Halting and Booting

Student Guide

Sun

Sun Microsystems Computer Corporation
Technical Education Services
MS UMIL07-14
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part Number SA-285
Revision B, July 1993

2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Contents

The Boot Process

1

Objectives

Upon completion of this lesson, you will be able to:

- Sequence the phases of the boot process.
- Recall the functionality available at each of the eight system run levels.
- Describe the two main roles of the `/sbin/init` program.
- Describe a run control script.
- State the advantage of having the actual run control files in the `/etc/init.d` directory.
- Describe how to add startup files for additional system services.

References

SunOS 5.1 Routine System Administration Guide, Chapter 13, "Understanding the Boot Process"

Introduction

This lesson describes the concepts and terms used in the Solaris® 2.*x* boot procedure.

Identifying the steps in the boot process is a key skill for system administrators who must troubleshoot systems that do not boot successfully.

The commands used to halt and boot a Sun™ workstation are described in another lesson in this module.

Boot Terminology

- The Boot PROM

  Each Sun system has a programmable read-only memory (PROM) chip with a monitor program that controls the system operation before the kernel is loaded.

  The boot PROM has built-in device drivers which allow it to identify and access devices that are attached to the system.

- Boot Blocks

  A system's primary boot program, or *boot blocks,* is used during the bootstrap process. This is when a boot program is used to load a secondary boot program into memory, so the operating system can *"bring itself up by its bootstraps."* The boot blocks are located at physical sectors 1-15 on the first partition of the boot device.

  The `installboot` command is used to install the boot blocks on a newly created / (`root`) partition. (This procedure is covered in another module.)

- Run Levels

  A run level is a system's software configuration with a specific set of system services (or processes).

- Run Control Scripts

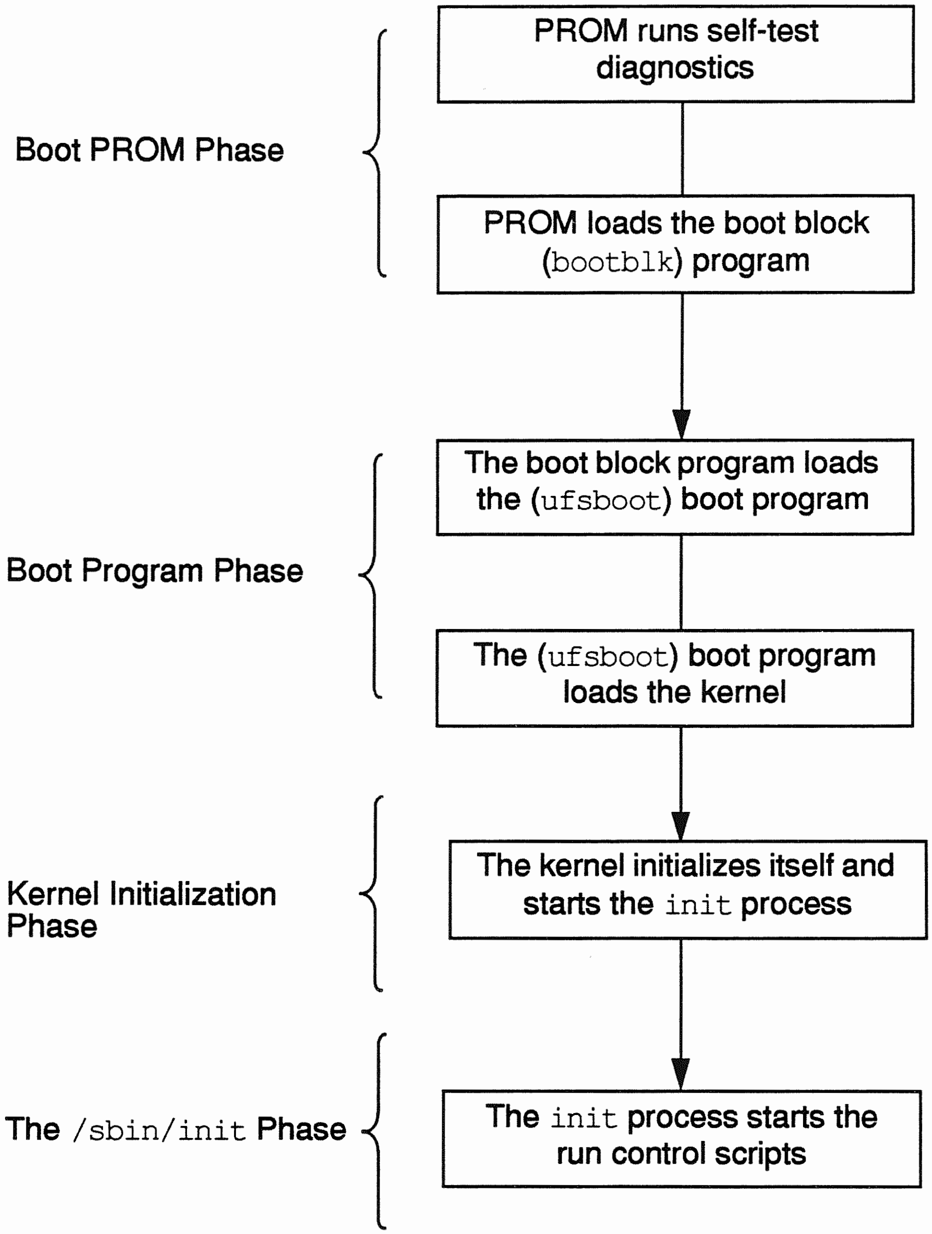
  The Solaris 2.*x* system provides a series of run control (`rc`) scripts that check and mount the file systems, start and/or stop processes such as the print daemon, the NFS client-server daemons, the sendmail daemon, and perform housekeeping tasks.

  The run control scripts are also used to build a specific run level.

- Loadable Kernel Modules

  The Solaris 2.*x* computing environment consists of a small static core (or kernel) and a set of dynamically loadable modules, which are programs that will be loaded into the kernel as they are executed. The loadable kernel modules include device drivers, file systems, and system calls.

System Boot Procedure



Boot PROM Phase
PROM runs self-test diagnostics
PROM loads the boot block (`bootblk`) program
Boot Program Phase
The boot block program loads the (`ufsboot`) boot program
The (`ufsboot`) boot program loads the kernel
Kernel Initialization Phase
The kernel initializes itself and starts the `init` process
The `/sbin/init` Phase
The `init` process starts the run control scripts

System Boot Procedure

Boot PROM Phase

The boot PROM performs the following steps during the first part of the boot sequence.

1. Displays system identification banner

   The model type, keyboard type, host ID, PROM revision number, and Ethernet address are displayed.

2. Runs self-test diagnostics

   The boot PROM runs a self-test routine to verify the system's hardware and memory. It then begins its boot sequence upon successful completion of the self-test diagnostics.

3. Finds the boot program from the default boot device programmed into the PROM.

   The boot PROM reads a system's primary boot program called `bootblk` (located at sectors 1-15) that contains a `ufs` file system reader.

   The boot PROM can be programmed to try an alternate boot device. (This is illustrated later in this module.)

4. Loads the boot program

   The file system reader opens the boot device, finds the secondary boot program called `/ufsboot`, and loads it into memory.

Boot Program Phase

At this point, the `/ufsboot` program takes over.

5. After the `/ufsboot` program is loaded, the boot PROM loads the kernel (`/kernel/unix`).

System Boot Procedure

Kernel Initialization Phase

6. The kernel begins loading modules using the /ufsboot program to read the files as soon as it initializes itself.

   When the kernel has read in the modules needed to mount the root partition, it unmaps the /ufsboot program from memory and continues initializing the system using its own resources.

The /sbin/init Phase

7. The kernel creates a user process and starts the /sbin/init program. The /sbin/init program starts processes by using information in the /etc/inittab file.

   The init process executes an rc script, or scripts, that executes a series of other scripts. These scripts (/sbin/rc*) check file systems and mount file systems, start various processes, and perform housekeeping tasks.

System Run Levels

By default, the system is brought to run level 3 (or full multiuser state) after a system has booted successfully.

The Solaris 2.*x* environment provides several run levels (or operational states) that determine various modes of system operation.

Run Level	Function
0	PROM monitor level
1 Avoid	Administrative state (single-user state with some file systems mounted and user logins disabled)
2	Multiuser level (with no resources shared)
3	Multiuser level (with resources shared)
4	Not currently used
5	Halt and interactive boot (`boot -a`)
6	Reboot to default run level 3
S, s	Single-user state with some file systems mounted and user logins disabled

Only denies new requests.

Single-user state means the virtual console terminal is assigned to the system console and is available to the superuser. You must know the `root` password to get to single-user mode, and no users can log in.

Multiuser state means all defined terminal and daemon processes are running.

The /sbin/init Process

The /sbin/init program has two main roles.

- Creates processes, which has the effect of bringing the system up to the default run level.
- Controls transitions between run states by re-reading the /etc/inittab file.

The /etc/inittab File

The inittab entries tell the init process what processes to create for each run level and what actions to perform.

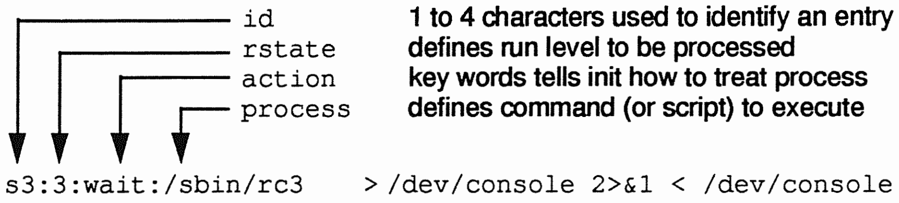
The /etc/inittab file defines three main items for the /sbin/init process.

