Troubleshooting System Administration Issues in Oracle® Solaris 11.3

Part No: E54840
October 2017

ORACLE

Troubleshooting System Administration Issues in Oracle Solaris 11.3

Part No: E54840

Access to Oracle Support

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

Référence: E54840

Accès aux services de support Oracle

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

Contents

Using This Documentation

- **Overview** – Describes Oracle Solaris troubleshooting issues on both SPARC and x86 platforms
- **Audience** – System administrators using the Oracle Solaris 11 release
- **Required knowledge** – Experience administering UNIX systems

Product Documentation Library

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

Feedback

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

♦ ♦ ♦ CHAPTER 1

Troubleshooting System Crashes

This chapter describes how to prepare for and troubleshoot a system crash in the Oracle Solaris OS.

This chapter covers the following topics:

Crash Files Now Deferred Until After Reboot in Oracle Solaris 11.3

Starting with Oracle Solaris 11.3, when a system crashes, the crash dump files might be preserved in memory until after the system reboots. When the system is rebooting, the crash dump files are extracted from memory to the file system that is defined in the dump configuration. Once these files are written, the system automatically reboots to normal multi-user configuration. This process is referred to as a *deferred dump*. Deferred dumps enable systems to return to a running state more quickly after a kernel panic. In addition, deferred dumps specifically benefit systems, such as the SPARC M7 series servers, which may ship without a local disk.

Caution - The first reboot performs the deferred dump. On x86 systems, a second reboot might automatically occur to eliminate any possible performance issues caused by memory fragmentation. Do not interrupt this process. The system is ready for use after the second reboot.

Deferred Dump System Messages

The deferred dump process includes the following system messages:

- When the system is performing a deferred dump to memory, a `Preserving kernel image in RAM` message with a progress bar appears.
- When the system reboots after the panic, a `Verified previous kernel image` notice followed by a `Reconciling deferred dump` message with a progress bar appears.
- When `savecore` is saving the crash dump files to the file system, an `Extracting crash dump` message with a progress bar appears.

Deferred Dump Support

On x86 systems, deferred dumps work in conjunction with the default *fast reboot* feature in Oracle Solaris. Therefore, a deferred dump on an x86 system can function only on platforms supporting fast reboot. For further information about using fast reboots, see "Accelerating the Reboot Process" in *Booting and Shutting Down Oracle Solaris 11.3 Systems*.

Deferred dumps are enabled by default on SPARC T4 systems that support at least `Sys Firmware 8.7` or `Sys Firmware 9.4` and later, and on Fujitsu M10 systems that support XCP 2290 or later with sufficient memory to preserve crash dump files until the system reboots.

Even with deferred dump, you should still ensure that the configured dump device has sufficient space to store crash dump information. This is in case a live system crash dump using `savecore -L` is necessary. Also, crash dump information will be automatically written to the dump device if possible without being held in memory. The writing operation occurs in the following situations:

- The platform does not support a fast reboot, or the fast reboot fails.
- The `savecore` utility is disabled or fails.
- The system is using kernel zones.

For further information about preparing for and responding to system crashes, review this chapter and see the `dumpadm(1M)` man page.

About System Crashes

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 it preserves a copy of

its physical memory in RAM or 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 memory or from the dump device and write the saved crash dump files to your `savecore` directory. These saved files provide invaluable information to aid in diagnosing the problem.

Note - The term "crash dump" refers to the overall result of this process, including the set of crash dump files, where they are located, and how these files are organized and formatted.

System Crash Dump Files

The `savecore` command runs automatically after a system crash to retrieve the crash dump information from memory or from the dump device and writes the information into a set of files. Afterwards, the `savecore` command can be invoked on the same system or another system to expand the compressed crash dump files.

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 the `/var/crash/` directory, which is a predetermined default directory. The saving of crash dump files is enabled by default.

