Troubleshooting Typical Issues in Oracle® Solaris 11.1

Part No: E29013–01
October 2012

ORACLE

121010@25097

Contents

Preface

Troubleshooting Typical Issues in Oracle Solaris 11.1 is part of a documentation set that provides a significant portion of the Oracle Solaris system administration information. This guide contains information for both SPARC based and x86 based systems.

This book assumes you have completed the following tasks:

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

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

Note – This Oracle Solaris release supports systems that use the SPARC and x86 families of processor architectures. The supported systems appear in the *Oracle Solaris OS: Hardware Compatibility Lists*. This document cites any implementation differences between the platform types.

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

Who Should Use This Book

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

Access to Oracle Support

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

Typographic Conventions

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

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

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

TABLE P–2 Shell Prompts

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

◆ ◆ ◆ CHAPTER 1

Managing System Crash Information (Tasks)

This chapter describes how to manage system crash information in the Oracle Solaris OS.

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

What's New in Managing System Crash Information

This section describes new or changed features for managing system resources in this Oracle Solaris release.

Changes to `savecore` Behavior

The `savecore` command now initially creates files with a `.partial` suffix that is appended to the file. After the file is completely written, it is renamed and the suffix is removed. Potential problems can prevent the file from being renamed and the suffix from being removed, for example, if the `savecore` command is still busy. Another example is if the `savecore` command is interrupted due to a system crash shortly after booting.

If the command is busy, you can use the `ps` command to search for the process ID (PID) of the running `savecore` process and then wait for the process to complete. If the process is interrupted, you can manually remove the leftover file and then recreate it by running the `savecore` command with the `-d` option.

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

System Crashes (Overview)

Keep the following key points in mind when you are working with system crash information:

- You must assume the `root` role to access and manage system crash information. See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.
- Do not disable the option of saving system crash dumps on the system. System crash dump files provide an invaluable way to determine what is causing the system to crash.
- Do not remove important system crash information until it has been sent to your customer service representative.

System crashes can occur due to hardware malfunctions, I/O problems, and software errors. If the system crashes, it will display an error message on the console, and then write a copy of its physical memory to the dump device. The system will then reboot automatically. When the system reboots, the `savecore` command is executed to retrieve the data from the dump device and write the saved crash dump to your `savecore` directory. The saved crash dump files provide invaluable information to aid in diagnosing the problem.

The crash dump information is written in a compressed format to the `vmdump.`*n* file, where *n* is an integer that identifies the crash dump. Afterwards, the `savecore` command can be invoked on the same system or another system to expand the compressed crash dump to a pair of files that are named `unix.`*n* and `vmcore.`*n*. The directory in which the crash dump is saved upon reboot can also be configured by using the `dumpadm` command.

Dedicated ZFS volumes are used for swap and dump areas. After an installation, you might need to adjust the size of swap and dump devices or possibly recreate the swam and dump volumes. For instructions, see "Managing Your ZFS Swap and Dump Devices" in *Oracle Solaris 11.1 Administration: ZFS File Systems*.

System Crash Dump Files

The `savecore` command runs automatically after a system crash to retrieve the crash dump information from the dump device and writes a pair of files, called `unix.`*x* and `vmcore.`*x*, where *x* identifies the dump sequence number. Together, these files represent the saved system crash dump information.

Note – Crash dump files are sometimes confused with *core* files, which are images of user applications that are written when the application terminates abnormally.

Crash dump files are saved in a predetermined directory, which by default, is `/var/crash/`. In previous releases, crash dump files were overwritten when a system rebooted, unless you manually enabled the system to save the images of physical memory in a crash dump file. Now, the saving of crash dump files is enabled by default.

System crash information is managed with the dumpadm command. For more information, see "Managing System Crash Dump Information With the dumpadm Command" on page 9.

Saving Crash Dumps

You can examine the control structures, active tables, memory images of a live or crashed system kernel, and other information about the operation of the kernel by using the mdb utility. Using the mdb utility to its full potential requires a detailed knowledge of the kernel, and is beyond the scope of this manual. For information about using this utility, see the mdb(1) man page.

Managing System Crash Dump Information With the dumpadm Command

Use the dumpadm command to manage system crash dump information in the Oracle Solaris OS.

- The dumpadm command enables you to configure crash dumps of the operating system. The dumpadm configuration parameters include the dump content, dump device, and the directory in which the crash dump files are saved.
- Dump data is stored in a compressed format on the dump device. Kernel crash dump images can be as large as 4 Gbytes, or more. Compressing the data means faster dumping and less disk space required for the dump device.
- The saving of crash dump files is run in the background, when a dedicated dump device, not the swap area, is part of the dump configuration. This means a system that is booting does not wait for the savecore command to complete before going to the next step. On large memory systems, the system can be available before savecore completes. See"Changes to savecore Behavior" on page 7 for potential issues.
- System crash dump files, generated by the savecore command, are saved by default.
- The savecore -L command enables you to get a crash dump of the live running the Oracle Solaris OS. This command is intended for troubleshooting a running system by taking a snapshot of memory during some bad state, such as a transient performance problem or service outage. If the system is up and you can still run some commands, you can execute the savecore -L command to save a snapshot of the system to the dump device, and then immediately write out the crash dump files to your savecore directory. Because the system is still running, you can only use the savecore -L command, if you have configured a dedicated dump device.

Dump configuration parameters are managed by the dumpadm command. The following table describes dumpadm's configuration parameters.

Dump Parameter	Description
dump device	The device that stores dump data temporarily as the system crashes. When the dump device is not the swap area, `savecore` runs in the background, which speeds up the boot process.
`savecore` directory	The directory that stores system crash dump files.
dump content	Type of memory data to dump.
minimum free space	Minimum amount of free space required in the `savecore` directory after saving crash dump files. If no minimum free space has been configured, the default is one Mbyte.

For more information, see `dumpadm`(1M).

How the `dumpadm` Command Works

During system startup, the `dumpadm` command is invoked by the `svc:/system/dumpadm:default` service to configure crash dumps parameters.

