11

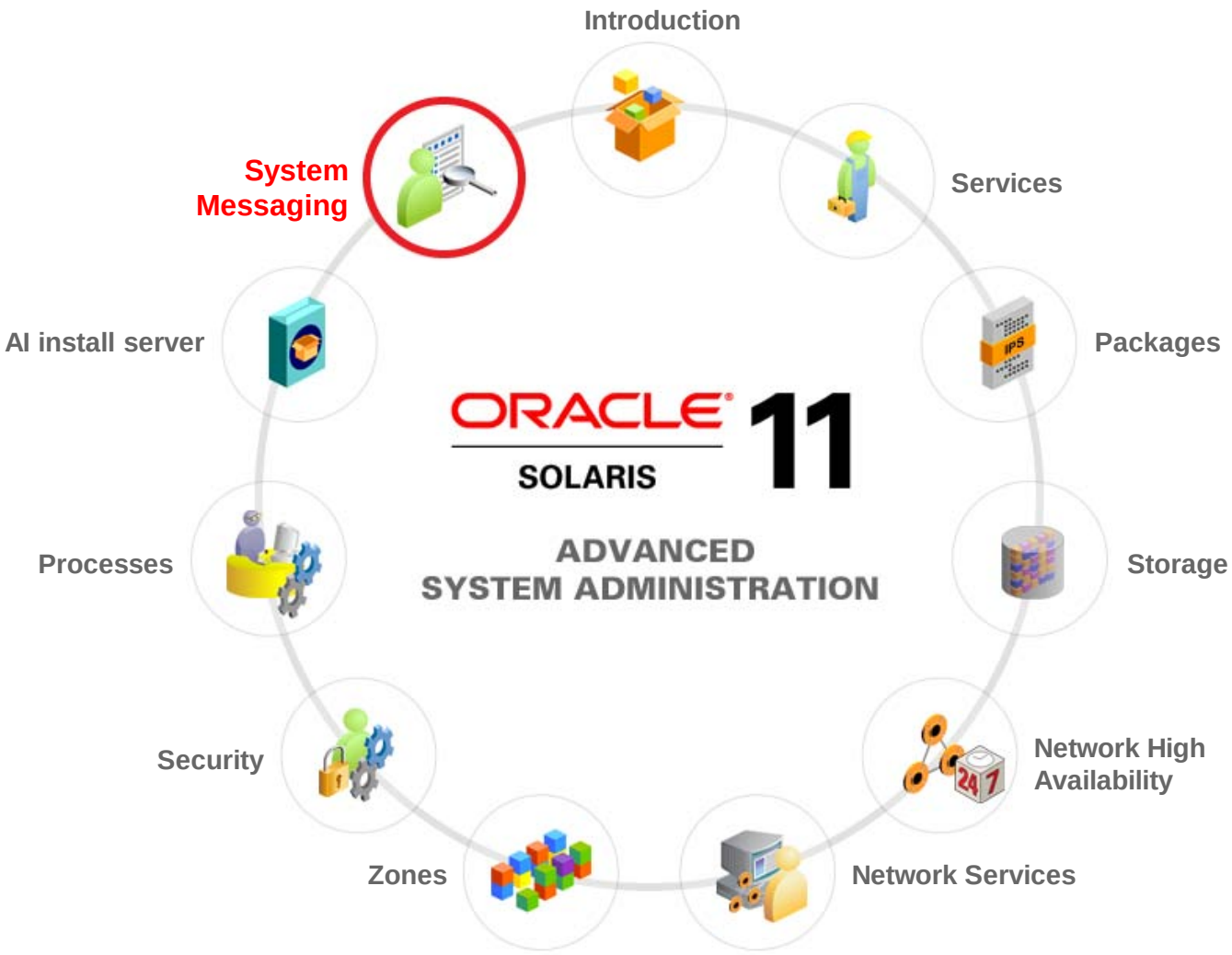
Implementing System Messaging and Diagnostic Facilities

Objectives

After completing this lesson, you should be able to:

- Monitor system logs
- Identify a crash dump file
- Identify a core dump file
- Troubleshoot a script execution issue
- Troubleshoot a software update failure
- Troubleshoot a network connectivity issue
- Troubleshoot a directory access issue
- Troubleshoot a default shell issue
- Configure the following:
  - System messaging
  - System crash facilities
  - Dump facilities for business application failure
- Explain the role of DTrace in diagnosing system issues

Job Workflow



Agenda

- Managing System Logs
- Configuring System Messaging
- Configuring System Crash Facilities
- Configuring Dump Facilities for Business Application Failure
- Explaining the Role of DTrace in Diagnosing System Issues