Restructured Files

Beginning with the Oracle Solaris 11.2 release, the contents of kernel crash dump files are divided into multiple new files based on their contents. This method enables more granularity in configuration so the files can more easily be accessed and studied.

The crash dump information is written to the set of `vmdump-`*section*`.`*n* files. The section value is the name of a file section that contains a specific kind of dump information. The *n* value is an integer which increments every time `savecore` is run to copy a crash dump and a new crash dump is found on the dump device. Possible files include:

- `vmdump-proc.`*n* – Dump file with compressed process pages
- `vmdump-zfs.`*n*– Dump file with compressed ZFS metadata
- `vmdump-other.`*n* – Dump file with other pages

Kernel crash dumps were previously stored in `vmdump.n`, `unix.n`, and `vmcore.n`.

The `vmdump.`*n* and `vmcore` files store kernel pages metadata and kernel pages data in compressed or uncompressed form, respectively.

For further information, see the `dumpadm`(1M) and `savecore`(1M) man pages.

`dumpadm` and `savecore` Commands

The `dumpadm` and `savecore` utilities configure and manage the creation of a crash dump as follows:

1. During system startup, the `dumpadm` command is invoked by the `svc:/system/dumpadm:default` service to configure crash dump parameters. It initializes the dump device and the dump content through the `/dev/dump` interface.
2. After the dump configuration is complete, `savecore` is invoked. It checks for crash dumps either in the RAM or in the dump device, and checks the content of the `minfree` file in the crash dump directory. System crash dump files generated by the `savecore` command are saved by default.
3. Dump data is stored in a compressed format on the dump device. Kernel crash dump images can be as large as 128GB or more. Compressing the data means faster dumping and less disk space required for the dump device.
4. By default, the installer will create a dedicated dump service. The system will then wait for the `savecore` command to complete before going on to the next step. On large memory systems, the system can become available before `savecore` completes.

The `savecore -L` command enables an administrator to get a crash dump of a system currently running the Oracle Solaris OS. This command is intended for troubleshooting a running system by taking an image of memory during some bad state, such as a transient performance problem or service outage. Note that this image of memory is imperfect due to changes that occur while the system is running. If the system is up and you can still run some commands, you can execute the `savecore -L` command to save the image of the system to the dump device and then immediately write out the crash dump files to your `savecore` directory. You can use the `savecore -L` command only if you have configured a dedicated dump device.

Configuring Your System for Crash Dumps

This section describes the tasks for managing crash dump procedures for your system.

Keep the following 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 "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.
- Starting with the Oracle Solaris 11.3 release, when a system crashes, a copy of its physical memory can be held in RAM until the system reboots, then written to the file system. This process is called a *deferred dump*. If a deferred dump is not possible, the physical memory is written to a dump device during a crash. See "Crash Files Now Deferred Until After Reboot in Oracle Solaris 11.3" on page 11.
- Do not disable the option of saving system crash dump files on the system. System crash dump files provide an invaluable way to determine what is causing the system to crash.
- Dedicated ZFS volumes are used for swap and dump areas. For instructions, see "Managing ZFS Swap and Dump Devices" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

Displaying the Current Crash Dump Configuration

To display the current crash dump configuration, assume the `root` role and issue the `dumpadm` command with no arguments.

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

This example output shows the following configuration:

- The dump content is kernel memory pages with ZFS metadata.
- Kernel memory will either be held in memory until a reboot or dumped on a dedicated dump device, `/dev/zvol/dsk/rpool/dump`.
- System crash dump files will be saved in the `/var/crash` directory.
- Saving crash dump files is enabled.
- Crash dump files are saved in compressed format.

Modifying the Configuration for Crash Dumps

To modify the crash dump configuration, assume the `root` role and use the `dumpadm` command.