Specifically, `dumpadm` initializes the dump device and the dump content through the `/dev/dump` interface.

After the dump configuration is complete, the `savecore` script looks for the location of the crash dump file directory. Then, `savecore` is invoked to check for crash dumps and check the content of the `minfree` file in the crash dump directory.

Managing System Crash Dump Information

This section describes tasks for managing system crash dump information.

Managing System Crash Dump Information (Task Map)

Task	Description	For Instructions
1. Display the current crash dump configuration.	Display the current crash dump configuration by using the `dumpadm` command.	"How to Display the Current Crash Dump Configuration" on page 11

Task	Description	For Instructions
2. Modify the crash dump configuration.	Use the `dumpadm` command to specify the type of data to dump, whether or not the system will use a dedicated dump device, the directory for saving crash dump files, and the amount of space that must remain available after crash dump files are written.	"How to Modify a Crash Dump Configuration" on page 12
3. Examine a crash dump file.	Use the `mdb` command to view crash dump files.	"How to Examine Crash Dump Information" on page 13
4. (Optional) Recover from a full crash dump directory.	The system crashes, but no room is available in the `savecore` directory, and you want to save some critical system crash dump information.	"How to Recover From a Full Crash Dump Directory (Optional)" on page 15
5. (Optional) Disable or enable the saving of crash dump files.	Use the `dumpadm` command to disable or enable the saving the crash dump files. Saving of crash dump files is enabled by default.	"How to Disable or Enable the Saving of Crash Dumps" on page 15

▼ How to Display the Current Crash Dump Configuration

1 Assume the `root` role.

See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Display the current crash dump configuration.

```
# dumpadm
 Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on
```

The preceding example output means:

- The dump content is kernel memory pages.
- Kernel memory will be dumped on a dedicated dump device, `/dev/zvol/dsk/rpool/dump`.
- System crash dump files will be written in the `/var/crash` directory.
- Saving crash dump files is enabled.
- Save crash dumps in compressed format.

▼ How to Modify a Crash Dump Configuration

1 Assume the `root` role.

See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Identify the current crash dump configuration.

```
# dumpadm
 Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on
```

This output identifies the default dump configuration for a system running the Oracle Solaris 11 release.

3 Modify the crash dump configuration.

```
#  /usr/sbin/dumpadm  [-nuy] [-c content-type] [-d dump-device] [-m mink | minm | min%]
[-s savecore-dir] [-r root-dir] [-z on | off]
```

`-c` *content*	Specifies the type of data to dump. Use `kernel` to dump of all kernel memory, `all` to dump all of memory, or `curproc`, to dump kernel memory and the memory pages of the process whose thread was executing when the crash occurred. The default dump content is kernel memory.
`-d` *dump-device*	Specifies the device that stores dump data temporarily as the system crashes. The primary dump device is the default dump device.
`-m` *nnnk* | *nnnm* | *nnn%*	Specifies the minimum free disk space for saving crash dump files by creating a `minfree` file in the current `savecore` directory. This parameter can be specified in Kbytes (`nnnk`), Mbytes (`nnnm`) or file system size percentage (`nnn%`). The `savecore` command consults this file prior to writing the crash dump files. If writing the crash dump files, based on their size, would decrease the amount of free space below the `minfree` threshold, the dump files are not written and an error message is logged. For information about recovering from this scenario, see "How to Recover From a Full Crash Dump Directory (Optional)" on page 15.
`-n`	Specifies that `savecore` should not be run when the system reboots. This dump configuration is not recommended. If system crash information is written to the swap device, and `savecore` is not enabled, the crash dump information is overwritten when the system begins to swap.

`-s`	Specifies an alternate directory for storing crash dump files. In Oracle Solaris 11, the default directory is `/var/crash`.
`-u`	Forcibly updates the kernel dump configuration based on the contents of the `/etc/dumpadm.conf` file.
`-y`	Modifies the dump configuration to automatically execute the `savecore` command upon reboot, which is the default for this dump setting.
`-z on | off`	Modifies the dump configuration to control the operation of the `savecore` command upon reboot. The `on` setting enables the saving of core file in a compressed format. The `off` setting automatically uncompresses the crash dump file. Because crash dump files can be extremely large and therefore require less file system space if they are saved in a compressed forma, the default is `on`.

Example 1–1 Modifying a Crash Dump Configuration

In this example, all of memory is dumped to the dedicated dump device, `/dev/zvol/dsk/rpool/dump`, and the minimum free space that must be available after the crash dump files are saved is 10% of the file system space.

```
# dumpadm
      Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on

# dumpadm -c all -d /dev/zvol/dsk/rpool/dump -m 10%
       Dump content: all pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash (minfree = 5697105KB)
  Savecore enabled: yes
   Save compressed: on
```

▼ How to Examine Crash Dump Information

1 Assume the `root` role.

See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Change to the directory where the crash dump information has been saved. For example:

```
# cd /var/crash
```

If you are unsure of the location of the crash dump, use the `dumpadm` command to determine where the system has been configured to store kernel crash dump files. For example:

```
# /usr/sbin/dumpadm
      Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash
  Savecore enabled: yes
   Save compressed: on
```

3 Examine the crash dump by using the modular debugger utility (`mdb`).

`# /usr/bin/mdb [-k]` *crashdump-file*

`-k` Specifies kernel debugging mode by assuming the file is an operating system crash dump file.

crashdump-file Specifies the operating system crash dump file.

For example:

```
# /usr/bin/mdb -K vmcore.0
```

Or, the command can be specified as follows:

```
# /usr/bin/mdb -k 0
```

4 Display the system crash status, as follows:

```
> ::status
   .
   .
   .
> ::system
   .
   .
   .
```

To use the `::system dcmd` command when examining a kernel crash dump, the core file *must* be a kernel crash dump, *and* the `-k` option must have been specified when starting the `mdb` utility.