- The system's default run level
- What processes to start, monitor, and restart if they die
- What actions to be taken when the system enters a new run level

newfs -N
will list all backup superblock locations

The /etc/inittab File Format

The /etc/inittab file contains entries with four fields.



id	1 to 4 characters used to identify an entry
rstate	defines run level to be processed
action	key words tells init how to treat process
process	defines command (or script) to execute

```
s3:3:wait:/sbin/rc3    > /dev/console 2>&1 < /dev/console
```

The maximum size of an entry is 512 characters.

The action keywords are:

initdefault
Identifies the default run level, which is 3 by default on Sun systems.

respawn Starts the process and restarts it when it dies.

powerfail
Starts the process when the init receives a power fail signal.

sysinit Starts the process before trying to access the console and waits for its completion before continuing.

wait Starts the process and waits for it to finish before going on to the next entry for this run state.

Several more are listed in the inittab man page.

The /etc/inittab File

This file is part of the Solaris 2.*x* system software. There is no need to manually edit this file.

File format:

Always Exe.

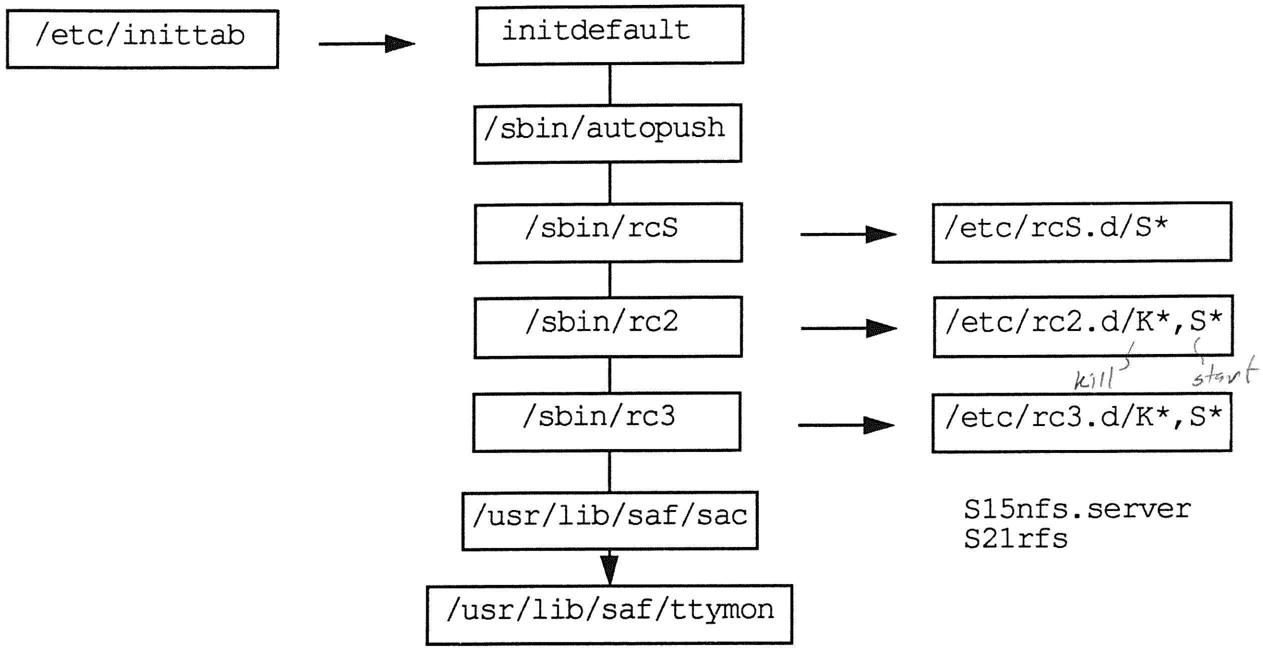
```
ap::sysinit:/sbin/autopush -f /etc/iu.ap
fs::sysinit:/sbin/rcS>/dev/console 2>&1 </dev/console
is:3:initdefault:
p3:s1234:powerfail:/sbin/shutdown -y -i0 -g0 >/dev/console 2>&1
s0:0:wait:/sbin/rc0 off>/dev/console 2>&1 </dev/console
s1:1:wait:/sbin/shutdown -y -iS -g0>/dev/console 2>&1 </dev/console
s2:23:wait:/sbin/rc2>/dev/console 2>&1 </dev/console
s3:3:wait:/sbin/rc3>/dev/console 2>&1 </dev/console
s5:5:wait:/sbin/rc5 ask>/dev/console 2>&1 </dev/console
s6:6:wait:/sbin/rc6 reboot>/dev/console 2>&1 </dev/console
of:0:wait:/sbin/uadmin 2 0>/dev/console 2>&1 </dev/console
fw:5:wait:/sbin/uadmin 2 2>/dev/console 2>&1 </dev/console
RB:6:wait:/sbin/sh -c ‘echo “\nThe system is being restarted.”’
>/dev/console 2>&1
rb:6:wait:/sbin/uadmin 2 1>/dev/console 2>&1 </dev/console
sc:234:respawn:/usr/lib/saf/sac -t 300
co:234:respawn:/usr/lib/saf/ttymon -g -h -p "`uname -n` console login: "
-T sun -d /dev/console -l console -m ldterm,ttcompat
```

is -> initialization state (default)

Do not edit this file.

The Run Control Scripts

Boot Sequence



1. The `init` process looks in the `inittab` file for the `initdefault` entry, which is set to run level 3.

2. The `init` process executes the commands (or scripts) for entries that have `sysinit` in the action fields, such as `/sbin/autopush` and `/sbin/rcS`. Then, it executes the scripts for any entry that has a `3` in the rstate field, such as the `/sbin/rc2`, `/sbin/rc3`, `/usr/lib/saf/sac`, and `/usr/lib/saf/ttymon` executables.

   The `sac` and `ttymon` processes are used to provide login services for the console and other TTY devices. (This topic is covered in another module.)

3. The `/sbin/rc2` and `/sbin/rc3` scripts invoke scripts under the `/etc/rc2.d` and `/etc/rc3.d` directories, respectively, that begin with the letters `K` and `S`.

The Run Control Scripts

There is at least one entry in the inittab file for each run level. A script is executed for each run level. The name of the script is /sbin/rc# where # equals run level number. (See A in the diagram on the facing page.)

Each run level script in turn executes files found under the /etc/rc#.d directory, where # equals run level number. (See B in the diagram on the facing page.)

Run Control Files

The files (or scripts) in these directories begin with either K or S. All K* files are run first and all S* files are run next. Files that start with K are used to kill processes. Files that start with S are used to start processes. Both types of files are run in alphanumeric order.

The S(tart) Files

The /etc/rc3.d directory contains only three files:

```
README        S15nfs.server      S21rfs
```

Because the scripts are run in alphanumeric order, the S10syslog file is run before the S15nfs.server file and so on.

The only differences between run level 2 and 3 are that file resources are shared, and the syslogd daemon is restarted, at run level 3. There are no K* files in the /etc/rc3.d directory.

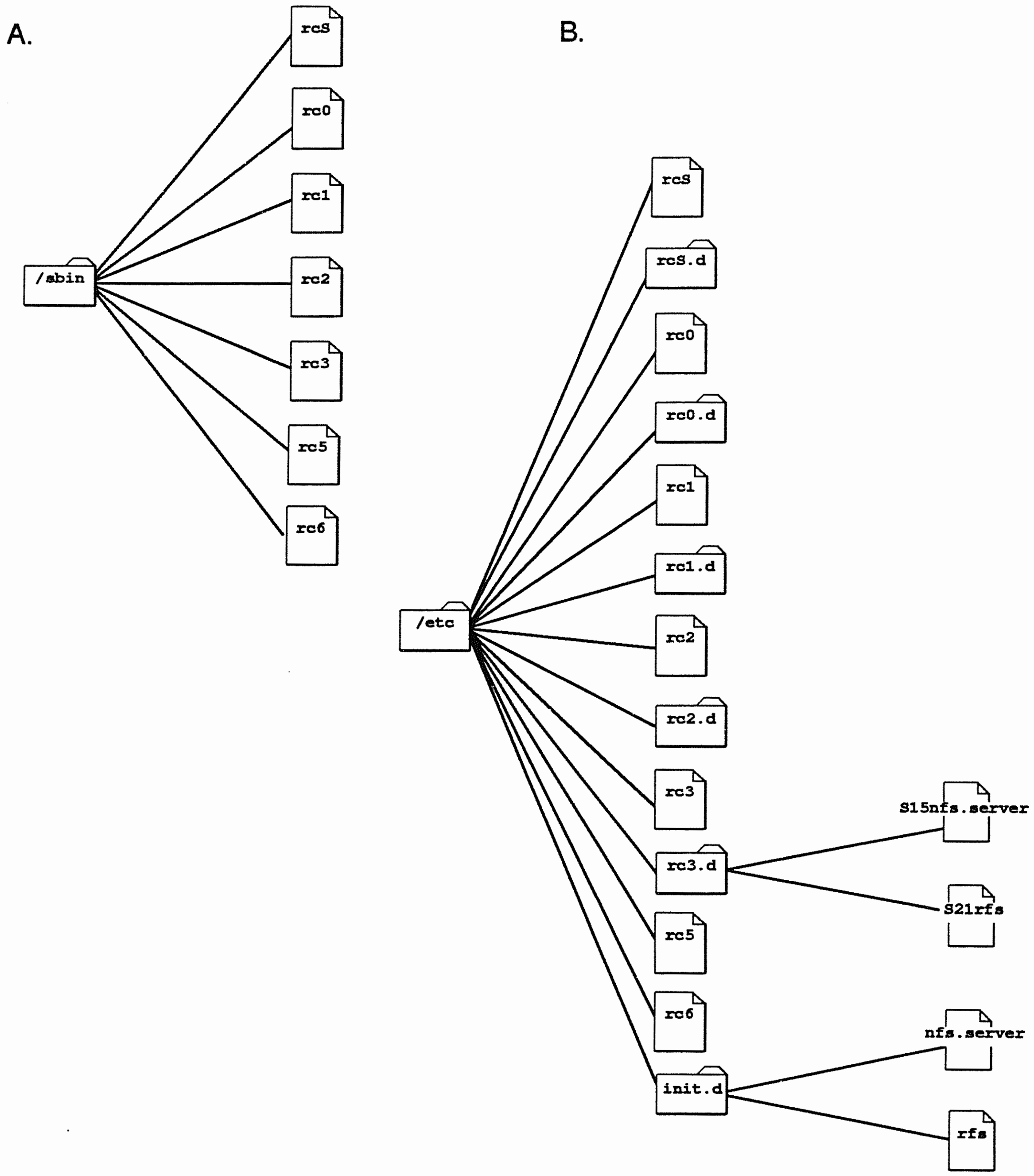
The K(ill) Files

Changing to run level 0 means the system is shutting down, so all processes must be killed. This means /etc/rc0.d contains only K* files.

```
K00ANNOUNCE   K55syslog     K65rfs          K70cron
K20lp         K57sendmail   K66nfs.server   K75nfs.client
K50fumounts   K60rumounts   K68rpc
```

sbin => Standalone Binary

Run Control Script File Locations



The /etc/init.d Directory

The rc scripts are contained in the /sbin directory, with symbolic links in the /etc directory.