The syntax for the `dumpadm` command is as follows:

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

-c *content-type*
: Specifies the type of data to dump. The possible values are as follows:

  - `kernel` which dumps the kernel memory pages only
  - `all` which dumps all memory pages
  - `curproc` which dumps kernel memory and the memory pages of the process whose thread was executing when the crash occurred
  - `allproc` which dumps kernel memory pages and all process pages
  - `zfs` which dumps kernel pages that store ZFS metadata

  The default dump content is kernel memory with ZFS metadata. For example:

  ```
  # dumpadm -c kernel
  # dumpadm -c +zfs
  # dumpadm -c -zfs
  # dumpadm -c curproc+zfs
  ```

-d *dump-device*
: Specifies the device that stores dump data temporarily when the system crashes. The primary dump device is the default dump device. When the dump device is not the swap area, `savecore` runs in the background, which speeds up the boot process.

Note - A deferred dump will store dump data in RAM temporarily instead of using a dump device. However, you should specify a dump device in case the deferred dump fails, cannot be performed, or the `savecore` directory does not have enough space to write the crash dump files directly into the file system.

-e
: Prints an estimate of disk space required for storing a compressed crash dump. The value is computed using the current configuration and currently running system.

-m *min*k | *min*m | *min*%
: 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 KB (`mink`), MB (`minm`) or file system size percentage (`min%`). If no minimum free space has been configured, the default is 1MB.

  The `savecore` command consults this file prior to writing the crash dump files. If writing the crash dump files would decrease the amount of

free space below the `minfree` threshold due to their size, the dump files are not written and an error message is logged. For information about recovering from this scenario, see "Saving Data When the Crash Dump Directory Is Full" on page 23.

`-n` — Specifies that `savecore` will not be run automatically when the system reboots. This dump configuration is not recommended because the system will attempt to preserve a crash dump image in memory.

`-p` — Produces machine-readable output.

`-s` *savecore-dir* — 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 a 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 format, the default is `on`.

EXAMPLE 1 Modifying a Crash Dump Configuration

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

```
# dumpadm
      Dump content: kernel with ZFS metadata
       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 = 5935131KB)
  Savecore enabled: yes
   Save compressed: on

# dumpadm -n
       Dump content: all pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash (minfree = 5935131KB)
  Savecore enabled: no
   Save compressed: on

# dumpadm -y
       Dump content: all pages
       Dump device: /dev/zvol/dsk/rpool/dump(dedicated)
Savecore directory: /var/crash (minfree = 5935131KB)
      Savecore enabled: yes
        Save compressed: on
```

EXAMPLE 2 Disabling the Saving of Crash Dumps

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

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

Caution - Oracle Solaris strongly recommends that you do not disable the saving of crash dumps. Crash dumps provide an invaluable way to determine what causes your system to crash.

EXAMPLE 3 Enabling the Saving of Crash Dumps

This example shows how to enable the saving of crash dumps 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
```

Troubleshooting After a System Crashes

If an Oracle Solaris system crashes, you will want to provide as much information as possible, including general system information and the specific information provided in the crash dump files.

To troubleshoot an existing crash dump, see the following:

What to Do If the System Crashes

In the event of a system crash:

1. Note 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 `/var/adm/messages` system error log file. For more information about viewing system error log files, see "Viewing System Messages and Logging" on page 45.
2. 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 support representative to examine.

   For a complete list of troubleshooting information to gather for your service provider, see "Checklist for Troubleshooting a System Crash" on page 21.
3. Check to see if a system crash dump was generated after the system crash.

Caution - Do not remove important system crash information until it has been sent to your customer support representative.

4. 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 11.3 Systems* for further instructions.

Examining Crash Dump Information

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.

Note - The following procedure provides only a limited example of how to use 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 further information about using this utility, see the *Oracle Solaris Modular Debugger Guide* and `mdb(1)` man page.

▼ How to Examine Crash Dump Information

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

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

   ```
   # 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. The following sample output shows that the default directory location has not been changed:

   ```
   # /usr/sbin/dumpadm
         Dump content: kernel with ZFS metadata
          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
```

The command can also be specified as follows:

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

4. Display the system crash status.

```
> ::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 4 Examining Crash Dump Information

This example shows sample output from the `mdb` utility, which includes system information and identifies the tunables that are set in the `/etc/system` file of the system.