5 Quit the `mdb` utility.

```
> $quit
```

Example 1–2 Examining Crash Dump Information

The following example shows sample output from the `mdb` utility, which includes system information and identifies the tunables that are set in this system's `/etc/system` file.

```
# cd /var/crash
# /usr/bin/mdb -k unix.0
```

```
Loading modules: [ unix krtld genunix ip nfs ipc ptm ]
> ::status
debugging crash dump /dev/mem (64-bit) from ozlo
operating system: 5.10 Generic sun4v
> ::system
set ufs_ninode=0x9c40 [0t40000]
set ncsize=0x4e20 [0t20000]
set pt_cnt=0x400 [0t1024]
> $q
```

▼ How to Recover From a Full Crash Dump Directory (Optional)

In this scenario, the system crashes but no room is left in the `savecore` directory, and you want to save some critical system crash dump information.

1. **After the system reboots, log in as the `root` role.**
2. **Clear out the `savecore` directory, typically, `/var/crash/`, by removing existing crash dump files that have already been sent to your service provider.**
   - **Alternatively, you can manually run the `savecore` command to specify an alternate directory that has sufficient disk space.**

     ```
     # savecore [ directory ]
     ```

▼ How to Disable or Enable the Saving of Crash Dumps

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

   See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.
2. **Disable or enable the saving of crash dumps on your system.**

   ```
   # dumpadm -n | -y
   ```

Example 1–3 Disabling the Saving of Crash Dumps

This example illustrates how to disable the saving of crash dumps on your system.

```
# Dump content: all pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash (minfree = 5697105KB)
  Savecore enabled: no
   Save compressed: on
```

Example 1–4 Enabling the Saving of Crash Dumps

This example illustrates how to enable the saving of crash dump on your system.

```
# dumpadm -y
      Dump content: all pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash (minfree = 5697105KB)
  Savecore enabled: yes
   Save compressed: on
```

◆ ◆ ◆ CHAPTER 2

Managing Core Files (Tasks)

This chapter describes how to manage core files with the `coreadm` command.

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

Managing Core Files

Core files are generated when a process or application terminates abnormally. Core files are managed with the `coreadm` command. For example, you can use the `coreadm` command to configure a system so that all process core files are placed in a single system directory. This means it is easier to track problems by examining the core files in a specific directory whenever a process or daemon terminates abnormally.

Configurable Core File Paths

Two following configurable `core` file paths that can be enabled or disabled independently of each other:

- A per-process core file path, which defaults to `core` and is enabled by default. If enabled, the per-process core file path causes a `core` file to be produced when the process terminates abnormally. The per-process path is inherited by a new process from its parent process.

  When generated, a per-process core file is owned by the owner of the process with read/write permissions for the owner. Only the owning user can view this file.
- A global core file path, which defaults to `core` and is disabled by default. If enabled, an *additional* core file with the same content as the per-process core file is produced by using the global core file path.

When generated, a global core file is owned by root, with read/write permissions for root *only*. Non-privileged users cannot view this file.

When a process terminates abnormally, it produces a core file in the current directory by default. If the global core file path is enabled, each abnormally terminating process might produce two files, one in the current working directory, and one in the global core file location.

By default, a `setuid` process does not produce core files using either the global or per-process path.

Expanded Core File Names

If a global `core` file directory is enabled, `core` files can be distinguished from one another by using the variables that are described in the following table.

Variable Name	Variable Definition
`%d`	Executable file directory name, up to a maximum of `MAXPATHLEN` characters
`%f`	Executable file name, up to a maximum of `MAXCOMLEN` characters
`%g`	Effective group ID
`%m`	Machine name (`uname -m`)
`%n`	System node name (`uname -n`)
`%p`	Process ID
`%t`	Decimal value of time(2)
`%u`	Effective user ID
`%z`	Name of the zone in which process is executed (`zonename`)
`%%`	Literal %

For example, if the global core file path is set to:

```
/var/core/core.%f.%p
```

and a `sendmail` process with PID `12345` terminates abnormally, it produces the following `core` file:

```
/var/core/core.sendmail.12345
```

Setting the Core File Name Pattern

You can set a core file name pattern on a global, zone, or per-process basis. In addition, you can set per-process defaults that persist across a system reboot.

For example, the following `coreadm` command sets the default per-process core file pattern. This setting applies to all processes that have not explicitly overridden the default core file pattern. This setting persists across system reboots. For example, the following `coreadm` command sets the global core file pattern for all processes that are started by the `init` process. This pattern will persist across system reboots.

```
# coreadm -i /var/core/core.%f.%p
```

The following `coreadm` command sets the per-process core file name pattern for any processes:

```
# coreadm -p /var/core/core.%f.%p $$
```

The `$$` symbols represent a placeholder for the process ID of the currently running shell. The per-process core file name pattern is inherited by all child processes.

After a global or per-process core file name pattern is set, it must be enabled with the `coreadm -e` command. See the following procedures for more information.

You can set the core file name pattern for all processes that are run during a user's login session by putting the command in a user's initialization file, for example, `.profile`.

Enabling setuid Programs to Produce Core Files

You can use the `coreadm` command to enable or disable `setuid` programs to produce core files for all system processes or on a per-process basis by setting the following paths:

- If the global `setuid` option is enabled, a global core file path allows all `setuid` programs on a system to produce `core` files.
- If the per-process `setuid` option is enabled, a per-process core file path allows specific `setuid` processes to produce `core` files.

By default, both flags are disabled. For security reasons, the global core file path must be a full pathname, starting with a leading `/`. If root disables per-process core files, individual users cannot obtain core files.

The `setuid` core files are owned by root, with read/write permissions for root only. Regular users cannot access these file, even if the process that produced the `setuid` core file is owned by an ordinary user.