The run control files that are used to start up or kill specific processes are actually stored in the /etc/init.d directory, with hard links in the /etc/rc*.d directories.

The advantage of having the actual run control files in the /etc/init.d directory is that you can start and stop individual services or processes without changing run levels.

For example, you can stop the lp print service using the following syntax:

```
# /etc/init.d/lp stop
```

Adding Run Control Files

Follow these steps if you need to add startup files for additional system services.

1. Read the instructions in /etc/init.d/README file for an example of adding a startup file.

2. Assign an appropriate sequence number for the startup file in the /etc/rc*.d directory, so that it doesn't conflict with an existing sequence number. Also, make sure the service you are starting comes after a service that it needs to start successfully.

3. Place the startup file in the /etc/init.d directory and create a link in the appropriate /etc/rc*.d directory, based on what run level you want the service to be started in.

```
# cd /etc/rc3.d
# ln /etc/init.d/myfs S22myfs
```

If you want to test a startup file and need to disable another one, rename the file to be disabled by using a dot (.) at the beginning of the name. File names that begin with dot (.) are not executed.

```
# mv S22myfs .S22myfs
```

Note: If you copy a script by adding a suffix to the original file name, both files will be executed.

```
# cp S22myfs S22myfs.old
```

Modifying Run Control Files

Sometimes it may be necessary to modify a startup file to customize the start up of some system service.

Example:

Add the `-Y` option to the `rpc.nisd` (NIS+) daemon to provide Network Information Service (NIS) compatibility.

1. Copy the `/etc/init.d/rpc` script.

   ```
   # cp /etc/init.d/rpc /etc/init.d/rpc.gen
   ```

2. Edit the `/etc/init.d/rpc` script and remove the comment symbol (#) from following line

   ```
   #                                    EMULYP="-Y"
   ```

3. Restart the `rpc` service:

   ```
   # /etc/init.d/rpc stop
   # /etc/init.d/rpc start
   ```

Run Control Script Summary

The following table summarizes the purpose of each script.

Script name	Purpose
/sbin/rcS	This script is used to set up minimal network plumbing for diskless/dataless client support, and check and mount / (root) and /usr file systems during all run level changes. Device entries are created, if necessary.
/sbin/rc0	This script is run to bring the system down to the PROM level. System services and all running processes are stopped. All file systems are unmounted.
/sbin/rc1	This script is run to bring the system down to run level 1. All system services and daemons and running processes are stopped. Most file systems are unmounted except / (root), /usr, /proc, and /dev/fd.
/sbin/rc2	This script is run to bring the system to run level 2—multiuser state. All file systems are mounted and all network services are configured except for sharing file systems.
/sbin/rc3	This script is run to bring the system to run level 3—multiuser state with shared file resources. The syslogd daemon is started, and nfs and rfs server process are started.
/sbin/rc5	This script is run to halt the machine and perform an interactive boot. All system processes and services are stopped and all file systems are unmounted.
/sbin/rc6	This script is run to halt and reboot the system. The /etc/rc0.d/K* scripts are executed. All system services and processes are stopped and all file systems are unmounted. Then the initdefault entry in /etc/inittab is executed.

Summary

In this lesson, you learned that:

- During the system boot procedure, the kernel dynamically loads the modules it needs and starts the /sbin/init process, the master of all processes.
- The /sbin/init process uses the /etc/inittab file to define a system's default run level and the processes to start, monitor, and restart.
- The Solaris 2.*x* environment provides various system run levels, several of which are used to perform system maintenance.
- The /sbin/rc* scripts are used to define the system levels of operation.
- The /etc/init.d/* files are used to start and stop specific services and sometimes must be customized.

Exercise 1-1

Write down the answers to the following questions.

1. Number the boot procedure steps in the correct order:

   _____ The kernel initializes itself and starts the `init` process.

   _____ PROM loads the boot block (`bootblk`) program

   _____ The boot program loads the kernel

   _____ The `init` process starts the run control scripts

   _____ PROM runs self-test diagnostics

   _____ The boot block program loads the (`ufsboot`) boot program

2. What are the two main functions of the `/sbin/init` program?

   ____

   ____

   ____

   ____

3.

   a. Which file does the `/sbin/init` program use to start processes?

      ____

   b. What is the purpose of this file?

      ____

      ____

4. What is the purpose of run control scripts?

   ____

   ____

5. Run control scripts invoke additional scripts, also called run control files. What is the purpose of these files?

   ____

   ____

Exercise 1-1

6. Name the directory which contains the run control files.

   ____

7. What is the advantage of having the run control files in this directory?

   ____

   ____

   ____

8. Describe the two steps for adding run control files.

   ____

   ____

   ____

   ____

   ____

   ____

   ____

   ____

9. Write the run level next to the function.

   ______ Halt and interactive boot

   ______ PROM monitor level

   ______ Multiuser level with resources shared

   ______ Single-user state with some file systems mounted and user logins disabled

   ______ Multiuser level with no resources shared

   ______ Reboot to default run level 3

Autoconfiguration 2

Objectives

Upon completion of this lesson, you will be able to:

- Describe the relationship between loadable kernel modules and autoconfiguration.
- State at least one benefit of the autoconfiguration process.
- Name the two directories that contain kernel modules.
- Name the file used to customize the kernel configuration process.
- Describe how to tune a kernel variable.

References

SunOS 5.1 Routine System Administration Guide, Chapter 13, "Understanding the Boot Process"

SunOS 5.1 Administering Security, Performance, and Accounting, Appendix A, "Tuning Kernel Parameters"

Introduction

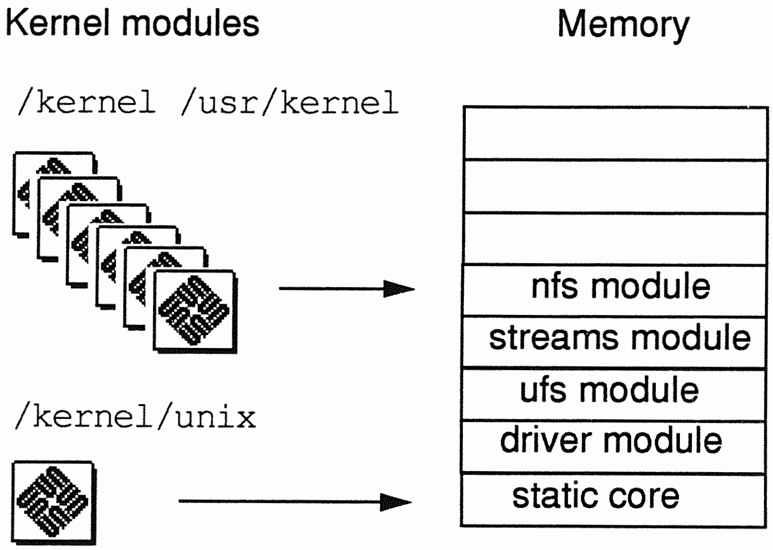
The previous lesson described the Solaris 2.*x* boot procedure. This lesson takes a closer look at the kernel configuration steps that are part of a system's boot process. Understanding the kernel configuration process is important for system administrators who must add new services and devices to a system.

Autoconfiguration Process

Introduction

The Solaris 2.*x* kernel consists of a small static core and a series of modules that are loaded on demand.

A kernel module is a hardware or software component that is used to perform a specific task within the system. An example of a loadable kernel module is a device driver which is loaded when the device is accessed.



At boot time, the system does a self-test and checks for all devices that are attached to it (as described in the previous lesson).

The kernel then configures itself dynamically, loading its modules into memory, as needed. At this time, a device driver is loaded when a device is accessed, such as the tape device. This process is called *autoconfiguration* because all kernel modules are loaded automatically when needed.