```
# 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
```

Checklist for Troubleshooting a System Crash

Answer the questions in the following checklist to help isolate the system problem and to prepare to consult with your Oracle Support providers.

Item	Your Data
Is a system crash dump available?	
Identify the operating system release and appropriate software application release levels. The user should receive information for `more/etc/release` or `pkg info entire`.	
Identify system hardware.	
Include the `prtdiag` output for SPARC and x86 systems. Include Explorer output, which is often requested by services for all systems.	
Are patches installed? Because Oracle Solaris 11 does not show `showrev -p` output, include information about installed SRUs & IDRs instead.	
Is the problem reproducible? A reproducible test case is often essential for debugging difficult problems. By reproducing the problem, the service provider can build kernels with special instrumentation to trigger, diagnose, and fix the bug.	
Does the system have any third-party drivers? Drivers run in the same address space as the kernel. With all the same privileges, they can cause system crashes if they have bugs.	
What was the system doing before it crashed? Unusual circumstances like running a new stress test or experiencing a load that is higher than usual might have led to the crash.	
Did any unusual console messages display right before the system crashed? Sometimes the system will show signs of distress before it actually crashes; this information is often useful.	
Did you add any parameters to the `/etc/system` file? `/etc/system` is now supplemented by `/etc/system.d/*` starting with Oracle Solaris 11.2.	
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.	

Saving Data When the Crash Dump Directory Is Full

- If system crashes with no room left in the `savecore` directory and you want to save some critical system crash dump information, use one of the following methods.

  After the system reboots, log in as the root role. Remove existing crash dump files that have already been sent to your service provider from the `savecore` directory.

  Note - The savecore directory is usually located at `/var/crash`.

- Alternately, after the system reboots, log in as the root role. Manually run the `savecore` command and specify an alternate directory that has sufficient disk space:

