of the following:**

   - **For SPARC platforms:**

     a. **Bring the system to the `ok` PROM prompt.**

        ```
        # init 0
        ```

     b. **Boot the system to a single-user state.**

        ```
        ok boot -s
        ```

   - **For x86 platforms:**

a. **Reboot a running system with the `-p` option of the `reboot` command.**

   ```
   # reboot -p
   ```

b. **When the GRUB menu is displayed, select the appropriate boot entry, then type `e` to edit that entry.**

c. **Using the arrow keys, navigate to the `$multiboot` line, then type `-s` at the end of the line.**

- **To exit the GRUB edit menu and boot the entry you just edited, press Control-X. If you have a system with UEFI firmware, and you are not using a serial console, pressing F10 also boots the entry.**

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

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

4. **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 an unknown `root` password or similar problem. 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 "Replacing Disks in a ZFS Root Pool" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

1. **Boot from the Oracle Solaris media by using one of the following options:**

   - **SPARC: Text installation – Boot from the installation media or from the network, then select the Shell option (option 3) from the text installation screen.**
   - **SPARC: Automated installation – Use the following command to boot directly from an installation menu that allows you to exit to a shell:**

     ```
     ok boot net:dhcp
     ```

- **x86: Live Media – Boot from the installation media, and use a GNOME terminal for the recovery procedure.**
- **x86: Text installation – From the GRUB menu, select the Text Installer and command line boot entry, then select the Shell option (option 3) from the text installation screen.**
- **x86: Automated installation – Boot from an AI server on the network. This method requires a PXE boot. Select the Text Installer and command line entry from the GRUB menu. Then, select the Shell option (option 3) from the text installation screen.**

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 the mount point `/a`.**

   `# beadm mount` *solaris-instance*`|`*be-name* `/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 to a single-user state, 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.**

See Also If there is a problem with the GRUB configuration that requires you to boot the system from media, follow the same steps for x86 platforms that are in this procedure> However, you

▼ x86: How to Boot From Media to Resolve a Problem With the GRUB Configuration That Prevents the System From Booting

If your x86 based system will not boot, the problem might be caused by a damaged boot loader or a missing or corrupt GRUB menu. Use the following procedure in these types of situations.

Note - This procedure does *not* require you to mount the boot environment.

If you need to recover a root pool or root pool snapshot, see "Replacing Disks in a ZFS Root Pool" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

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 Shell option (option 3) from the text installation screen.**

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

2. **Import the root pool.**

```
# zpool import -f rpool
```

3. **To resolve a GRUB configuration issue, do one of the following:**

   - **If the system will not boot, but no error messages are displayed, the boot loader might be damaged. To resolve the problem, see "Installing GRUB 2 by Using the `bootadm install-bootloader` Command" on page 55.**

   - **If the GRUB menu is missing, a "cannot open grub.cfg" error message is displayed at boot time. To resolve this problem, see "How to Manually Regenerate the GRUB Menu" on page 34.**

   - **If the GRUB menu has become corrupted, other error messages might be displayed as the system attempts to parse the GRUB menu at boot time. See also "How to Manually Regenerate the GRUB Menu" on page 34.**

4. **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

The following procedures are provided in this section:

- SPARC: How to Force a Crash Dump and Reboot of the System
- x86: How to Force 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 "Configuring Your System for Crash Dumps" in *Troubleshooting System Administration Issues in Oracle Solaris 11.3*.

▼ SPARC: How to Force a Crash Dump and Reboot of the System

Use this procedure to force a crash dump of a SPARC based system. The example that follows this procedure shows how to use the `halt -d` command to force a crash dump of the system. You will need to manually reboot the system after running this command.

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

2. **Synchronize the file systems and write the crash dump.**

   ```
   > n
   ok sync
   ```

   After the crash dump is written to disk, the system will continue to reboot.