Autoconfiguration Process

Benefits

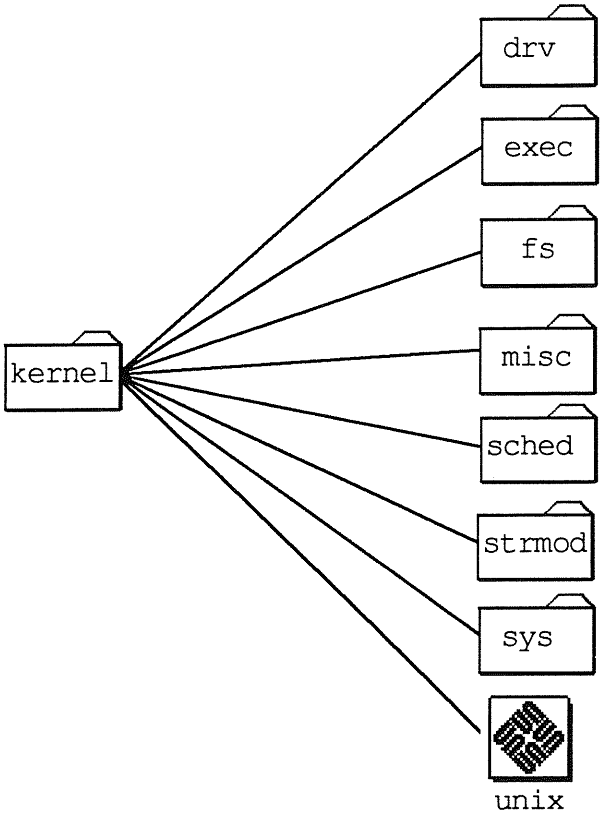
There are several benefits to the autoconfiguration process:

- Modules are loaded when they are needed, resulting in a more efficient use of main memory.
- Time is not spent reconfiguring the kernel, which was done in previous SunOS™ releases when new devices were added to the system.
- Drivers can be loaded without having to rebuild the kernel and reboot the system.

System administrators use the autoconfiguration process when adding a new device (and driver) to the system. This is done by using the `boot -r` command which performs a reconfiguration boot so that the system will recognize the new device. (These steps are described in another module.)

The /kernel Directory

Default loadable kernel modules are found in the /kernel directory.



Each subdirectory contains a particular type of module.

drv	Contains device drivers and pseudo device drivers
exec	Contains modules used to run various executable files
fs	Contains file system modules such as ufs, nfs, and proc
misc	Contains miscellaneous modules needed for virtual memory operation and inter-process communication
sched	Contains scheduling classes and corresponding dispatch tables modules
strmod	Contains STREAMS modules
sys	Contains loadable system calls such as those involved with semaphore operation

The /usr/kernel Directory

The /usr/kernel directory is also used to store loadable kernel modules. This directory contains:

- Additional modules that are not needed to complete the boot process, such as the system accounting module
- Modules that can be shared across platforms, such as the audio driver

Both the /kernel and /usr/kernel directories are used to load modules during the boot procedure.

If a module is not loaded at boot time, it may be loaded when a service is requested.

Notes

The /etc/system File

```
ident  "@(#)system    1.3    89/12/12 SMI    /* SVr4.0 1.5   */
** SYSTEM SPECIFICATION FILE
*
* moddir:
*       Set the search path for modules.  This has a format similar to the
*       csh path variable. If the module isn't found in the first directory
*       it tries the second and so on. The default is /kernel:/usr/kernel
*
*       Example:
*               moddir: /kernel /usr/kernel /sbin/modules
*
* root and swap configuration:
*       The following may be used to override the defaults provided by
*       the boot program:
*       rootfs:         Set the filesystem type of the root (rootfs:ufs).
*       rootdev:        Set the root device.  This should be a physical
*                       pathname e.g. rootdev:/sbus/esp@0,800000/sd@3,0:a.
*                       The default is the physical pathname of the device
*                       where the boot program resides.
*       swapfs:         Set the filesystem type of swap (swapfs:specfs).
*       swapdev:        Set the swap device instead of using the default
*                       as assumed by the kernel (which is to use slice 1
*                       on the same disk as the root partition).
*                       swapdev:/sbus/esp@0,800000/sd@3,0:b
* exclude:
*       Modules appearing in the moddir path which are NOT to be loaded,
*       even if referenced. Note that `exclude' expects a  module name,
*       not a filename.
*       exclude: win
*
* forceload:
*       Cause these modules to be loaded at boot time, (just after mounting
*       the root filesystem) rather than at first reference. Note that
*        forceload expects a filename which includes the directory.
*        forceload: drv/foo
*
* set:
*      Set an integer variable in the kernel or a module to a new value.
*
*        For example to set a kernel variable:
*                set nautopush=32
*
*        To set a variable in the module named 'test_module'
*                set test_module:debug=1
```

Modifying the Kernel Configuration File

The /etc/system file can be customized to change the kernel configuration process. This configuration file is read at boot time and is empty by default (except for comments).

The file can be customized in several ways.

- Use the moddir variable modify the search path of the modules to be loaded at boot time.
- Use the exclude variable to determine which modules are never loaded, even if referenced.
- Use the rootdev variable to determine an alternate root device.
- Use set *variable=value* to override a default kernel parameter.

Tuning Kernel Variables

Occasionally, your Sun service support person or third-party software vendor may suggest that you modify a kernel variable to best suit your system's needs.

For example, the maxusers kernel variable approximates the maximum number of active users on a system. It is set to 8 by default, but it can be reset to reflect the actual load of a system.

For heavily used standalone systems or servers, the value may need to be increased further. Keep in mind that the value of maxusers is used to set the size of certain kernel tables, so there is a performance penalty for setting it too high.

Modifying the Kernel Configuration File

Tuning Kernel Variables (continued)

To override a default kernel variable, modify the /etc/system file to reflect the variable change.

1. Copy the generic /etc/system file.

```
# cp /etc/system /etc/system.gen
```

2. Edit the /etc/system file and add the kernel variable.

```
set maxusers=64
```

3. Reboot the system.

Care should be taken when modifying this file because it changes the operation of the kernel. You can recover from any errors in the /etc/system file by performing an interactive boot, which prompts you for the name of an alternate system file. (The boot command is covered in a later lesson in this module.)

Note: A full discussion of how to tune the kernel is beyond the scope of this module. All of the tunable kernel parameters are listed in Appendix A of the *SunOS 5.1 Administering Security, Performance, and Accounting* Guide.

For more info see AnswerBook

Summary

In this lesson, you learned that:

- The Solaris 2.*x* kernel configures itself dynamically, loading its modules as they are needed. Therefore, device drivers are loaded into memory when the devices are accessed.

- The `/kernel` and `/usr/kernel` directories contain the loadable kernel modules.

- Kernel parameters can be changed by editing the `/etc/system` file.

- The kernel configuration process can be modified using various kernel parameters.

Exercise 2-1

Write down the answers to the following questions.

1. Describe at least two benefits of the autoconfiguration process.

   ____
   ____
   ____
   ____
   ____
   ____

2. Describe the relationship between loadable kernel modules and autoconfiguration.

   ____
   ____
   ____

3. Name the two directories that contain kernel modules.

   ____
   ____

4. Name the file used to customize the kernel configuration process.

   ____

5. List the steps for modifying the kernel configuration process.

   ____
   ____
   ____

Device Configuration

3

Objectives

Upon completion of this lesson, you will be able to:

- Describe the structure that the kernel uses to identify devices connected to the system.
- Describe the physical and instance device names that are used to identify a system's devices.
- Describe the function of the `/etc/path_to_inst` file.
- Display system configuration information with the `prtconf` and `sysdef` commands.

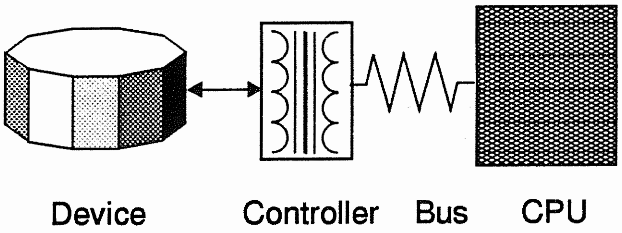
References

Solaris 2.x Handbook for SMCC Peripherals

Introduction

In order to manage the devices on your system, you should be familiar with the hardware terminology and corresponding device naming conventions that are used to identify those devices.

Device Terminology



- Controller or Interface

  All peripheral devices are connected to some kind of controller or interface, which is a hardware device that controls the peripheral device based upon signals from the Central Processing Unit (CPU).

- Host Adapter

  Small Computer System Interface (SCSI) devices have embedded controllers that are connected to a SCSI host adapter that connects to the SCSI bus. (Details on identifying SCSI devices are covered in another module.)

- Built-in Device

  Many systems have a built-in SCSI host adapter on the CPU board. This means you do not have add an interface card unless you purchase additional hardware.

- Bus

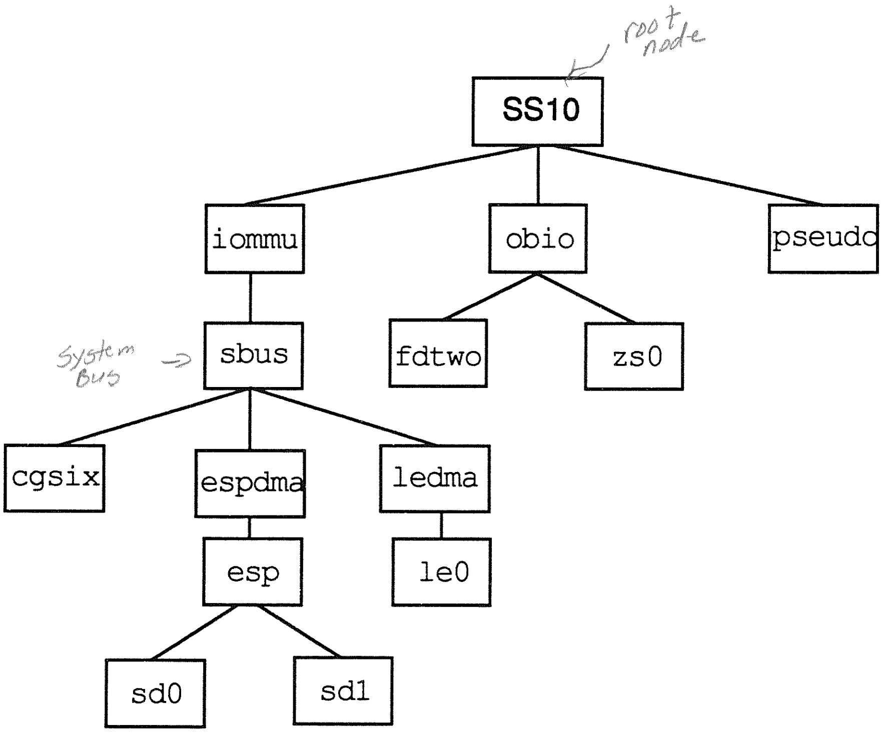
  A *bus* is a channel or pathway between hardware devices, such as the connection between the controller or host adapter and the CPU.