  ```
  # savecore directory
  ```

Managing Incidents With Oracle Enterprise Manager Ops Center

It is necessary to manage incidents for physical and virtual operating systems, servers, and storage devices within a large deployment. Rather than just monitoring incidents within individual systems, you can use the comprehensive system management solutions available in the Oracle Enterprise Manager Ops Center.

Using the Enterprise Manager Ops Center, you can set up alerts that tell you when part of your data center is not performing as expected, manage these incident reports, and attempt repairs.

For more information, see Oracle Solaris Enterprise Manager.

♦ ♦ ♦ CHAPTER 2

Troubleshooting When a System Hangs or Rebooting Fails

This chapter describes what you can do when a system hangs or if you have problems rebooting a system.

This chapter contains following topics:

What to Do If Rebooting Fails

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

Cause of System Not Booting	How to Fix the Problem
The system cannot 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 11.3 Systems*.
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. If the primary boot environment is not bootable, boot the backup boot environment. Alternatively, you can boot from the Live CD or USB media.
An invalid entry is 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 11.3 Systems*.

Cause of System Not Booting	How to Fix the Problem
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 11.3 Systems*.
A hardware problem exists 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 Forget the root Password or Cannot Boot the System

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

1. Stop the system.
2. Follow the directions in "How to Boot From Media to Resolve an Unknown root Password" in *Booting and Shutting Down Oracle Solaris 11.3 Systems*.
3. If the root password is the problem, remove the root password from the /etc/shadow file.
4. Reboot the system.
5. 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 a software process is stuck. Follow these steps to recover from a hung system.

1. If the system is running a window environment, 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 the window system appears to be hung, identify the process and kill it.

2. Press Control-\ to force quit the running program and 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 more information, see "Forcing a Crash Dump and Reboot of the System" in *Booting and Shutting Down Oracle Solaris 11.3 Systems*.
7. If the system still does not respond, turn the power off, wait a minute or so, and then turn the power back on.
8. If you cannot get the system to respond at all, contact your local service provider for help.

♦ ♦ ♦ CHAPTER 3

Troubleshooting File System Problems

This chapter describes how to fix file system issues including the following:

What to Do If a File System Fills Up

When an UNIX file system (UFS) fills up, you will see the following message in the console window:

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

However, with ZFS datasets, the `file system full` message is not reported when dataset is filled up.

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

A ZFS File System Will Not Report `file system full` on Exceeding Quota

Cause: On ZFS, file system (dataset) size can either be the full size of the underlying `zpool` or the size of the quota set within the dataset property. Unlike traditional UFS, when a ZFS file system (dataset) is filled up, the system does not report `file system full` in `/var/adm/messages`. On ZFS file system that do not have quota property set, the size of the file system is the size of the underlying `zpool`.

Solution: To monitor for dataset usage and `zpool` usage, you can use the `zfs list` and the `zpool list` commands. For more information, see "Resolving ZFS Space Issues" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

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

Cause: Someone accidentally copied a file or directory to the wrong location or an application crashed and wrote a large `core` file to the file system.

Solution: Log in and assume the `root` role. Use the `ls -tl` command in the 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

Cause: A TMPFS file system is trying to write more than is allowed or current processes are using a lot of memory.

Solution: For information about recovering from `tmpfs`-related error messages, see the `tmpfs`(7FS) man page.

What to Do If File Access Control Lists Are Lost After Copy or Restore

Cause: Files or directories with access control lists (ACLs) were copied or restored into the `/tmp` directory and the ACL attributes were lost. The `/tmp` directory is usually mounted as a temporary file system, which doesn't support UFS attributes such as ACLs.

Solution: Copy or restore files into the `/var/tmp` directory instead.

Troubleshooting File Access Problems

Users who cannot access a program, a file, or a directory that they previously could, often call a system administrator for help:

When you encounter such a problem, investigate one of the following possibilities:

- Whether the user's search has been changed or the directories in the search path are not in the proper order.
- Whether the file or directory has the proper permissions or ownership.
- Whether the configuration of a system accessed over the network has changed.

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

Solving Problems With Search Paths

When you try to access a command, you might get the wrong version of the command or the message `Command not found` might display.

In order to fix a search path problem, you need to know the pathname of the directory where the command is stored. Check the man page for the command to see its typical location.

Accessing an Incorrect Version of the Command

If the wrong version of the command is found, a directory including the same command name is in your search path. The correct directory might appear later in the search path or may not be included at all.

▼ How to Diagnose and Correct Search Path Problems

1. **Determine which version of the command you are using.**