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

Managing Core Files (Task Map)

Task	Description	For Instructions
1. Display the current core dump configuration.	Display the current core dump configuration by using the `coreadm` command.	"Displaying the Current Core Dump Configuration" on page 20
2. Modify the core dump configuration.	Modify the core dump configuration to do one of the following: ■ Set a core file name pattern. ■ Enable a per-process core file path. ■ Enable a global core file path.	"How to Set a Core File Name Pattern" on page 20 "How to Enable a Per-Process Core File Path" on page 21 "How to Enable a Global Core File Path" on page 21
3. Examine a core dump file.	Use the `proc` tools to view a core dump file.	"Examining Core Files" on page 22

Displaying the Current Core Dump Configuration

Use the `coreadm` command without any options to display the current core dump configuration.

```
$ coreadm
              global core file pattern:
     global core file content: default
       init core file pattern: core
       init core file content: default
            global core dumps: disabled
       per-process core dumps: enabled
      global setid core dumps: disabled
 per-process setid core dumps: disabled
     global core dump logging: disabled
```

▼ How to Set a Core File Name Pattern

- **Determine whether you want to set a per-process or global core file and select one of the following:**

  a. **Set a per-process file name pattern.**

     ```
     $ coreadm -p $HOME/corefiles/%f.%p $$
     ```

  b. **Assume the `root` role.**

c. **Set a global file name pattern.**

```
# coreadm -g /var/corefiles/%f.%p
```

How to Enable a Per-Process Core File Path

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

   See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Enable a per-process core file path.**

```
# coreadm -e process
```

3. **Display the current process core file path to verify the configuration.**

```
# coreadm $$
1180:   /home/kryten/corefiles/%f.%p
```

How to Enable a Global Core File Path

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

   See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Enable a global core file path.**

```
# coreadm -e global -g /var/core/core.%f.%p
```

3. **Display the current process core file path to verify the configuration.**

```
# coreadm
       global core file pattern: /var/core/core.%f.%p
     global core file content: default
       init core file pattern: core
       init core file content: default
            global core dumps: enabled
       per-process core dumps: enabled
      global setid core dumps: disabled
 per-process setid core dumps: disabled
     global core dump logging: disabled
```

Troubleshooting Core File Problems

Error Message

```
NOTICE: 'set allow_setid_core = 1' in /etc/system is obsolete
NOTICE: Use the coreadm command instead of 'allow_setid_core'
```

Cause

You have an obsolete parameter that allows `setuid` core files in your `/etc/system` file.

Solution

Remove `allow_setid_core=1` from the `/etc/system` file. Then use the `coreadm` command to enable global `setuid` core file paths.

Examining Core Files

The `proc` tools enable you to examine process core files, as well as live processes. The `proc` tools are utilities that can manipulate features of the `/proc` file system.

The `/usr/proc/bin/pstack`, `pmap`, `pldd`, `pflags`, and `pcred` tools can be applied to core files by specifying the name of the core file on the command line, similar to the way you specify a process ID to these commands.

For more information about using `proc` tools to examine core files, see `proc(1)`.

EXAMPLE 2–1 Examining Core Files With `proc` Tools

```
$ ./a.out
Segmentation Fault(coredump)
$ /usr/proc/bin/pstack ./core
core './core' of 19305: ./a.out
 000108c4 main     (1, ffbef5cc, ffbef5d4, 20800, 0, 0) + 1c
 00010880 _start   (0, 0, 0, 0, 0, 0) + b8
```

◆ ◆ ◆ CHAPTER 3

Troubleshooting System and Software Problems (Tasks)

This chapter provides a general overview of troubleshooting software problems, including information about troubleshooting system crashes, managing crash dump information, and viewing and managing system messages.

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

Troubleshooting a System Crash

If a system that is running Oracle Solaris crashes, provide your service provider with as much information as possible, including crash dump files.

What to Do If the System Crashes

The following list describes the most important information to remember in the event of a system crash:

1. Write down the system console messages.
   - If a system crashes, making it run again might seem like your most pressing concern. However, before you reboot the system, examine the console screen for messages. These messages can provide some insight about what caused the crash. Even if the system reboots automatically and the console messages have disappeared from the screen, you might be able to check these messages by viewing the system error log, the `/var/adm/messages` file. For more information about viewing system error log files, see "How to View System Messages" on page 26.

- If you have frequent crashes and cannot determine the cause, gather all of the information you can from the system console or the `/var/adm/messages` file and have it ready for a customer service representative to examine. For a complete list of troubleshooting information to gather for your service provider, see "Troubleshooting a System Crash" on page 23.

2. Check to see if a system crash dump was generated after the system crash. System crash dumps are saved by default. For information about crash dumps, see Chapter 1, "Managing System Crash Information (Tasks)."
3. If the system fails to boot after a system crash, see "Shutting Down and Booting a System for Recovery Purposes" in *Booting and Shutting Down Oracle Solaris 11.1 Systems* for further instructions.

Gathering Troubleshooting Data

Answer the following questions to help isolate the system problem. Use "Troubleshooting a System Crash Checklist" on page 25 for gathering troubleshooting data for a crashed system.

TABLE 3–1 Identifying System Crash Data

Question	Description
Can you reproduce the problem?	This is important because a reproducible test case is often essential for debugging really hard problems. By reproducing the problem, the service provider can build kernels with special instrumentation to trigger, diagnose, and fix the bug.
Are you using any third-party drivers?	Drivers run in the same address space as the kernel, with all the same privileges, so they can cause system crashes if they have bugs.
What was the system doing just before it crashed?	If the system was doing anything unusual like running a new stress test or experiencing higher-than-usual load, that might have led to the crash.
Were there any unusual console messages right before the crash?	Sometimes the system will show signs of distress before it actually crashes; this information is often useful.
Did you add any tuning parameters to the `/etc/system` *file*?	Sometimes tuning parameters, such as increasing shared memory segments so that the system tries to allocate more than it has, can cause the system to crash.
Did the problem start recently?	If so, did the onset of problems coincide with any changes to the system, for example, new drivers, new software, different workload, CPU upgrade, or a memory upgrade.

Troubleshooting a System Crash Checklist

Use this checklist when gathering system data for a crashed system.