Importance of Implementing System Messaging and Diagnostic Facilities

System messaging and diagnostic facilities implementation ensures that:

- Controls are in place to monitor system activities so that issues can be addressed quickly and efficiently
- System crashes and core dumps are captured and reported so that major problems can be analyzed and corrected

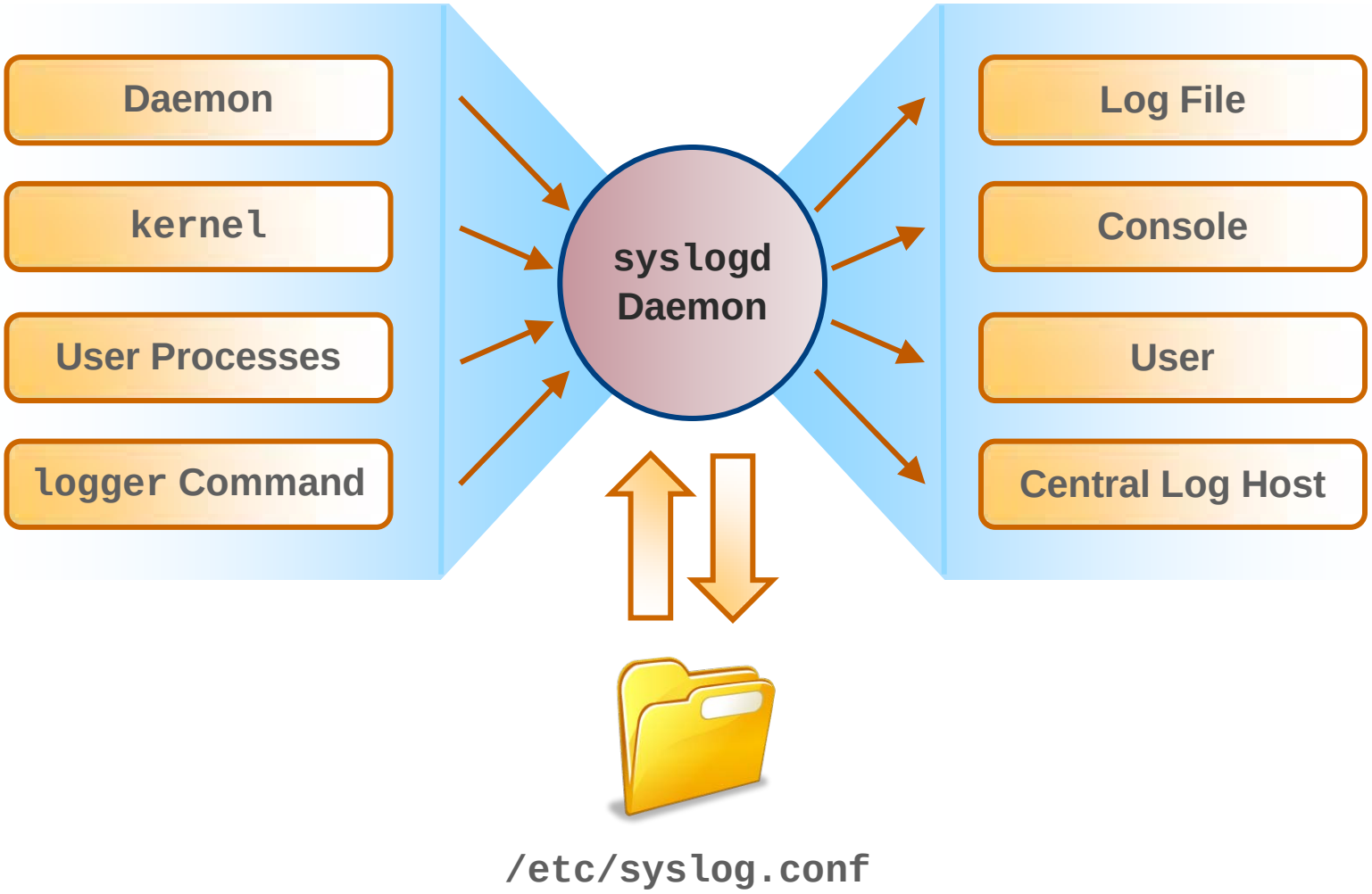
Monitoring System Logs

System messages are stored in the `/var/adm` directory.

```
client1:/var/adm# ls
acct        lastlog    messages    messages.2  sa          sulog
aculog      log        messages.0  messages.3  sm.bin      utmpx
exacct      loginlog   messages.1  pool        streams     wtmpx
```