   For example:

   ```
   $ which acroread
   /usr/bin/acroread
   ```

2. **Display the current search path.**

   ```
   $ echo $PATH
   ```

3. **Check the current search path to see whether the correct directory is included and whether other directories including the same command name are listed before the correct directory.**

4. **In the path listed in the `.profile` file in your home directory, either add the correct directory or move the correct directory to an earlier position in the path.**

Use a colon to separate path names.

5. **(Optional) If you need to use the command before your next system login, activate the new path.**

   ```
   $ . $HOME/.profile
   ```

6. **(Optional) If you have activated the new path, verify that the command will be accessed through the correct path.**

   $ **which** *command*

Accessing Commands That Are Not Found

The error message `Command Not Found` is displayed due to one of the following reasons:

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

If the command is not available on the system, contact your system administrator.

▼ How to Include a Search Path in Your Path

1. **Display the current search path and make sure that the directory for the command is not in your path or that the path is not misspelled.**

   ```
   echo $PATH
   ```

2. **Add the directory for the command to the `PATH` entry in the `$HOME/.profile` file.**
   Use a colon to separate path names.

3. **Activate the new path.**

   ```
   $ . $HOME/.profile
   ```

4. **Verify that the correct path is now displayed for the command.**

   $ **which** *command*

Changing File and Group Ownerships

Frequently, file and directory ownerships change because someone edited the files as an administrator. 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 of, 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 1, “Controlling Access to Files” in *Securing Files and Verifying File Integrity in Oracle Solaris 11.3*.

Solving File Access Problems

When users cannot access files or directories that they could previously access, the permissions or ownership of the files or directories have probably 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 11.3* for information about problems with network access and problems with accessing systems through AutoFS.

♦ ♦ ♦ CHAPTER 4

Prepare for Possible Process Failures by Using Core Files

This chapter describes how to set up the specifications for the core files that a system produces when a process fails and how to examine these core files after a failure. This chapter includes the following topics:

About Process Failures and Core Files

When a process or an application terminates abnormally, the system automatically generates a set of files. This process is described as a *core dump*. The files that are created are *core files*. A core file is a disk copy of the contents of the process address space at the time of its termination, along with additional information about the state of the process. Typically, core files are produced following abnormal termination of a process resulting from a bug in the corresponding application. A core file provides invaluable information for you to use in diagnosing the problem that cause the process failure. For more information, see "Parameters for Core File Creation" on page 36.

As part of your ongoing system administration, you can use the `coreadm` command to control the creation specifications for core files. 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 so you can track problems easily. For more information, see "Administering Your Core File Specifications" on page 38.

When a process terminates abnormally, you can inspect the core files that are created using a debugger such as `mdb` or by using a `proc` tool. For more information, see "Examining Core Files After a Process Failure" on page 41.

Generating Core Files

This section describes different aspects in the generation of core files during crash dumps.

Parameters for Core File Creation

When a process fails, the system tries to create up to two core files for each failed process, using a global core file name pattern and a per-process core file name pattern to create each core file name. The `coreadm` command controls these name patterns and specifies the location of the core files. This section describes some of the file path and file name parameters. For a full description of the core dump process, see the `core(4)` man page. For the full description of the `coreadm` options, see the `coreadm(1M)` man page.

Configurable Core File Paths

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 another in the global core file location. The file paths that are used are configurable parameters.

Two configurable core file paths can be enabled or disabled independently of each other as follows:

- 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.

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

Expanded Core File Names

The name of a core file contains fields with information about the failed process. For a full description of the core file name fields, see the coreadm(1M) man page. This section focuses on the global variables.

If a global core file directory is enabled, core files can be distinguished from one another by using the following variables:

%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, suppose /var/core/core.%f.%p is set as the global core file path. If a sendmail process with PID 12345 terminates abnormally, it would produce /var/core/core.sendmail.12345 as the core file.

Improving Core File Dump Performance

You can improve the performance of core file dumping on a system by excluding some parts of a process's binary image from the core dump. When you use the coreadm command to customize your core dump specifications, you could specify exclusion of, for example, DISM mappings, or ISM mappings, or System V shared memory from a core dump. For more information, see the coreadm(1M) man page.

Administering Your Core File Specifications

This section provides the following information about managing core files:

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
```