Item	Your Data
Is a system crash dump available?	
Identify the operating system release and appropriate software application release levels.	
Identify system hardware.	
Include `prtdiag` output for SPARC systems. Include Explorer output for other systems.	
Are patches installed? If so, include `showrev -p` output.	
Is the problem reproducible?	
Does the system have any third-party drivers?	
What was the system doing before it crashed?	
Were there any unusual console messages right before the system crashed?	
Did you add any parameters to the `/etc/system` file?	
Did the problem start recently?	

Managing System Messages

The following sections describe system messaging features in Oracle Solaris.

Viewing System Messages

System messages display on the console device. The text of most system messages look like this:

[ID *msgid facility.priority*]

For example:

```
[ID 672855 kern.notice] syncing file systems...
```

If the message originated in the kernel, the kernel module name is displayed. For example:

```
Oct 1 14:07:24 mars ufs: [ID 845546 kern.notice] alloc: /: file system full
```

When a system crashes, it might display a message on the system console like this:

```
panic: error message
```

Less frequently, this message might be displayed instead of the panic message:

```
Watchdog reset !
```

The error logging daemon, `syslogd`, automatically records various system warnings and errors in message files. By default, many of these system messages are displayed on the system console and are stored in the `/var/adm` directory. You can direct where these messages are stored by setting up system message logging. For more information, see "Customizing System Message Logging" on page 28. These messages can alert you to system problems, such as a device that is about to fail.

The `/var/adm` directory contains several message files. The most recent messages are in `/var/adm/messages` file (and in `messages.*`), and the oldest are in the `messages.3` file. After a period of time (usually every ten days), a new `messages` file is created. The `messages.0` file is renamed `messages.1`, `messages.1` is renamed `messages.2`, and `messages.2` is renamed `messages.3`. The current `/var/adm/messages.3` file is deleted.

Because the `/var/adm` directory stores large files containing messages, crash dumps, and other data, this directory can consume lots of disk space. To keep the `/var/adm` directory from growing too large, and to ensure that future crash dumps can be saved, you should remove unneeded files periodically. You can automate this task by using the `crontab` file. For more information about automating this task, see "How to Delete Crash Dump Files" in *Oracle Solaris 11.1 Administration: Devices and File Systems* and Chapter 4, "Scheduling System Tasks (Tasks)," in *Managing System Information, Processes, and Performance in Oracle Solaris 11.1*.

▼ How to View System Messages

- **Display recent messages generated by a system crash or reboot by using the `dmesg` command.**

  ```
  $ dmesg
  ```

  Or, use the `more` command to display one screen of messages at a time.

  ```
  $ more /var/adm/messages
  ```

Example 3–1 Viewing System Messages

The following example shows output from the `dmesg` command on an Oracle Solaris 10 system.

```
$ dmesg
Mon Sep 13 14:33:04 MDT 2010
Sep 13 11:06:16 sr1-ubrm-41 svc.startd[7]: [ID 122153 daemon.warning] ...
Sep 13 11:12:55 sr1-ubrm-41 last message repeated 398 times
Sep 13 11:12:56 sr1-ubrm-41 svc.startd[7]: [ID 122153 daemon.warning] ...
```

```
Sep 13 11:15:16 sr1-ubrm-41 last message repeated 139 times
Sep 13 11:15:16 sr1-ubrm-41 xscreensaver[25520]:  ,,,
Sep 13 11:15:16 sr1-ubrm-41 xscreensaver[25520]: ...
Sep 13 11:15:17 sr1-ubrm-41 svc.startd[7]: [ID 122153 daemon.warning]...
.
.
.
```

See Also For more information, see the `dmesg(1M)` man page.

System Log Rotation

System log files are rotated by the `logadm` command from an entry in the root `crontab` file. The `/usr/lib/newsyslog` script is no longer used.

The system log rotation is defined in the `/etc/logadm.conf` file. This file includes log rotation entries for processes such as `syslogd`. For example, one entry in the `/etc/logadm.conf` file specifies that the `/var/log/syslog` file is rotated weekly unless the file is empty. The most recent `syslog` file becomes `syslog.0`, the next most recent becomes `syslog.1`, and so on. Eight previous `syslog` log files are kept.

The `/etc/logadm.conf` file also contains time stamps of when the last log rotation occurred.

You can use the `logadm` command to customize system logging and to add additional logging in the `/etc/logadm.conf` file as needed.

For example, to rotate the Apache access and error logs, use the following commands:

```
# logadm -w /var/apache/logs/access_log -s 100m
# logadm -w /var/apache/logs/error_log -s 10m
```

In this example, the Apache `access_log` file is rotated when it reaches 100 MB in size, with a `.0`, `.1`, (and so on) suffix, keeping 10 copies of the old `access_log` file. The `error_log` is rotated when it reaches 10 MB in size with the same suffixes and number of copies as the `access_log` file.

The `/etc/logadm.conf` entries for the preceding Apache log rotation examples look similar to the following:

```
# cat /etc/logadm.conf
.
.
.
/var/apache/logs/error_log -s 10m
/var/apache/logs/access_log -s 100m
```

For more information, see `logadm(1M)`.

You can use the `logadm` command as superuser or by assuming an equivalent role (with Log Management rights). With RBAC, you can grant non-root users the privilege of maintaining log files by providing access to the `logadm` command.

For example, add the following entry to the `/etc/user_attr` file to grant user `andy` the ability to use the `logadm` command:

```
andy::::profiles=Log Management
```

Customizing System Message Logging

You can capture additional error messages that are generated by various system processes by modifying the `/etc/syslog.conf` file. By default, the `/etc/syslog.conf` file directs many system process messages to the `/var/adm/messages` files. Crash and boot messages are stored here as well. To view `/var/adm` messages, see "How to View System Messages" on page 26.

The `/etc/syslog.conf` file has two columns separated by tabs:

facility.level ... action

facility.level — A *facility* or system source of the message or condition. May be a comma-separated listed of facilities. Facility values are listed in Table 3–2. A *level*, indicates the severity or priority of the condition being logged. Priority levels are listed in Table 3–3.