- The `/var/adm` directory contains several message files:
  - Most recent messages: `/var/adm/messages`
  - Oldest messages: `messages.3`
- Message files are rotated about every 10 days:
  - `messages.0` is renamed `messages.1`.
  - `messages.1` is renamed `messages.2`.
  - `messages.2` is renamed `messages.3`.

syslogd Daemon



/etc/syslog.conf File

An /etc/syslog.conf file configuration entry consists of:

- The selector field, which contains two components:
  - ***facility:*** Category of system process that can generate messages
  - ***facility.level:*** Severity or importance of a message
- The action field, which determines where the message is sent

```
*.err                    /var/adm/messages
```

Interpreting the /etc/syslog.conf File Selector *facility* Field

Field	Description
kern	Messages generated by the kernel
user	Messages generated by user processes. This is the default priority for messages from programs or facilities that are not listed in this file.
mail	Messages generated by the mail system
daemon	Messages generated by system daemons, such as the in.ftpd and the telnetd daemons
auth	Messages generated by the authorization system, including the login, su, and getty commands
lpr	Messages generated by the line printer spooling system, such as the lpr and lpc commands

Interpreting the `/etc/syslog.conf` File Selector *`facility`* Field

Field	Description
`news`	Files reserved for the USENET network news system
`uucp`	Designated for the UNIX-to-UNIX copy (UUCP) system, which does not currently use the `syslog` function
`cron`	Designated for `cron/at` messages generated by systems that do logging through `syslog`. The current version of the Oracle Solaris Operating Environment does not use this facility for logging.
`audit`	Designated for audit messages generated by systems that audit by means of `syslog`
`local0-7`	Fields reserved for local use
`mark`	The time when the message was last saved
`*`	All facilities, except the `mark` facility

Interpreting the `/etc/syslog.conf` File Selector *`level`* Field

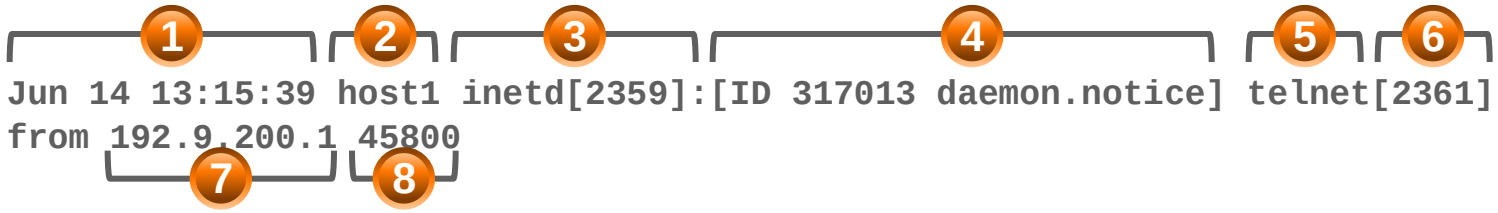
Level	Priority	Description
`emerg`	0	Panic conditions that are normally broadcast to all users
`alert`	1	Conditions that should be corrected immediately, such as a corrupted system database
`crit`	2	Warnings about critical conditions, such as hard device errors
`err`	3	Errors other than hard device errors
`warning`	4	Warning messages
`notice`	5	Nonerror conditions that might require special handling
`info`	6	Informational messages
`debug`	7	Messages that are normally used only when debugging a program
`none`	8	Messages not sent from the indicated facility to the selected file

Interpreting the /etc/syslog.conf File Action Field

Field	Description
/pathname	This indicates the full path name to the targeted file.
@host	The @ sign denotes that messages must be forwarded to a remote host. Messages are forwarded to the syslogd daemon on the remote host.
user1, user2	The user1 and user2 entries receive messages if they are logged in.
*	All logged-in users receive messages.

Monitoring a `syslog` File in Real Time

To view messages sent to the `/var/adm/messages` file, use `tail -f /var/adm/messages`.



Number	Field	Result
1	Date/time	`Jun 14 13:15:39`
2	Local host name	`host1`
3	Process name/PID number	`inetd[2359]`
4	MsgID number/selector facility.level	`[ID 317013 daemon.notice]`
5	Incoming request	`telnet`
6	PPID number	`[2361]`
7	IP address	`192.9.200.1`
8	Port number	`45800`