3. **Verify that the system boots to run level 3.**

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

   hostname `console login:`

Example 49 SPARC: Forcing a Crash Dump and Reboot of a System by Using the `halt -d` Command

This example shows how to force a crash dump and reboot of a SPARC based system by using the `halt -d` command.

```
# halt -d
Jul 21 14:13:37 jupiter halt: halted by root
```

```
panic[cpu0]/thread=30001193b20: forced crash dump initiated at user request

000002a1008f7860 genunix:kadmin+438 (b4, 0, 0, 0, 5, 0)
  %l0-3: 0000000000000000 0000000000000000 0000000000000004 0000000000000004
  %l4-7: 00000000000003cc 0000000000000010 0000000000000004 0000000000000004
000002a1008f7920 genunix:uadmin+110 (5, 0, 0, 6d7000, ff00, 4)
  %l0-3: 0000030002216938 0000000000000000 0000000000000001 0000004237922872
  %l4-7: 000000423791e770 0000000000004102 0000030000449308 0000000000000005

syncing file systems... 1 1 done
dumping to /dev/dsk/c0t0d0s1, offset 107413504, content: kernel
100% done: 5339 pages dumped, compression ratio 2.68, dump succeeded
Program terminated
ok boot
Resetting ...

.
.
Rebooting with command: boot
Boot device: /pci@1f,0/pci@1,1/ide@3/disk@0,0:a
File and args: kernel/sparcv9/unix
configuring IPv4 interfaces: hme0.
add net default: gateway 198.51.100.240
Hostname: jupiter
The system is coming up.  Please wait.
NIS domain name is example.com
.
.
.
System dump time: Wed Jul 21 14:13:41 2013
Jul 21 14:15:23 jupiter savecore: saving system crash dump
in /var/crash/jupiter/*.0
Constructing namelist /var/crash/jupiter/unix.0
Constructing corefile /var/crash/jupiter/vmcore.0
100% done: 5339 of 5339 pages saved
.
.
.
```

▼ x86: How to Force a Crash Dump and Reboot of the System

If you cannot use the `reboot -d` or the `halt -d` command, you can use the kernel debugger (`kmdb`) to force a crash dump. The kernel debugger must have been loaded, either at boot 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. So, first exit any window system.

---

1. **Access the kernel debugger.**

   The method that is 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 50 x86: Forcing a Crash Dump and Reboot of the System by Using the `halt -d` Command

This example shows how to force a crash dump and reboot of an x86 based system by using the `halt -d` command.

```
# 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
```

Booting a System With the Kernel Debugger (kmdb) Enabled

The following procedures are provided in this section:

- "How to Boot a System With the Kernel Debugger (kmdb) Enabled" on page 138
- "How to Boot a System With the Kernel Debugger (kmdb) Enabled" on page 139

If you need to troubleshoot system problems, running a system under the kernel debugger can be very helpful. The kernel debugger can help you investigate system hangs. For example, if you are running the kernel while the kernel debugger is active, and you experience a hang, you might be able to break into the debugger to examine the system state. Also, if the system panics, the panic can be examined before the system is rebooted. In this way, you can get an idea of which section of code might be causing the problem.

The following procedures describe the basic steps for troubleshooting system problems by booting with the kernel debugger enabled.

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

This procedure shows how to load the kernel debugger (kmdb) on a SPARC based system.

Note - Use the `reboot` command and the `halt` command with the `-d` option if you do not have time to debug the system interactively. Running the `halt` command with the `-d` option requires a manual reboot of the system afterward. However, if you use the `reboot` command, the system boots automatically. See the reboot(1M) for more information.

---

1. **Halt the system, causing it to display the `ok` prompt.**

   To halt the system cleanly, use the `halt` command.