  For example, the SCSI bus connects SCSI devices to the SCSI host adapter. The SBUS and VME bus interfaces connect peripheral devices (such as the SCSI host adapter) to the CPU and are known as *peripheral buses.*

Device Information Tree

When the system is booted, a device hierarchy (or tree structure) is created that represents all devices attached to the system.

For example, a SPARCstation™ 10 may have the following device configuration tree structure. (Not all devices are included in the diagram.)



ESP
Emulex
Scsi
Processor

The boxes in the above diagram represent different devices on the system. For example, the `cgsix` object represents the `cgsix` color frame buffer and the `sbus` object represents the peripheral bus that is used to connect many other devices to the system.

The topmost object in the hierarchy is called the *root node* of the device information tree. An intermediate object below the `root` node has a device driver associated with it, and is called a *leaf* or *bus nexus* node.

The kernel uses the information in this device tree to associate drivers with their appropriate devices, and provides a set of pointers to the nodes (or drivers) that perform specific operations.

Device Names

In the Solaris 2.*x* environment, devices are referenced in three different ways:

- Physical
- Instance
- Logical →

A *physical device name* represents the full device pathname in the device information hierarchy (or tree) displayed on the facing page.

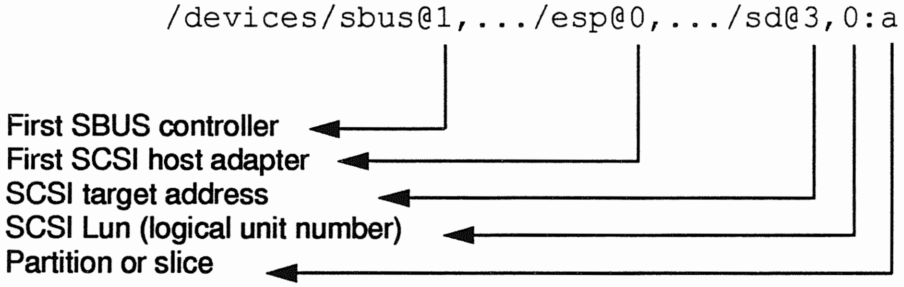
Instance names are the kernel's abbreviation names for every possible device on the system. In the device information tree, the `sd0` and `sd1` represent the instance names for two disk devices.

Logical device names are used by administrators to reference devices, and are symbolically linked to their corresponding physical device names. (These names are covered in another module.)

Physical Device Names

A physical device name represents the full device pathname in the device information hierarchy (or tree) which was described previously. Physical device files are found in the `/devices` directory and correspond to the device names used at the PROM level.

SPARCstation 2 and Older Systems



```
/devices/sbus@1,.../esp@0,.../sd@3,0:a
```

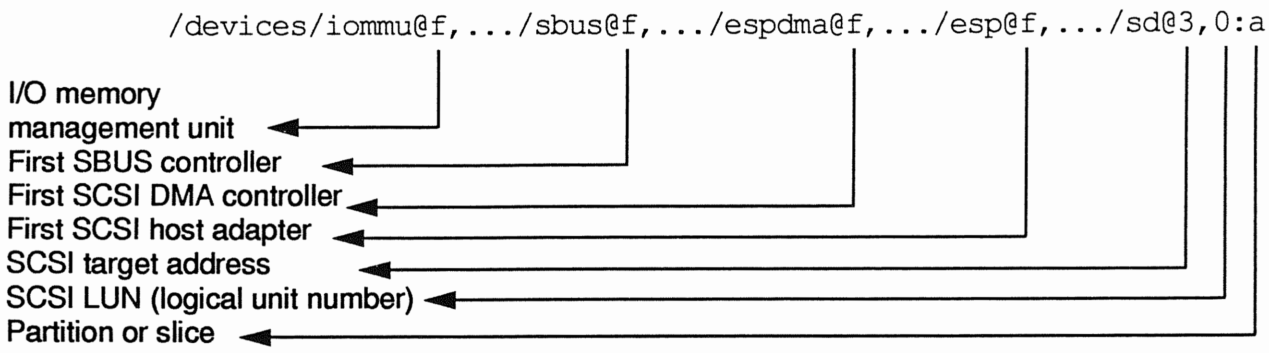
First SBUS controller
First SCSI host adapter
SCSI target address
SCSI Lun (logical unit number)
Partition or slice

The ellipses (. . .) represent the virtual memory address of the corresponding device.

The above example describes the first SCSI disk (target address 3) connected to the first SCSI host adapter.

Physical Device Names

SPARCstation 10 and Newer Systems


/devices/iommu@f,.../sbus@f,.../espdma@f,.../esp@f,.../sd@3,0:a
I/O memory management unit
First SBUS controller
First SCSI DMA controller
First SCSI host adapter
SCSI target address
SCSI LUN (logical unit number)
Partition or slice

The SPARCstation 10 and newer systems have a separate I/O memory management unit (iommu) on the CPU board. The architecture of these systems also use DMA controllers for SCSI devices and ethernet devices.

A *direct memory access* (DMA*) controller* is a specialized device that allows data to be transferred directly between memory without going through the CPU.

The built-in SCSI host adapter connects to the SCSI DMA controller.

The virtual address locations of the SPARCstation 10 devices begin `f,...`, which identify a built-in device. The second SBUS device might be addressed as `sbus@0,...` which represents the first (lowest numbered) SBUS slot.

Device Instance Names

An instance name is the kernel's abbreviation for the physical device name. Instance names are allocated when the kernel finds an instance of a device for the first time.

In the Solaris 2.*x* environment, the instance name is bound to the physical name by references in the /etc/path_to_inst file.

The /etc/path_to_inst File

```
#     Caution! This file contains critical kernel state
#
“/iommu@f,e0000000” 0
“/iommu@f,e0000000/sbus@f,e0001000” 0
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000” 0
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000” 0
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/st@5,0” 5
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/st@4,0” 4
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/st@6,0” 6
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/st@1,0” 1
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/st@0,0” 0
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/st@3,0” 3
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/st@2,0” 2
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@5,0” 5
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@4,0” 4
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@6,0” 6
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@1,0” 1
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@0,0” 0
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0” 3
“/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@2,0” 2
“/iommu@f,e0000000/sbus@f,e0001000/sbusmem@f,0” 15
“/iommu@f,e0000000/sbus@f,e0001000/cgsix@2,0” 0
“/iommu@f,e0000000/sbus@f,e0001000/sbusmem@0,0” 0
“/iommu@f,e0000000/sbus@f,e0001000/sbusmem@1,0” 1
“/iommu@f,e0000000/sbus@f,e0001000/sbusmem@2,0” 2
“/iommu@f,e0000000/sbus@f,e0001000/sbusmem@3,0” 3
“/iommu@f,e0000000/sbus@f,e0001000/ledma@f,400010” 0
“/iommu@f,e0000000/sbus@f,e0001000/ledma@f,400010/le@f,c00000” 0
“/iommu@f,e0000000/sbus@f,e0001000/ledma@f,400010/le@f,c00000” 0
“/iommu@f,e0000000/sbus@f,e0001000/SUNW,bpp@f,4800000” 0
“/iommu@f,e0000000/sbus@f,e0001000/SUNW,DBRIe@f,8010000” 0
                                    .
                                    .
                                    .
```

Device Instance Names

The /etc/path_to_inst File (continued)

The device instance is the number on the right side of the file. (They are bolded in the displayed output.) The kernel uses these names to identify every possible device instance.

It may look like the instance number is same as the target number in the example on the facing page. However, if an additional SCSI controller was added to the system, the instance numbers would increment sequentially, but the target numbers would remain the same (0-7).

```
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000”  1
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/sd@0,0”  7
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/sd@1,0”  8
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/sd@2,0”  9
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/sd@3,0”  10
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/sd@4,0”  11
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/sd@5,0”  12
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/sd@6,0”  13
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/st@0,0”  7
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/st@1,0”  8
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/st@2,0”  9
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/st@3,0”  10
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/st@4,0”  11
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/st@5,0”  12
"/iommu@f,e0000000/sbus@f,e0001000/esp@1,200000/st@6,0”  13
```

This file is consulted each time the system is booted and is recreated whenever a new device is added and a reconfiguration boot is performed. (The boot command is covered in the next lesson.)

This file should not be modified and is maintained by the kernel.

Displaying System Configuration

The prtconf Command

The prtconf command displays a system's configuration information, including memory and peripheral configurations. (See the output on the facing page.)

Note the use of device instance names to distinguish the possible devices that may be connected to the same interface. For example, the devices connected to the SCSI bus (sbus) are listed immediately underneath the sbus device listing.

The disk and tape device instance names correspond to possible address locations on the SCSI host adapter, which are called *target addresses*. Assigning target address locations is how the system distinguishes different devices connected to the same interface.

The driver not attached message means that no driver for that device is in use at the time the prtconf command is issued, or there is no driver for that device instance.

The sysdef Command

The sysdef command is also used to display system configuration information.