Interpreting System Messages

```
# tail -f /var/adm/messages
Nov 9 09:25:53 server1 named[472]: [ID 873579 daemon.notice] running
Nov 9 09:26:24 server1 mac: [ID 736570 kern.info] NOTICE: net1
unregistered
#
```

```
# tail -f /var/adm/messages
Nov 9 09:40:03 client1 genunix: [ID 936769 kern.info] fssnap0 is
/pseudo/fssnap@0
Nov 9 09:40:07 client1 gnome-session[784]: [ID 702911 daemon.warning]
WARNING: IceListenForConnections returned 2 non-local listeners:
inet/client1:39166,inet6/client1:38708
#
```

Configuring the /etc/syslog.conf File

You configure this file to:

- Define target locations for the syslog message files
- Use a selector level of err to indicate that all events of priority error (and higher) are logged to the target defined in the action field

```
*.err;kern.notice;auth.notice                  /dev/sysmsg
*.err;kern.debug;daemon.notice;mail.crit       /var/adm/messages
*.alert;kern.err;daemon.err                    operator
*.alert                                        root
usr.emerg                                      *
```

Note: Whenever you make changes to this file, you must restart the syslogd daemon.

Stopping and Starting the syslogd Daemon

- The syslogd daemon can be started:
  - Automatically during boot
  - Manually from the command line
- Each time the syslogd daemon starts, the /etc/syslog.conf configuration file is read.
- After you have modified the configuration file, you can:
  - Manually stop or start the syslogd daemon
  - Send the syslogd daemon a refresh command

```
# svcadm disable svc:/system/system-log:default
# svcadm enable svc:/system/system-log:default
# svcadm refresh svc:/system/system-log:default
```

logger Command

- This command enables you to send messages to the syslogd daemon.
- You can write administrative shell scripts that report the status of backups or other functions.

```
logger [ -i ] [ -f file ] [ -p priority ] [ -t tag ] [ message ]
```

```
# logger System rebooted
```

```
# logger -p user.err System rebooted
```

Agenda

- Managing System Logs
- Configuring System Messaging
- Configuring System Crash Facilities
- Configuring Dump Facilities for Business Application Failure
- Explaining the Role of DTrace in Diagnosing System Issues

Configuring System Messaging

This section covers the following topics:

- TCP tracing
- Logging a message by using TCP trace
- Setting up message routing
- Restarting the message logging daemon
- Using TCP trace to log a message
- Monitoring message logging in real time

TCP Tracing

- You can use the `inetadm` command to enable the trace option on one or more services.
- The `inetadm` command uses the `syslog` command to record and log the following:
  - Client's IP address
  - TCP port number
  - Name of the service
- You can configure `/etc/syslog.conf` so that the `syslogd` daemon selectively distributes messages sent to it from the `inetd` daemon.

```
# grep daemon.notice /etc/syslog.conf
*.err;kern.debug;daemon.notice;mail.crit /var/adm/messages
```

Logging a Message by Using TCP Trace

```
# inetadm -m spray tcp_trace=TRUE
# inetadm -l svc:/network/rpc/spray:default
SCOPE    NAME=VALUE
         name="sprayd"
         endpoint_type="tli"
         proto="datagram_v"
         isrpc=TRUE
         rpc_low_version=1
         rpc_high_version=1
         wait=TRUE
         exec="/usr/lib/netsvc/spray/rpc.sprayd"
         user="root"
default  bind_addr=""
default  bind_fail_max=-1
default  bind_fail_interval=-1
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
default  failrate_interval=60
default  inherit_env=TRUE
default  tcp_trace=TRUE
default  tcp_wrappers=FALSE
default  connection_backlog=10
default  tcp_keepalive=FALSE
```

Setting Up Message Routing

1. By using the `pfedit` utility, edit the contents of the `/etc/syslog.conf` file to append the following entry to the file:

   ```
   local0.notice                    @hostname
   ```

2. Restart the `syslogd` daemon to activate the new configuration.
3. On the local host, create the `/var/log/loca10.log` file.
4. Modify the `/etc/syslog.conf` file and add the entry as follows:

   ```
   local0.notice                    /var/log/local0.log
   ```

5. Restart the `syslogd` daemon to activate the new configuration.

Setting Up Message Routing: Example

```
root@host1:~# vi /etc/syslog.conf
<content omitted>
local0.notice                @host2
root@host1:~# svcadm refresh system/system-log
root@host2:~# touch /var/log/local0.log
root@host2:~# vi /etc/syslog.conf
root@host2:~# grep local0 /etc/syslog.conf
local0.notice                     /var/log/local0.log
root@host2:~# svcadm refresh system-log
```

Monitoring a `syslog` File in Real Time

To view the messages sent to the `/var/adm/messages` file, use `tail -f /var/adm/messages`.

```
# tail -f /var/adm/messages
…
…
Dec 20 06:10:05 client1 inetd[655]: [ID 317013 daemon.notice]
    ftp[3044] from 192.168.0.100 61017
```