2. **Type `boot -k` to request the loading of the kernel debugger. Press return.**

3. **Access the kernel debugger.**

   The method used to enter the debugger depends on the type of console that is used to access the system:

   - **If you are using a locally attached keyboard, press Stop-A or L1–A, depending on the type of keyboard.**

   - **If you are using a serial console, send a break by using the method that is appropriate for your type of serial console.**

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

```
Rebooting with command: kadb
Boot device: /iommu/sbus/espdma@4,800000/esp@4,8800000/sd@3,0
.
.
.
```

Example 51 SPARC: Booting a System With the Kernel Debugger (kmdb) Enabled

The following example shows how to boot a SPARC based system with the kernel debugger (kmdb) enabled.

```
ok boot -k
Resetting...

Executing last command: boot kmdb -d
Boot device: /pci@1f,0/ide@d/disk@0,0:a File and args: kmdb -d
Loading kmdb...
```

▼ x86: How to Boot a System With the Kernel Debugger (kmdb) Enabled

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

1. **Boot the system.**

2. **When the GRUB menu is displayed, type `e` to access the GRUB edit menu.**

3. **Use the arrow keys to select the `$multiboot` line.**

4. **In the GRUB edit menu, type `-k` at the end of the `$multiboot` line.**

   To direct the system to stop (break) in the debugger before the kernel executes, include `-d` option with the `-k` option.

5. **To exit the GRUB edit menu and boot the entry you just edited, press Control-X. If you have a system with UEFI firmware, and you are not using a serial console, pressing F10 also boots the entry.**

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

6. **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 (`kmdb`) 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`, see the `kmdb(1)` man page.

x86: Troubleshooting Issues With Fast Reboot

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.

The following information is provided in this section:

- “Debugging Early Panics That Might Occur” on page 141
- “Conditions Under Which Fast Reboot Might Not Work” on page 141

If you need to manually update the Oracle Solaris boot archive on an x86 based system that does not support the Fast Reboot feature, see “How to Clear a Failed Automatic Boot Archive Update on a System That Does Not Support Fast Reboot” on page 126.

x86: Debugging Early Panics That Might Occur

Because the `boot-config` service has dependencies on the multiuser milestone, users who need to debug early panics can patch a global variable, `fastreboot_onpanic` in the `/etc/system` file, as shown in the following example:

```
# echo "set fastreboot_onpanic=1" >> /etc/system
# echo "fastreboot_onpanic/W" | mdb -kw
```

x86: Conditions Under Which Fast Reboot Might Not Work

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

- GRUB configuration 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 driver for the network interface card (NIC) does not implement the `quiesce` function, you can attempt to unplumb the interface first, then retry 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)

Troubleshooting Issues With Booting and the Service Management Facility

The following are issues you might encounter when booting a system:

- Services do not start at boot time.

  A system might hang during boot time, if there are problems starting any Service Management Facility (SMF) services. To troubleshoot this type of issue, you can boot the system without starting any services. For more information, see "How to Investigate Problems Starting Services at System Boot" in *Managing System Services in Oracle Solaris 11.3*
- `system/filesystem/local:default` SMF service fails during boot.

  Local file systems that are not required to boot the system are mounted by the `svc:/system/filesystem/local:default` service. When any of those file systems cannot be mounted, the service enters a maintenance state. System startup continues, and any services that do not depend on `filesystem/local` are started. Subsequently, any services that require `filesystem/local` to be online before starting through dependencies are not started. The workaround for this issue is to change the configuration of your system so that a `sulogin` prompt is displayed immediately after the service fails instead of allowing the system startup to continue. For more information, see "How to Force Single-User Login if the Local File System Service Fails During Boot" in *Managing System Services in Oracle Solaris 11.3*.

Problems Booting After an Installation

After installing onto a GPT labeled disk that had previously used MBR partitioning, sometimes the x86 firmware can not boot from the disk. This happens if your BIOS implementation requires that at least one hard disk have at least one MBR partition that's marked as bootable to boot in BIOS mode. The problem exists because UEFI specification forbids the EFI protective MBR partition from being marked as active. To fix this situation, use the `fdisk` command to set the partition's status to be active before attempting to boot.