The sysdef output basically identifies the same device information plus the following additional information:

- Host ID
- Pseudo devices
- Loadable modules
- Selected kernel parameter values

Displaying System Configuration

Example:

```
# prtconf
System Configuration:  Sun Microsystems  sun4m ← Kernel Info
Memory size: 32 Megabytes
System Peripherals (Software Nodes):
SUNW,SPARCstation-10
    packages (driver not attached)
        disk-label (driver not attached)
        deblocker (driver not attached)
        obp-tftp (driver not attached)
    options, instance #0
    aliases (driver not attached)
    openprom (driver not attached)
    iommu, instance #0
        sbus, instance #0
            espdma, instance #0
                esp, instance #0
                    sd (driver not attached)
                    st (driver not attached)
                    sd, instance #0 (driver not attached)
                    sd, instance #1 (driver not attached)
                    sd, instance #2 (driver not attached)
                    sd, instance #3
                    sd, instance #4 (driver not attached)
                    sd, instance #5 (driver not attached)
                    sd, instance #6
            ledma, instance #0
                le, instance #0
            SUNW,bpp (driver not attached)
            SUNW,DBRIe (driver not attached)
                mmcodec (driver not attached)
            cgsix, instance #0
    obio, instance #0
        zs, instance #0
        zs, instance #1
        eeprom (driver not attached)
        counter (driver not attached)
        interrupt (driver not attached)
        SUNW,fdtwo, instance #0
        auxio (driver not attached)
        power (driver not attached)
    memory (driver not attached)
    virtual-memory (driver not attached)
    eccmemctl (driver not attached)
    TI,TMS390Z50 (driver not attached)
    pseudo, instance #0
```

Reconfiguring Devices

When new devices are added to the system, a reconfiguration boot is performed to recognize the new devices. This reconfiguration process begins by creating a new device information tree.

The steps for adding a new device to the system are covered in another module.

ok boot -r => to reconfig with new device

Summary

In this lesson, you learned that:

- The kernel builds a hierarchy or tree structure that represents all the devices on the system.
- Devices currently connected to the system can be displayed with the `prtconf` or `sysdef` commands.
- The `/etc/path_to_inst` file contains a listing of all the device instances connected to the system since the last reboot. The instance names stored in this file are abbreviation names which the kernel uses to describe different devices.
- Physical device names represents the full path name of the devices in the device information hierarchy. Device files representing the physical device names are found in the `/devices` directory.

Exercise 3-1

Part I

Write down the answers to these questions.

1. What type of structure is created during bootup that represents all devices attached to the system?

   ____

   ____

2. Describe the physical and instance device names.

   ____

   ____

   ____

   ____

   ____

   ____

   ____

   ____

   ____

Part II

The purpose of this exercise is to identify the devices that are connected to your system. In addition, identify the device names that are used to describe the devices.

1. Use the `prtconf` or `sysdef` commands to display device information on your system.

2. Using the output of the `prtconf` or `sysdef` commands, find the `/devices` entry for one of your disk devices.

Changing Run Levels 4

Objectives

Upon completion of this lesson, you will be able to:

- Recall at least two reasons for halting a system.
- List the four commands used to change system run levels.
- Change run levels using the `init` and `shutdown` commands.
- Name the command to boot a system to single-user mode from the PROM monitor level.
- Describe the function of these PROM monitor level commands: `banner`, `printenv`, `devalias`, `probe-scsi`, `setenv`.
- Name the command used to perform an interactive boot.
- Recall how to boot a system when the `/etc/passwd` file is missing or corrupted.
- Describe when and how to abort a system.

References

SunOS 5.1 Routine System Administration Guide, Chapter 1, "Introducing System Administration"

Introduction

This lesson describes the commands used to change system run levels.

Sun systems are designed to be left powered on continuously. However, you must halt or shut down the system, and sometimes turn the power off, when performing several maintenance tasks:

- Installing a new release of the operating system
- Turning off a system's power due to anticipated power outage
- Adding or removing hardware from the system
- Performing file system maintenance, such as a backup or restoring important system data

Scheduling System Shutdowns

It is important for system administrators to schedule tasks that affect their users' ability to use a system.

Tasks like replacing defective hardware or troubleshooting problems that require system downtime are not easily anticipated. Routine tasks such as file system maintenance and backups should be scheduled so that users can schedule their work accordingly.

There are different ways to notify users of impending system downtime:

- Send messages to logged in users with the `wall` command.
- Send messages to a network of users with the `rwall` command.
- Send electronic mail messages to affected users.
- Use the `/etc/motd` (message-of-the-day) file to supply a message to users who log in to the system about scheduled down time.

See the *SunOS 5.1 Routine System Administration Guide*, Chapter 1, "Introducing System Administration," for information on communicating with users.

Changing System Run Levels

All of the Solaris commands used to change system run levels require superuser privilege to run.

The commands that are used to shutdown a system to run level S or the PROM monitor level perform "clean shutdowns."

A *clean shutdown* means the operating system is shut down in an orderly fashion where all processes are stopped, file systems are unmounted, and data in memory is copied back to disk.

Several commands are available to change system run levels.

- `shutdown`
- `init`
- `halt`
- `reboot`

The shutdown Command

The /usr/sbin/shutdown command is used to change a system's run level. The system processes that are stopped, and file systems that are unmounted, depend on the run level the system is to be changed to. In most cases, it is used to get from run level 3 to run level S. More important, it notifies users of the event.

If the shutdown command is used to bring the system to run level 0, the operating system is cleanly shut down—meaning all processes are stopped and all file systems are unmounted.

Command format:

```
shutdown [ -y ] [ -gseconds ] [ -irun_level ]
```

Options:

y — Use this option to continue with the system shutdown without intervention; otherwise, you are prompted to continue the shutdown process.

g — Allows you to specify a time (in seconds) before the system is shutdown. This overrides the 60-second default.

i — Allows you to bring the system to a different run level other than the default run level S. Choices are run levels 0, 1, 2, 5, and 6.

Exit the OpenWindows™ environment and cd to the / (root) directory before using the shutdown command (as the superuser).

Examples:

1. Bring the system to run level S.

```
# cd /
# shutdown

Shutdown started. Tue May 17 15:07:00 PDT 1993
Do you want to continue? (y or n): y
```

The shutdown Command

Users logged in on this system see the following warning messages:

```
Broadcast Message from root (console) on venus Tue May 17
15:07:01...
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.
```

The above warning messages are displayed immediately after the shutdown command is issued. A final message is sent before the shutdown begins.

2. Bring the system to run level S in 5 minutes without further interaction. (Exit the OpenWindows environment first.)

```
# shutdown -y -g300 -iS
Shutdown started. Wed May 25 14:11:46 PDT 1993
Broadcast Message from root (console) on venus Wed May 25
14:11:49...
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.

The system is coming down. Please wait.
System services are now being stopped.
Print services stopped.
Stopping the syslog service.
May 25 14:12:20 venus syslogd: going down on signal 15
The system is down.
Changing to init state s - please wait
#
INIT: New run level: S

INIT: SINGLE USER MODE

Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance):  xxx
Entering System Maintenance Mode
Sun Microsystems Inc. SunOS 5.2 Generic March 1993
#
```

You can vi the shutdown script to get a message.
/usr/bsd for old commands

The shutdown Command

Use the who -r command to identify the current run level.

```
# who -r
  . run-level S Jun 10 12:13 S 1 3
#
```

# of times you have been here before

previous run level

Once the maintenance procedure is completed at run level S, bring the system back to run level 3. Press Control-D and type 3, and press Return.

```
Entering System Maintenance Mode

Sun Microsystems Inc. SunOS 5.2 Generic March 1993
# ^D
ENTER RUN LEVEL (0-6,s or S):  3
will change to state 3

INIT: New run level: 3
The system is coming up. Please wait.
checking filesystems
         .
         .
         .
The system is ready.

venus console login:
```

3. Halt the system in 5 minutes and reboot without further interaction. (Exit the OpenWindows environment first.)

```
# shutdown -y -g300 -i6
```

The init Command

The init command can be used instead of the shutdown command to change system run levels. However, the init command does not send warning messages before changing run levels.

Both the shutdown and init commands place the system at the specified run level in a clean manner.

You must become superuser before using the init command to change run levels.

Command format:

```
init [ 012356QqSs ]
```

The 0-6, and S, s options correspond to the appropriate run levels.

Option	Action
0	Bring the system to PROM monitor level.
1	Bring the system to single-user level where some file systems are mounted and user logins are disabled.
2	Bring the system to multiuser level (with no resources shared).
3	Bring the system to multiuser level (with resources shared).
5	Halt the system and perform an interactive boot (boot -a).
6	Halt and reboot the system to run level 3.
S, s	Bring the system to single-user level where some file systems are mounted and user logins are disabled.
Q, q	Tell the init program to re-read the /etc/inittab file.

The init Command

Examples:

1. Bring the system to the PROM monitor level.

```
# init 0
INIT: New run level: 0
The system is coming down. Please wait.
System services are now being stopped.
Print services stopped.
Stopping the syslog service.
May 17 15:32:42 venus syslogd: going down on signal 15
The system is down.
Halted
Program terminated
Type help for more information
ok
```

2. Reboot the system to run level 3.

```
# init 6
INIT: New run level: 6
The system is coming down. Please wait.
System services are now being stopped.
Print services stopped.
Stopping the syslog service.
May 17 15:40:23 venus syslogd: going down on signal 15
The system is down.
rebooting...
SPARCstation 10 (1 X 390Z50), Keyboard Present
ROM Rev. 2.10, 32 MB memory installed, Serial #3159808.
Ethernet address 8:0:20:1a:e7:3f, Host ID: 72303700.
Rebooting with command:
Boot device: /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0
SunOS Release 5.2 Version Generic [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1993, Sun Microsystems, Inc.
Hostname: venus
The system is coming up. Please wait.
checking filesystems
        .
        .
        .
The system is ready.
venus console login:
```

Other Commands

The `halt` Command

Using the `/usr/sbin/halt` command is equivalent to using `init 0`, which performs a clean shutdown and brings the system to PROM monitor level.