Do not put two entries for the same facility on the same line, if the entries are for different priorities. Putting a priority in the syslog file indicates that all messages of that all messages of that priority or higher are logged, with the last message taking precedence. For a given facility and level, `syslogd` matches all messages for that level and all higher levels.

action — The action field indicates where the messages are forwarded.

The following example shows sample lines from a default `/etc/syslog.conf` file.

```
user.err                                        /dev/sysmsg
user.err                                        /var/adm/messages
user.alert                                      'root, operator'
user.emerg                                      *
```

This means the following user messages are automatically logged:

- User errors are printed to the console and also are logged to the `/var/adm/messages` file.
- User messages requiring immediate action (`alert`) are sent to the `root` and `operator` users.
- User emergency messages are sent to individual users.

Note – Placing entries on separate lines might cause messages to be logged out of order if a log target is specified more than once in the `/etc/syslog.conf` file. Note that you can specify multiple selectors in a single line entry, each separated by a semicolon.

The most common error condition sources are shown in the following table. The most common priorities are shown in Table 3–3 in order of severity.

TABLE 3–2 Source Facilities for `syslog.conf` Messages

Source	Description
`kern`	The kernel
`auth`	Authentication
`daemon`	All daemons
`mail`	Mail system
`lp`	Spooling system
`user`	User processes

Note – The number of `syslog` facilities that can be activated in the `/etc/syslog.conf` file is unlimited.

TABLE 3–3 Priority Levels for `syslog.conf` Messages

Priority	Description
`emerg`	System emergencies
`alert`	Errors requiring immediate correction
`crit`	Critical errors
`err`	Other errors
`info`	Informational messages
`debug`	Output used for debugging
`none`	This setting doesn't log output

▼ How to Customize System Message Logging

1. **Assume the `root` role or a role that has the `solaris.admin.edit/etc/syslog.conf` authorization assigned to it.**

   See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Use the `pfedit` command to edit the `/etc/syslog.conf` file, adding or changing message sources, priorities, and message locations according to the syntax described in `syslog.conf`(4).**

   ```
   $ pfedit /etc/syslog.conf
   ```

3. **Save the changes.**

Example 3–2 Customizing System Message Logging

This sample `/etc/syslog.conf` `user.emerg` facility sends user emergency messages to `root` *and* individual users.

```
user.emerg                                      'root, *'
```

Enabling Remote Console Messaging

The following new console features improve your ability to troubleshoot remote systems:

- The `consadm` command enables you to select a serial device as an *auxiliary* (or remote) console. Using the `consadm` command, a system administrator can configure one or more serial ports to display redirected console messages and to host `sulogin` sessions when the system transitions between run levels. This feature enables you to dial in to a serial port with a modem to monitor console messages and participate in `init` state transitions. (For more information, see `sulogin`(1M) and the step-by-step procedures that follow.)

  While you can log in to a system using a port configured as an auxiliary console, it is primarily an output device displaying information that is also displayed on the default console. If boot scripts or other applications read and write to and from the default console, the write output displays on all the auxiliary consoles, but the input is only read from the default console. For more information about using the `consadm` command during an interactive login session, see "Guidelines for Using the `consadm` Command During an Interactive Login Session" on page 32.

- Console output now consists of kernel and `syslog` messages written to a new pseudo device, `/dev/sysmsg`. In addition, `rc` script startup messages are written to `/dev/msglog`. Previously, all of these messages were written to `/dev/console`.

Scripts that direct console output to `/dev/console` need to be changed to `/dev/msglog` if you want to see script messages displayed on the auxiliary consoles. Programs referencing `/dev/console` should be explicitly modified to use `syslog()` or `strlog()` if you want messages to be redirected to an auxiliary device.

- The `consadm` command runs a daemon to monitor auxiliary console devices. Any display device designated as an auxiliary console that disconnects, hangs up or loses carrier, is removed from the auxiliary console device list and is no longer active. Enabling one or more auxiliary consoles does not disable message display on the default console; messages continue to display on `/dev/console`.

Using Auxiliary Console Messaging During Run Level Transitions

Keep the following in mind when using auxiliary console messaging during run level transitions:

- Input cannot come from an auxiliary console if user input is expected for an `rc` script that is run when a system is booting. The input must come from the default console.
- The `sulogin` program, invoked by `init` to prompt for the superuser password when transitioning between run levels, has been modified to send the superuser password prompt to each auxiliary device in addition to the default console device.
- When the system is in single-user mode and one or more auxiliary consoles are enabled using the `consadm` command, a console login session runs on the first device to supply the correct superuser password to the `sulogin` prompt. When the correct password is received from a console device, `sulogin` disables input from all other console devices.
- A message is displayed on the default console and the other auxiliary consoles when one of the consoles assumes single-user privileges. This message indicates which device has become the console by accepting a correct superuser password. If there is a loss of carrier on the auxiliary console running the single-user shell, one of two actions might occur:
  - If the auxiliary console represents a system at run level 1, the system proceeds to the default run level.
  - If the auxiliary console represents a system at run level S, the system displays the `ENTER RUN LEVEL (0-6, s or S):` message on the device where the `init s` or `shutdown` command had been entered from the shell. If there isn't any carrier on that device either, you will have to reestablish carrier and enter the correct run level. The `init` or `shutdown` command will not re-display the run-level prompt.
- If you are logged in to a system using a serial port, and an `init` or `shutdown` command is issued to transition to another run level, the login session is lost whether this device is the auxiliary console or not. This situation is identical to releases without auxiliary console capabilities.
- Once a device is selected as an auxiliary console using the `consadm` command, it remains the auxiliary console until the system is rebooted or the auxiliary console is unselected. However, the `consadm` command includes an option to set a device as the auxiliary console across system reboots. (See the following procedure for step-by-step instructions.)