♦ ♦ ♦ A P P E N D I X A

Oracle Solaris Boot Process

This appendix includes the following topics:

- "Changes to the Boot Process" on page 143
- "Booting From a Fallback Image" on page 144

Changes to the Boot Process

The boot process has been enhanced to allow different types of devices to be used. In prior releases, the boot process was as follows:

1. The boot process starts with bootstrapping, which are automatic steps that load the boot program.
2. The boot archive is loaded.
3. The root pool is imported, and the root file system is mounted.
4. `init` is run, which in turn starts `svc.startd`, which starts all of the services.

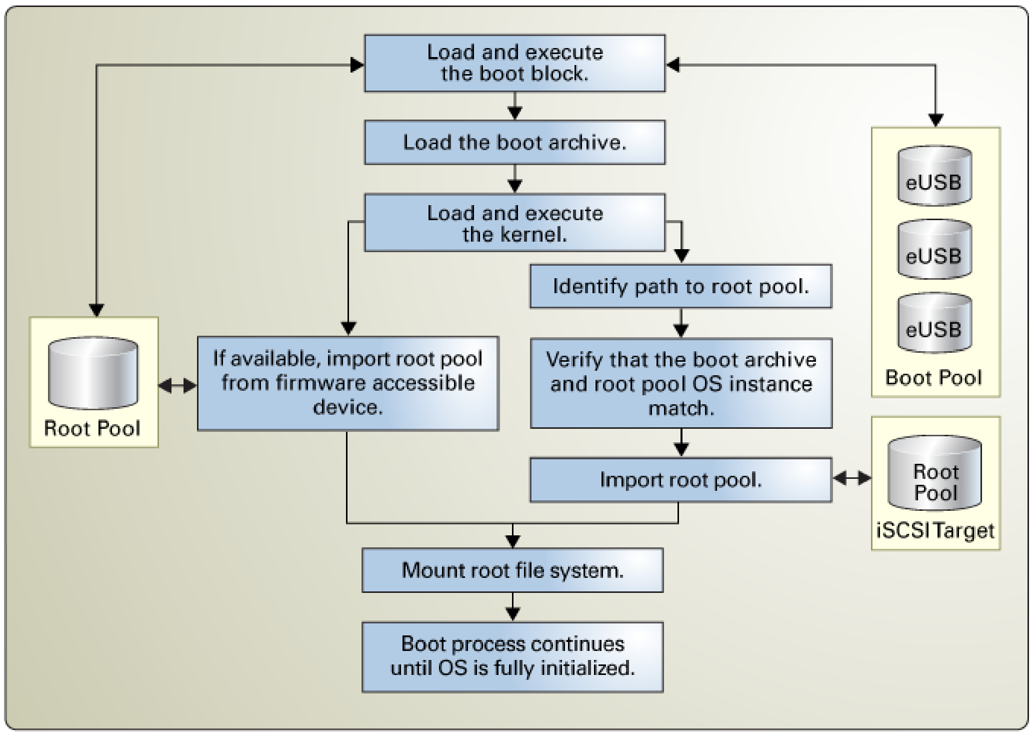
On systems that are booting from firmware-inaccessible storage devices with the separation of the boot pool from the root pool, the boot process after the boot program and the boot archive are loaded is as follows:

1. The boot pool is identified and imported.
2. The path to the root pool is identified. The software confirms that the OS instance in the boot archive and the root file system match.

The following figure illustrates this series of steps.

Note - On the SPARC M7 series servers, the boot pool consists of one or more eUSB devices. On the SPARC T7 series servers, the boot pool consists of one eUSB device.

FIGURE 1 Oracle Solaris Boot Process


Load and execute the boot block.
Load the boot archive.
Load and execute the kernel.
Identify path to root pool.
Verify that the boot archive and root pool OS instance match.
Import root pool.
If available, import root pool from firmware accessible device.
Root Pool
eUSB
eUSB
eUSB
Boot Pool
Root Pool
iSCSI Target
Mount root file system.
Boot process continues until OS is fully initialized.