The `reboot` Command

The `/usr/sbin/reboot` command performs a clean shutdown and brings the system to run level 3 by default, just like the `init 6` command.

Use the `reboot -- -r` command to reboot the system and perform a reconfiguration boot.

The PROM Monitor

Once a system is brought to the PROM monitor level, use the boot command to bring it up to a different run level.

The boot command requires an argument representing the boot device, if there is no default boot device. The boot device is specified differently, depending on the PROM level of the machine.

The Open Boot PROM

The open boot PROM (OBP) refers to Sun's philosophy about open systems.

The SPARCstation 2, ELC™ , IPX™ and newer machines use the OBP Revision 2.0 and above, which means the boot device is specified by the physical device name found in the device information hierarchy.

Older systems such as the SPARCstation 1, 1+, IPC™, and SLC™ machines use the OBP Revision 1.*x*, which has different features and syntax than OBP Revision 2.*x*.

Use the banner command at the ok prompt to identify your system's PROM version number.

Example:

```
ok banner
SPARCstation 10 (1 X 390Z50), Keyboard Present
ROM Rev. 2.10, 32 MB memory installed, Serial #3159808.
Ethernet address 8:0:20:1a:e7:3f, Host ID: 72303700.
ok
```

OBP Revision 2.*x* Device Names

In order to specify a boot device on a SPARCstation 2 and above, without using the physical device name, use the `devalias` command to identify possible boot devices.

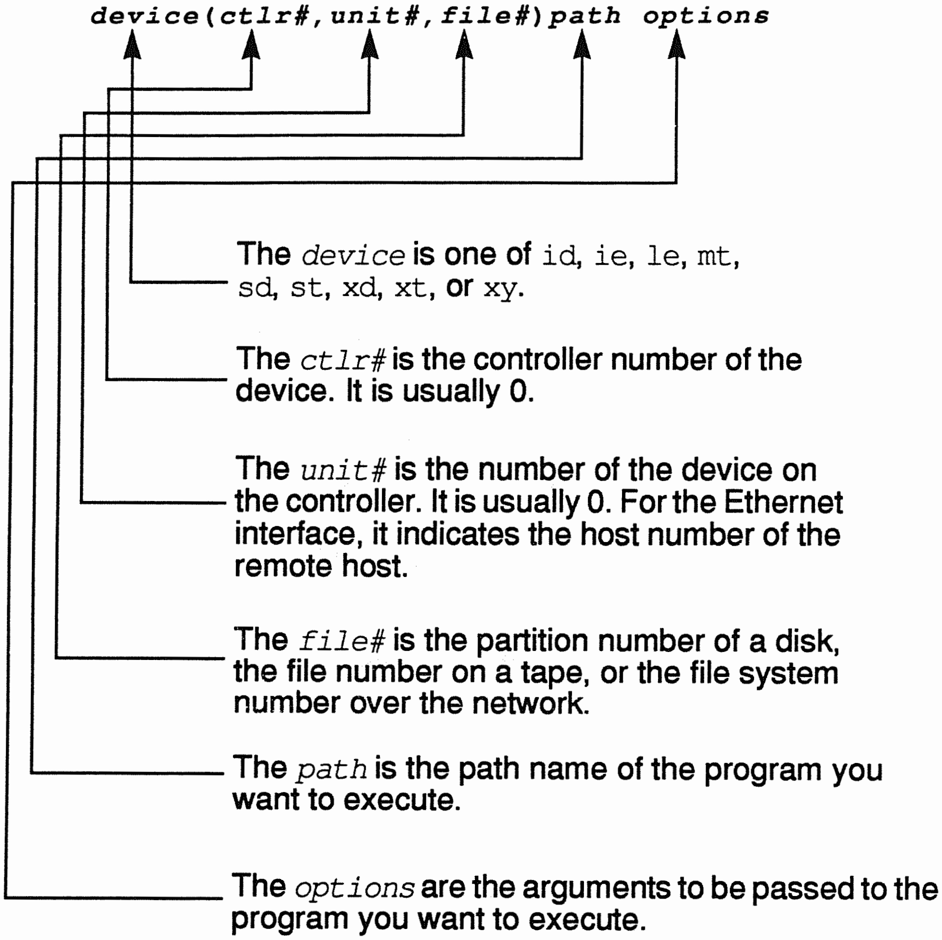
```
ok devalias
screen      /iommu@f,e0000000/sbus@f,e0001000/cgsix@2,0
floppy      /obio/SUNW,fdtwo
scsi        /iommu/sbus/espdma@f,400000/esp@f,800000
net-aui     /iommu/sbus/ledma@f,400010:aui/le@f,c00000
net-tpe     /iommu/sbus/ledma@f,400010:tpe/le@f,c00000
net         /iommu/sbus/ledma@f,400010/le@f,c00000
disk        /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0
cdrom       /iommu/sbus/espdma@f,400000/esp@f,800000/sd@6,0:d
tape        /iommu/sbus/espdma@f,400000/esp@f,800000/st@4,0
tape0       /iommu/sbus/espdma@f,400000/esp@f,800000/st@4,0
tape1       /iommu/sbus/espdma@f,400000/esp@f,800000/st@5,0
disk3       /iommu/sbus/espdma@f,400000/esp@f,800000/sd@0,0
disk2       /iommu/sbus/espdma@f,400000/esp@f,800000/sd@2,0
disk1       /iommu/sbus/espdma@f,400000/esp@f,800000/sd@1,0
disk0       /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0
ttyb        /obio/zs@0,100000:b
ttya        /obio/zs@0,100000:a
keyboard!   /obio/zs@0,0:forcemode
keyboard    /obio/zs@0,0
```

The device alias name is listed on the left side of the output. In the above output, the bolded `disk` device alias usually identifies the system's default boot device.

It is important to note that the device alias lists *possible* boot device names.

OBP Revision 1.*x* Device Names

Device names at the PROM level are specified with the `boot` command using the following format on older machines.



```
device(ctlr#,unit#,file#)path options
```

The *device* is one of `id`, `ie`, `le`, `mt`, `sd`, `st`, `xd`, `xt`, or `xy`.

The *ctlr#* is the controller number of the device. It is usually 0.

The *unit#* is the number of the device on the controller. It is usually 0. For the Ethernet interface, it indicates the host number of the remote host.

The *file#* is the partition number of a disk, the file number on a tape, or the file system number over the network.

The *path* is the path name of the program you want to execute.

The *options* are the arguments to be passed to the program you want to execute.

The boot Command

Use the `boot` command from the PROM monitor level to change to a different run level.

Command format:

```
ok boot [device_name] [options]

> b [device_name] [options]
```

Options:

`a` Performs an interactive boot that prompts for `root` and swap devices and several important system files

`r` Performs a reconfiguration boot where the system probes all attached devices and creates entries for all found devices in the `/devices` and `/dev` directories

`s` Brings the system to run level S, but first prompts for the password

`v` Displays detailed startup messages

`w` Makes the file system writeable

Allows you to boot and perform with options to the mounted file systems. (usually mounted ro) ←

ok boot cdrom -sw 2.X

Examples: boot sd(0,6,2) -sw 1.X

1. Boot a Sun-4c™ system to run level 3 using the default boot device.

   ```
   ok boot
   ```

2. Boot a Sun-4c system using an alternate boot device.

   ```
   ok boot sd(0,3,0)
   ```

3. Boot a diskless client.

   ```
   ok boot net
   ```

The boot Command

4. This example illustrates the boot of any Sun-4m™ to run level S.

```
ok boot -s
Resetting ...
SPARCstation 10 (1 X 390Z50), Keyboard Present
ROM Rev. 2.10, 32 MB memory installed, Serial #3159808.
Ethernet address 8:0:20:1a:e7:3f, Host ID: 72303700.

Rebooting with command: -s
Boot device: /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0
File and args: -s
SunOS Release 5.2 Version Generic [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1993, Sun Microsystems, Inc.
configuring network interfaces: le0.
Hostname: venus

INIT: SINGLE USER MODE

Type Ctrl-d to proceed with normal startup,
(or give root password for system maintenance):  xxx
Entering System Maintenance Mode

Sun Microsystems Inc. SunOS 5.2 Generic March 1993
# ^D
INIT: New run level: 3
The system is coming up. Please wait.
checking filesystems
       .
       .
       .
The system is ready.

venus console login:
```

The root password must be supplied to enter run level S. Press Control-D when you are ready to bring the system up to run level 3.

What if the System Won't Boot?

There are several different reasons why a system won't boot. Some scenarios are covered over the next several pages.

Wrong Boot Device

If a system won't boot because the boot device isn't specified correctly, use the `probe-scsi`, `probe-scsi-all`, or `probe-ipi` command from the PROM monitor to identify what devices are connected to the system.

```
ok probe-scsi-all
/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000
Target 3
  Unit 0   Disk     SEAGATE ST1480   SUN0424626600190016Copyright (c) ...
Target 6
  Unit 0   Removable Read Only device     SONY    CD-ROM CDU-8012 3.1a
ok
```

In the bolded output above, the only disk device on this system is the first disk on the first controller (at target address 3), or device instance name `sd3`.

The default `devalias` name for the only disk on the system is `disk`.

What if the System Won't Boot?

The printenv Command

Use the printenv command to display the default boot device.