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 for all processes that are started by the `init` process. This setting applies to all processes that have not explicitly overridden the default core file pattern. This setting persists 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. For example:

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

Alternately, you could assume the root role and set a global file name pattern:

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

After a global or per-process core file name pattern is set, it must be enabled with the coreadm -e command.

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

Enabling File Paths

You can enable a per-process or global core file path.

- To enable a per-process core file path, assume the root role and issue the following command:

  ```
  # coreadm -e process
  ```

  If you want to verify the configuration, display the current process core file path.

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

- To enable a global core file path, assume the root role and issue the following command:

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

  If you want to verify the configuration, display the current process core file path.

  ```
  # 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
```

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 appropriate path.

- 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 files 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.

Reverting to the Default Core File Settings

As `root`, run one of the following commands to disable the core file path and remove the core file name pattern.

- For global core file settings:

  ```
  # coreadm -d global -g ""
  ```

  Note - `""` is an empty string with no space.

- For per-process core file settings:

  ```
  # coreadm -d process -g ""
  ```

  The `-d` option disables the core file path. The `-g` option with the empty string variable removes the core file name pattern. The core file path and core file name pattern are returned to the original default settings.

Correcting Obsolete Core File Parameters

The following message appears if you have an obsolete parameter that allows `setuid` core files in your `/etc/system` file:

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

To correct this problem, 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 After a Process Failure

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.

You can apply `/usr/proc/bin/pstack`, `pmap`, `pldd`, `pflags`, and `pcred` tools 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 the `proc` tools to examine core files, see the `proc(1)`.

EXAMPLE 5 Examining Core Files With the `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 5

Managing System Logs and Messaging

This chapter describes how to view and manage system logs and system messages. It covers the following information:

About System Logs and Messaging

The `syslogd` command enables system message display and forwarding capabilities. It reads and forwards system messages to the appropriate log files or users, based on the priority of a message and the location of its system facility. While the `syslog` configuration file controls where messages are forwarded to, the `syslogd` command logs a timestamp for every message with interval minutes. Note that the default interval time is set to 20 minutes.

If you want to grant non-root users the privilege of maintaining log files, you can grant `Log Management` as a rights profile for that user. You can make that grant by using the `-P` option with either the `useradd` command for new users or the `usermod` command for an existing user. For instructions, see the `useradd`(1M) and the `usermod`(1M) man pages.

Managing System Messages and Logging

This section describes the system messaging features in Oracle Solaris.

System Message Formats

System messages display on the console device. The text of most system messages has the following format:

`[ID` *msgid facility*`.]`

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 panic message on the system console, formatted as follows:

`panic:` *error message*

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

```
Watchdog reset !
```

System Message Logging

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 46. 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 "Removing Dump Files" in *Managing Devices in Oracle Solaris 11.3* and Chapter 4, "Scheduling System Tasks" in *Managing System Information, Processes, and Performance in Oracle Solaris 11.3*.

Viewing System Messages and Logging

To display recent messages generated by a system crash or reboot, use the `dmesg` command.

```
$ dmesg
```

For large message logs, use the `more` command to display only one screen of messages at a time.

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

EXAMPLE 6 Viewing System Messages

The following example shows output from the `dmesg` command on an Oracle Solaris 11 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]...
.
.
.
```

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, suppose 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` 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. Note that you must first assume `root` or a role that has `syslog.conf` authorization.

For example, to rotate the Apache access and error logs, you would 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 ten 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 Apache log rotation examples look similar to the following example:

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

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

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, which also contain crash and boot messages. To view `/var/adm` messages, see “Viewing System Messages and Logging” on page 45.

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

facility.level ... action

facility.level	A *facility* is a system source of the message or condition. May be a comma-separated list of facilities. A *level* indicates the severity or priority of the condition being logged. 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 at least that priority 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	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                                      *
```

These entries cause the following user messages to be logged automatically:

- 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:

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.

The most common priority levels for `syslog.conf` messages, in priority order, are:

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

2. **Use the `pfedit` command and edit the `/etc/syslog.conf` file.**

   You can add or change message sources, priorities, and message locations. For more information, see the `syslog.conf`(4).

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

3. **Save the changes.**

Example 7 Customizing System Message Logging

This example shows a entry for a `/etc/syslog.conf` `user.emerg` facility that sends user emergency messages to `root` and individual users.

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

Extended System Logging With the rsyslogd Command

This Oracle Solaris release includes the option of installing and using the rsyslogd package for managing system logging. rsyslogd is derived from the syslogd daemon implementation, with a modular design that supports several features such as filtering, TCP, encryption, and high-precision time-stamps, as well as output control.

The syslog SMF service, svc:/system/system-log:default, continues to be the default logging service. To use the rsyslog service, you need to install the rsyslog package and enable the rsyslogd service.