Guidelines for Using the `consadm` Command During an Interactive Login Session

If you want to run an interactive login session by logging in to a system using a terminal that is connected to a serial port, and then using the `consadm` command to see the console messages from the terminal, note the following behavior:

- If you use the terminal for an interactive login session while the auxiliary console is active, the console messages are sent to the `/dev/sysmsg` or `/dev/msglog` devices.
- While you issue commands on the terminal, input goes to your interactive session and not to the default console (`/dev/console`).
- If you run the `init` command to change run levels, the remote console software kills your interactive session and runs the `sulogin` program. At this point, input is accepted only from the terminal and is treated like it's coming from a console device. This allows you to enter your password to the `sulogin` program as described in "Using Auxiliary Console Messaging During Run Level Transitions" on page 31.

  Then, if you enter the correct password on the (auxiliary) terminal, the auxiliary console runs an interactive `sulogin` session, locks out the default console and any competing auxiliary console. This means the terminal essentially functions as the system console.
- From here you can change to run level 3 or go to another run level. If you change run levels, `sulogin` runs again on all console devices. If you exit or specify that the system should come up to run level 3, then all auxiliary consoles lose their ability to provide input. They revert to being display devices for console messages.

  As the system is coming up, you must provide information to `rc` scripts on the default console device. After the system comes back up, the `login` program runs on the serial ports and you can log back into another interactive session. If you've designated the device to be an auxiliary console, you will continue to get console messages on your terminal, but all input from the terminal goes to your interactive session.

▼ How to Enable an Auxiliary (Remote) Console

The `consadm` daemon does not start monitoring the port until after you add the auxiliary console with the `consadm` command. As a security feature, console messages are only redirected until carrier drops, or the auxiliary console device is unselected. This means carrier must be established on the port before you can successfully use the `consadm` command.

For more information about enabling an auxiliary console, see the `consadm`(1m) man page.

1 Log in to the system and assume the `root` role.

See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Enable the auxiliary console.

```
# consadm -a devicename
```

3. **Verify that the current connection is the auxiliary console.**

   ```
   # consadm
   ```

Example 3–3 Enabling an Auxiliary (Remote) Console

```
# consadm -a /dev/term/a
# consadm
 /dev/term/a
```

▼ How to Display a List of Auxiliary Consoles

1. **Log in to the system and assume the `root` role.**

   See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Select one of the following steps:**

   a. **Display the list of auxiliary consoles.**

      ```
      # consadm
      /dev/term/a
      ```

   b. **Display the list of persistent auxiliary consoles.**

      ```
      # consadm -p
      /dev/term/b
      ```

▼ How to Enable an Auxiliary (Remote) Console Across System Reboots

1. **Log in to the system and assume the `root` role.**

   See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Enable the auxiliary console across system reboots.**

   `# consadm -a -p` *devicename*

   This adds the device to the list of persistent auxiliary consoles.

3. **Verify that the device has been added to the list of persistent auxiliary consoles.**

   ```
   # consadm
   ```

Example 3–4 Enabling an Auxiliary (Remote) Console Across System Reboots

```
# consadm -a -p /dev/term/a
# consadm
/dev/term/a
```

▼ How to Disable an Auxiliary (Remote) Console

1. **Log in to the system and assume the `root` role.**

   See "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Select one of the following steps:**

   a. **Disable the auxiliary console.**

      ```
      # consadm -d devicename
      ```

      or

   b. **Disable the auxiliary console and remove it from the list of persistent auxiliary consoles.**

      ```
      # consadm -p -d devicename
      ```

3. **Verify that the auxiliary console has been disabled.**

   ```
   # consadm
   ```

Example 3–5 Disabling an Auxiliary (Remote) Console

```
# consadm -d /dev/term/a
# consadm
```

Troubleshooting File Access Problems

Users frequently experience problems, and call on a system administrator for help, because they cannot access a program, a file, or a directory that they could previously use.

Whenever you encounter such a problem, investigate one of three areas:

- The user's search path may have been changed, or the directories in the search path may not be in the proper order.
- The file or directory may not have the proper permissions or ownership.
- The configuration of a system accessed over the network may have changed.

This chapter briefly describes how to recognize problems in each of these three areas and suggests possible solutions.

Solving Problems With Search Paths (Command not found)

A message of `Command not found` indicates one of the following:

- The command is not available on the system.
- The command directory is not in the search path.

To fix a search path problem, you need to know the pathname of the directory where the command is stored.

If the wrong version of the command is found, a directory that has a command of the same name is in the search path. In this case, the proper directory may be later in the search path or may not be present at all.

You can display your current search path by using the `echo $PATH` command.

Use the `type` command to determine whether you are running the wrong version of the command. For example:

```
$ type acroread
acroread is /usr/bin/acroread
```

▼ How to Diagnose and Correct Search Path Problems

1 Display the current search path to verify that the directory for the command is not in your path or that it isn't misspelled.

```
$ echo $PATH
```

2 Check the following:

- Is the search path correct?
- Is the search path listed before other search paths where another version of the command is found?
- Is the command in one of the search paths?

If the path needs correction, go to step 3. Otherwise, go to step 4.

3 Add the path to the appropriate file, as shown in this table.

Shell	File	Syntax	Notes
bash and ksh93	`$HOME/.profile`	`$ PATH=`*`$HOME/bin:/sbin:/usr/local/bin ...`* `$ export PATH`	A colon separates path names.

4 **Activate the new path as follows:**

Shell	Path Location	Command to Activate The Path
bash and ksh93	.profile	. $HOME/.profile
	.login	hostname$ source $HOME/.login

5 **Verify the new path.**

$ **which** *command*

Example 3–6 Diagnosing and Correcting Search Path Problems

This example shows that the `mytool` executable is not in any of the directories in the search path using the `type` command.

```
$ mytool
 -bash: mytool: command not found
 $ type mytool
 -bash: type: mytool: not found
 $ echo $PATH
 /usr/bin:
 $ vi $HOME/.profile
 (Add appropriate command directory to the search path)
 $ . $HOME/.profile
 $ mytool
```

If you cannot find a command, look at the man page for its directory path.

Changing File and Group Ownerships

Frequently, file and directory ownerships change because someone edited the files as superuser. When you create home directories for new users, be sure to make the user the owner of the dot (.) file in the home directory. When users do not own " . " they cannot create files in their own home directory.