In the following example, the boot device on a SPARCstation 2 is erroneously set to disk2, which isn't connected to the system.

```
ok printenv
Parameter Name      Value                Default Value

tpe-link-test?      true                   true
output-device       ttya                   screen
input-device        ttya                   keyboard
sbus-probe-list     f0123                  f0123
keyboard-click?     false                  false
keymap
ttyb-rts-dtr-off    false                  false
ttyb-ignore-cd      true                   true
ttya-rts-dtr-off    false                  false
ttya-ignore-cd      true                   true
ttyb-mode           9600,8,n,1,-           9600,8,n,1,-
ttya-mode           9600,8,n,1,-           9600,8,n,1,-
fcode-debug?        false                  false
diag-file
diag-device         net                    net
boot-file
boot-device         disk2                  disk
auto-boot?          true                   true
More [<space>,<cr>,q] ? q
```

The setenv Command

Use the setenv command to reset PROM settings, printenv to verify the new setting, and reset to confirm the change and reboot. In this example, the default boot device is reset disk2 to disk.

```
ok setenv boot-device disk
boot-device =       disk
ok printenv boot-device
boot-device         disk                   disk
ok reset
```

What if the System Won't Boot?

Important System File Is Missing or Invalid

If an important system file, such as /etc/system, is missing or contains an invalid entry, the system probably won't boot.

Perform an interactive boot using the -a option to prompt you for a copy of the appropriate system file.

```
ok boot -a
Resetting ...
SPARCstation 10 (1 X 390Z50), Keyboard Present
ROM Rev. 2.10, 32 MB memory installed, Serial #3159808.
Ethernet address 8:0:20:1a:e7:3f, Host ID: 72303700.

Rebooting with command: -a
Boot device: /iommu/sbus/espdma@f,400000/esp@f,800000/sd@3,0 File and args: -a
Enter filename [/kernel/unix]: Return
SunOS Release 5.2 Version Generic [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1993, Sun Microsystems, Inc.
Name of system file [etc/system]: /etc/system.gen
Name of default directory for modules [/kernel /usr/kernel]: Return
Enter name of device instance number file [/etc/path_to_inst]: Return
root filesystem type [ufs]: Return
Enter physical name of root device
[/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0:a]: Return
configuring network interfaces: le0.
Hostname: venus
The system is coming up. Please wait.
checking filesystems
      .
      .
      .
The system is ready.

venus console login:
```

What if the System Won't Boot?

Critical System File Is Missing or Invalid

If the `/etc/passwd` file is missing or invalid, the system will probably boot, but no one will be able to login.

The following procedure describes how to recover from an invalid password file.

1. Perform a standalone boot from the Solaris 2.*x* installation CD-ROM.
2. Mount the appropriate file system and repair the invalid file.
3. Unmount the file system.
4. Reboot the system.

```
ok boot cdrom -sw
Resetting ...
SPARCstation 10 (1 X 390Z50), Keyboard Present
ROM Rev. 2.10, 32 MB memory installed, Serial #3159808.
Ethernet address 8:0:20:1a:e7:3f, Host ID: 72303700.

Rebooting with command: cdrom -sw
Boot device: /iommu/sbus/espdma@f,400000/esp@f,800000/sd@6,0:d
SunOS Release 5.2 Version Beta_1.0 [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1992, Sun Microsystems, Inc.
WARNING: clock gained 447 days -- CHECK AND RESET THE DATE!
Configuring the /devices directory
Configuring the /dev directory

INIT: SINGLE USER MODE
# mount /dev/dsk/c0t3d0s0 /mnt
# TERM=sun
# export TERM
# cd /mnt/etc
# vi /etc/passwd
# cd /
# umount /mnt
# reboot
```

target

slice # => partition

controller

device (SCSI will always be 0)

bourne shell to get vi to work

Interrupting the System

Sometimes it is necessary to abort the system if the machine "hangs" and you can't log in remotely from another machine for any recovery attempts.

Aborting the system stops the processor at once. It does not synchronize the file systems (write changes to disk with the `sync` command) or provide any warning messages.

If all other recovery procedures have failed, try the following steps.

1. Attempt to abort the operating system using the Stop-A keys (L1-A on an older keyboard). If successful, the system is placed at the PROM monitor mode.

   If an ASCII terminal is being used as the system console, press the BREAK key.

2. Synchronize the disks and reboot. On SPARCstations and later models use:

   ```
   > n
   ```
   then
   ```
   ok sync
   ```

3. If you aborted the system inadvertently, you can attempt to recover with the following command on SPARCstations and later models:

   ```
   ok go
   ```
   → pick up where it left off after making firmware changes

   or

   ```
   > c
   ```

Ok old-mode
> n
Ok

} To go from new to old prompt
} To go from old to new prompt

Summary

In this lesson, you learned that:

- Several different commands are used to change system run levels.
  - `shutdown`
  - `init`
  - `halt`
  - `reboot`
- The syntax for specifying a boot device from the PROM monitor level depends on the PROM revision number.
- Default boot devices can be displayed and changed with the `printenv` and `setenv` commands at the PROM monitor level.
- Recovering from missing or invalid system files can be facilitated by performing an interactive or standalone boot procedure.

Exercise 4-1

Part I

Write down the answers to the following questions.

1. What is the difference between the `shutdown` command and the `init` command?

2. The `reboot` command is the equivalent of what command?

3. The `halt` command is the equivalent to what command?

4. Match the commands to their function:

   ______ `banner` a. Displays the default boot device

   ______ `printenv` b. Resets PROM parameters

   ______ `devalias` c. Displays a system's PROM version number

   ______ `probe-scsi` d. Displays the devices connected to the SCSI bus

   ______ `setenv` e. Displays the aliases for physical device names

5. List the steps for performing a standalone boot for troubleshooting purposes.

Exercise 4-1

Part II

Write down the commands and perform the following tasks.

1. Bring the system to the PROM level in two minutes, sending warning messages and without additional interaction.
   shutdown -y -g120 -i0
2. Write your system's PROM version number.
   ok banner
   ROM Rev 1.6
3. Display the aliases for physical devices.
   Won't work under Rev 1.6
4. Display your default boot device.
   ok printenv
5. Boot your system to full multiuser mode.
   ok boot
6. Use the `shutdown` command to change to run level S without additional interaction. (Exit the OpenWindows environment first.)
   shutdown -y -is
7. Bring your system up to full multiuser mode.
   control d
8. Reboot the system.
   ____

Answer Key A

Lesson 1: The Boot Process

Exercise 1-1

1.

___5___ The kernel initializes itself and starts the `init` process.

___2___ PROM loads the boot block (`bootblk`) program

___4___ The (`ufsboot`) boot program loads the kernel

___6___ The `init` process starts the run control scripts

___1___ PROM runs self-test diagnostics

___3___ The boot block program loads the (`ufsboot`) boot program

2. The `/sbin/init` process:

   - Creates processes, which has the effect of bringing the system up to the default run level.

   - Controls transitions between run states by re-reading the `/etc/inittab` file.

3.

   a. `/etc/inittab`

   b. The entries in this file tell the `init` process what processes to create for each run level and what actions to perform. This file is part of the Solaris 2.*x* system software and need not be edited manually.

4. The purpose of run control scripts is to define system levels of operation by checking and mounting file systems, starting various processes, and performing other housekeeping tasks.

5. Run control files start up or kill specific processes.

6. `/etc/init.d`

Lesson 1: The Boot Process

Exercise 1-1 (continued)

7. Having the run control files in the `/etc/init.d` directory allows you to start and stop individual services or processes without changing run levels.

8.

   a. Place the startup file in the `/etc/init.d` directory and create a link in the appropriate `/etc/rc*.d` directory.

   b. Assign an appropriate sequence number for the startup file in the `/etc/rc*.d` directory so that it does not conflict with an existing sequence number. Make sure that the service you are starting comes after a service that it needs to start successfully.

9.

___5___ Halt and interactive boot

___0___ PROM monitor level

___3___ Multiuser level with resources shared

_S, s, 1__ Single-user state with some file systems mounted and user logins disabled

___2___ Multiuser level with no resources shared

___6___ Reboot to default run level 3

Lesson 2: Autoconfiguration

Exercise 2-1

1.

   a. Modules are loaded when they are needed, resulting in more efficient use of main memory.

   b. Time is not spent configuring the kernel.

   c. Drivers can be loaded without having to rebuild the kernel and reboot the system.

2. The autoconfiguration process is based on the fact that kernel modules are dynamically loaded so the kernel can configure itself.

3. `/kernel` and `/usr/kernel`

4. `/etc/system`

5.

   a. Copy the generic `/etc/system` file to another file, if not already done.

   b. Edit the `/etc/system` file and make the desired change.

   c. Reboot the system.

Lesson 3: Device Configuration

Exercise 3-1

Part I

1. A device hierarchy or tree structure.

2. A physical device name represents the full path name of a device. Physical device files are found in the `/devices` directory and correspond to the device names used at the PROM level.

   An instance name is the kernel's abbreviation for the physical device name. Instance names are allocated when the kernel finds an instance of a device for the first time.

Part II

Follow the steps as described. Output may vary for each system.

Lesson 4: Changing Run Levels

Exercise 4-1

Part I

1. The `init` command does not send any warning messages.

2. `init 6`

3. `init 0`

4. An open boot PROM allows devices to easily be added to the system because device drivers and booting functions are no longer linked to boot PROM code.

5.

___c___	`banner`	a. Displays the default boot device
___a___	`printenv`	b. Resets PROM parameters
___e___	`devalias`	c. Displays a system's PROM version number
___d___	`probe-scsi`	d. Displays the devices connected to the SCSI bus
___b___	`setenv`	e. Displays the aliases for physical device names

6.

   a. Perform a standalone boot from the Solaris 2.*x* installation CD-ROM.

   b. Set the terminal type.

   c. Mount the appropriate file system and make the necessary repairs.

   d. Unmount the file system.

   e. Reboot the system.

Lesson 4: Changing Run Levels

Exercise 4-1 (continued)

Part II

1. `cd /` and
   `shutdown -y -g 120 -i0`

2. ok **banner**

   The output may vary for each system.

3. ok **devalias**

   The output may vary for each system.

4. ok **printenv**

5. ok **boot**
   or
   > **b**

6. `cd /` and
   `shutdown -y`

7. Press Control-D and type `3`, and press Return

8. `reboot` or `init 6`