▼ How to Install and Enable rsyslog

1. **Check whether the rsyslog package is already installed on your system by trying to enable the service as follows:**

```
root@pcclone: ~# svcadm enable svc:/system/system-log:rsyslog
```

If the rsyslog package is not installed, the following message displays:

```
svcadm: Pattern 'svc:/system/system-log:rsyslog' doesn't match any instance.
```

2. **If the rsyslog package is not installed, install it.**

```
root@pcclone:~# pkg install rsyslog
           Packages to install:  3
            Services to change:  1
       Create boot environment: No
Create backup boot environment: No

DOWNLOAD                                  PKGS         FILES    XFER (MB)   SPEED
Completed                                  3/3         68/68      1.7/1.7  354k/s

PHASE                                          ITEMS
Installing new actions                     147/147
Updating package state database               Done
Updating package cache                         0/0
Updating image state                          Done
Creating fast lookup database                 Done
```

3. **Confirm the rsyslog instance.**

```
root@pcclone:~# svcs -a | grep "system-log"
```

```
disabled       18:27:16 svc:/system/system-log:rsyslog
online         18:27:21 svc:/system/system-log:default
```

This output confirms that the `rsyslog` instance exists, though it is disabled.

4. **Switch to the `rsyslog` service.**

```
root@pcclone:~# svcadm disable svc:/system/system-log:default
root@pcclone:~# svcadm enable svc:/system/system-log:rsyslog
root@pcclone:~# svcs -xv
```

These commands disable the default service, enable `rsyslog`, and report on status.

Next Steps After `rsyslog` is installed and enabled, you can configure `syslog` in the `/etc/rsyslog.conf` file. For more information, see `rsyslogd(1m)`.

Enabling Remote Console Messaging

The following console features improve your ability to troubleshoot Oracle Solaris 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.

  Although 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, see "Guidelines for Using the `consadm` Command During an Interactive Login Session" on page 52.
- 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`.
- 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 redirected only until the carrier drops or the auxiliary console device is deselected. Therefore, the carrier must be established on the port before you can successfully use the `consadm` command.

Using Auxiliary Console Messaging During Run Level Transitions

Note the following information 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 root password when transitioning between run levels, has been modified to send the root password prompt to each auxiliary device in addition to the default console device.
- The user should never directly invoke `sulogin`. The user must have the `solaris.system.maintenance authorization` to use this utility.
- 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 root 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 root 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 that device also has no carrier, you will have to re-establish the carrier and enter the correct run level. The `init` or `shutdown` command will not display the run-level prompt again.

- 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 regardless of whether this device is the auxiliary console. 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 deselected. However, the `consadm` command includes an option to set a device as the auxiliary console across system reboots. For more information, see the following procedure.

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

You can 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 as if it is coming from a console device. This process allows you to enter your password to the `sulogin` program as described in "Using Auxiliary Console Messaging During Run Level Transitions" on page 51.

  If you enter the correct password on the auxiliary terminal, the auxiliary console runs an interactive `sulogin` session and locks out the default console and any competing auxiliary console. This behavior means that 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 Display a List of Auxiliary Consoles

1. **Issue the `consadm` command as `root`, with no arguments.**
   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

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

   a. **Displays a list of auxiliary consoles.**

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

   b. **To display a list of persistent auxiliary consoles, use the `-p` option.**

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

How to Enable an Auxiliary (Remote) Console

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

2. **Enable the auxiliary console.**

   ```
   # consadm -a device name
   ```

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

   ```
   # consadm
   ```

Example 8 Enabling an Auxiliary (Remote) Console

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

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

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

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

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

   ```
   # consadm -a -p device name
   ```

   This command 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 9 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 "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.3*.

2. **Disable the auxiliary console.**

   ```
   # consadm -d devicename
   ```

   To also remove the auxiliary console from the list of persistent auxiliary consoles, add the **-p** option.

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

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

   ```
   # consadm
   ```

Example 10 Disabling an Auxiliary (Remote) Console

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

Index

U

V

W