Access problems can also arise when the group ownership changes or when a group of which a user is a member is deleted from the `/etc/group` database.

For information about how to change the permissions or ownership of a file that you are having problems accessing, see Chapter 7, "Controlling Access to Files (Tasks)," in *Oracle Solaris 11.1 Administration: Security Services*.

Solving File Access Problems

When users cannot access files or directories that they previously could access, the permissions or ownership of the files or directories probably has changed.

Recognizing Problems With Network Access

If users have problems using the `rcp` remote copy command to copy files over the network, the directories and files on the remote system may have restricted access by setting permissions. Another possible source of trouble is that the remote system and the local system are not configured to allow access.

See "Strategies for NFS Troubleshooting" in *Managing Network File Systems in Oracle Solaris 11.1* for information about problems with network access and problems with accessing systems through AutoFS.

◆ ◆ ◆ CHAPTER 4

Troubleshooting Miscellaneous System and Software Problems (Tasks)

This chapter describes miscellaneous system and software problems that might occur occasionally and are relatively easy to fix. The troubleshooting process usually includes solving problems that are not related to a specific software application or topic, such as unsuccessful reboots and full file systems.

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

What to Do If Rebooting Fails

If the system does not reboot completely, or if the system reboots and then crashes again, there might be a software or hardware problem that is preventing the system from booting successfully.

Cause of System Not Booting	How to Fix the Problem
The system can't find `/platform/‘uname -m‘/kernel/sparcv9/unix`.	You may need to change the `boot-device` setting in the PROM on a SPARC based system. For information about changing the default boot device, see "Displaying and Setting Boot Attributes" in *Booting and Shutting Down Oracle Solaris 11.1 Systems*.

Cause of System Not Booting	How to Fix the Problem
The Oracle Solaris boot archive has become corrupted. Or, the SMF boot archive service has failed. An error message is displayed if you run the `svcs -x` command.	Create a second boot environment that is a backup of the primary boot environment. In the event the primary boot environment is not bootable, boot the backup boot environment. Alternatively, you can boot from the live CD or USB media.
There is an invalid entry in the `/etc/passwd` file.	For information about recovering from an invalid `passwd` file, see "How to Boot From Media to Resolve an Unknown root Password" in *Booting and Shutting Down Oracle Solaris 11.1 Systems*.
The x86 boot loader (GRUB) is damaged. Or, the GRUB menu is missing or has become corrupt.	For information about recovering from a damaged x86 boot loader or a missing or corrupt GRUB menu, see "How to Boot From Media to Resolve a Problem With the GRUB Configuration That Prevents the System From Booting" in *Booting and Shutting Down Oracle Solaris 11.1 Systems*.
There's a hardware problem with a disk or another device.	Check the hardware connections: ■ Make sure the equipment is plugged in. ■ Make sure all the switches are set properly. ■ Look at all the connectors and cables, including the Ethernet cables. ■ If all these steps fail, turn off the power to the system, wait 10 to 20 seconds, and then turn on the power again.

If none of the above suggestions solve the problem, contact your local service provider.

What to Do If You Forgot the Root Password or Problem That Prevents System From Booting

If you forget the `root` password or experience another problem that prevents the system from booting, do the following:

- Stop the system.
- Follow the directions in "How to Boot From Media to Resolve an Unknown root Password" in *Booting and Shutting Down Oracle Solaris 11.1 Systems*.
- If the `root` password is the problem, remove the root password from the `/etc/shadow` file.
- Reboot the system.
- Log in and set the `root` password.

What to Do If a System Hang Occurs

A system can freeze or hang rather than crash completely if some software process is stuck. Follow these steps to recover from a hung system.

1. Determine whether the system is running a window environment and follow these suggestions. If these suggestions do not solve the problem, go to step 2.
   - Make sure the pointer is in the window where you are typing the commands.
   - Press Control-q in case the user accidentally pressed Control-s, which freezes the screen. Control-s freezes only the window, not the entire screen. If a window is frozen, try using another window.
   - If possible, log in remotely from another system on the network. Use the `pgrep` command to look for the hung process. If it looks like the window system is hung, identify the process and kill it.
2. Press Control-\ to force quit the running program and (probably) write out a `core` file.
3. Press Control-c to interrupt the program that might be running.
4. Log in remotely and attempt to identify and kill the process that is hanging the system.
5. Log in remotely, assume the `root` role and then reboot the system.
6. If the system still does not respond, force a crash dump and reboot. For information about forcing a crash dump and booting, see "Forcing a Crash Dump and Reboot of the System" in *Booting and Shutting Down Oracle Solaris 11.1 Systems*.
7. If the system still does not respond, turn the power off, wait a minute or so, then turn the power back on.
8. If you cannot get the system to respond at all, contact your local service provider for help.

What to Do If a File System Fills Up

When the root (`/`) file system or any other file system fills up, you will see the following message in the console window:

```
.... file system full
```

There are several reasons why a file system fills up. The following sections describe several scenarios for recovering from a full file system.

File System Fills Up Because a Large File or Directory Was Created

Reason Error Occurred	How to Fix the Problem
Someone accidentally copied a file or directory to the wrong location. This also happens when an application crashes and writes a large `core` file to the file system.	Log in and assume the `root` role, then use the `ls -tl` command in the specific file system to identify which large file is newly created and then remove it.

A TMPFS File System Is Full Because the System Ran Out of Memory

Reason Error Occurred	How to Fix the Problem
This can occur if `TMPFS` is trying to write more than it is allowed or some current processes are using a lot of memory.	For information about recovering from `tmpfs`-related error messages, see the `tmpfs(7FS)` man page.

What to Do If File ACLs Are Lost After Copy or Restore

Reason Error Occurred	How to Fix the Problem
If files or directories with ACLs are copied or restored into the `/tmp` directory, the ACL attributes are lost. The `/tmp` directory is usually mounted as a temporary file system, which doesn't support UFS file system attributes such as ACLs.	Copy or restore files into the `/var/tmp` directory instead.

Index