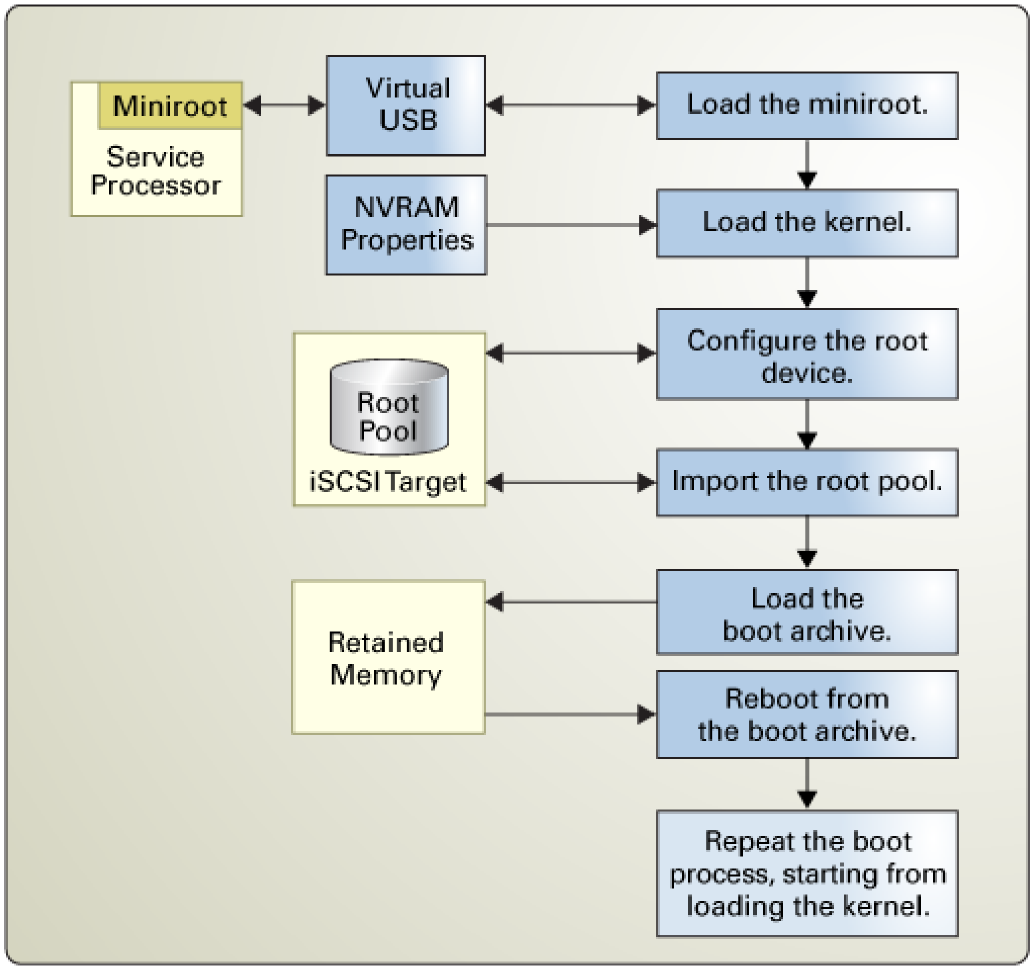
Booting From a Fallback Image

In addition to booting from firmware-inaccessible storage devices, some new servers provide the ability to use a fallback image if the boot pool is unavailable. The fallback process depends on a small version of the Oracle Solaris OS, called a miniroot, that is stored on one or more service processors. The servers are preconfigured with fallback images. If you need to update the fallback image, see "How to Update the Fallback Image" on page 118.

During the fallback boot process, once the kernel from the miniroot is loaded, the root device is configured. This configuration includes a pointer to the location of the root pool on the iSCS device. Next, the root pool is imported and the boot archive is loaded into retained memory. The system is then rebooted from the boot archive and the boot process repeats, starting with loading the kernel. However the kernel in the second boot is from the boot archive, not from the miniroot in the fallback image.

The following figure shows this process.

FIGURE 2 Fallback Boot Process



Index