Practice 11: Overview

You will perform the following practices:

- Practice 11-1: Troubleshooting a script execution issue
- Practice 11-2: Troubleshooting a software update failure
- Practice 11-3: Troubleshooting a network connectivity issue
- Practice 11-4: Troubleshooting directory access issues
- Practice 11-5: Using the man pages
- Practice 11-6: Setting up system messaging

Agenda

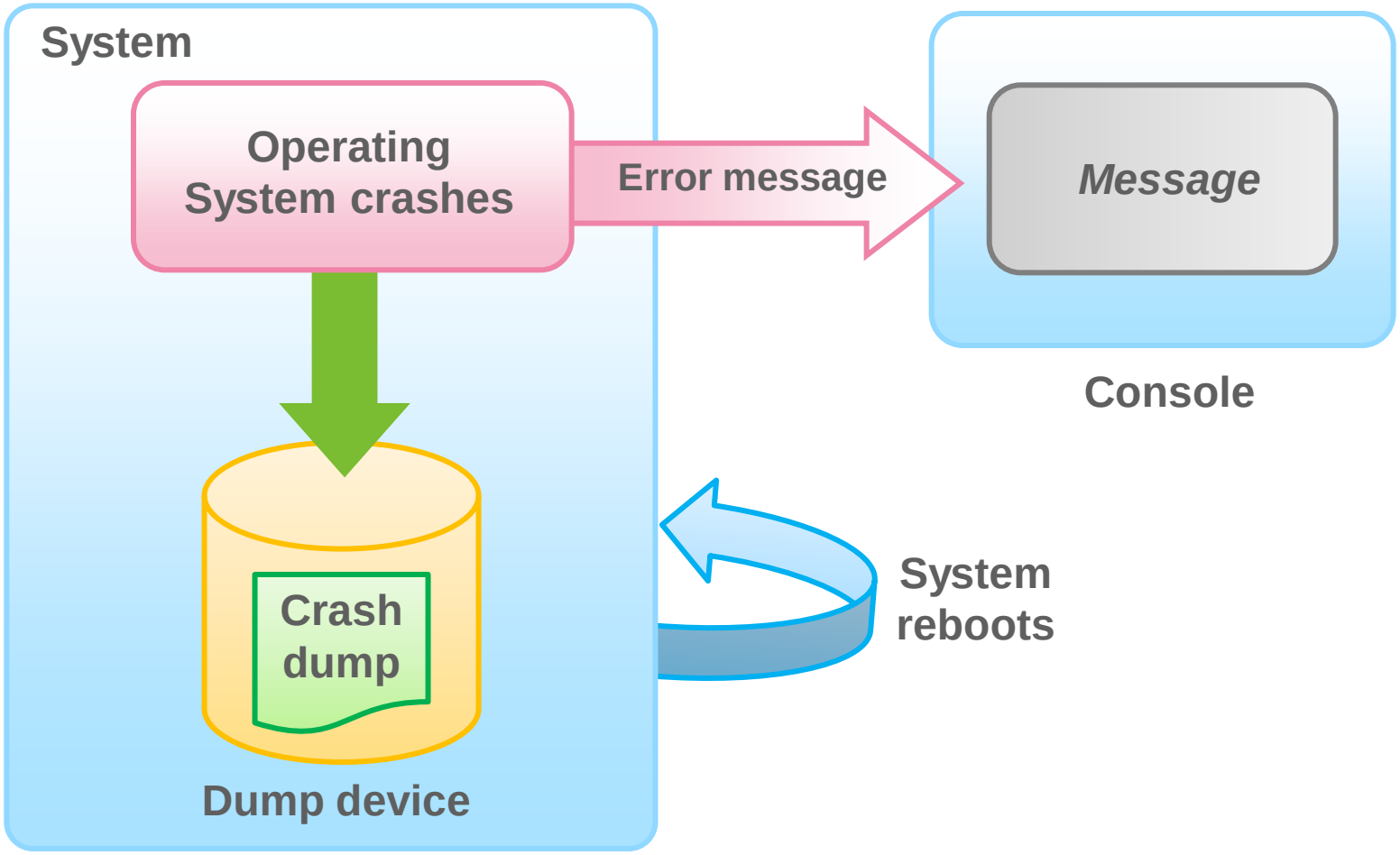
- Managing System Logs
- Configuring System Messaging
- Configuring System Crash Facilities
- Configuring Dump Facilities for Business Application Failure
- Explaining the Role of DTrace in Diagnosing System Issues

Configuring System Crash Facilities

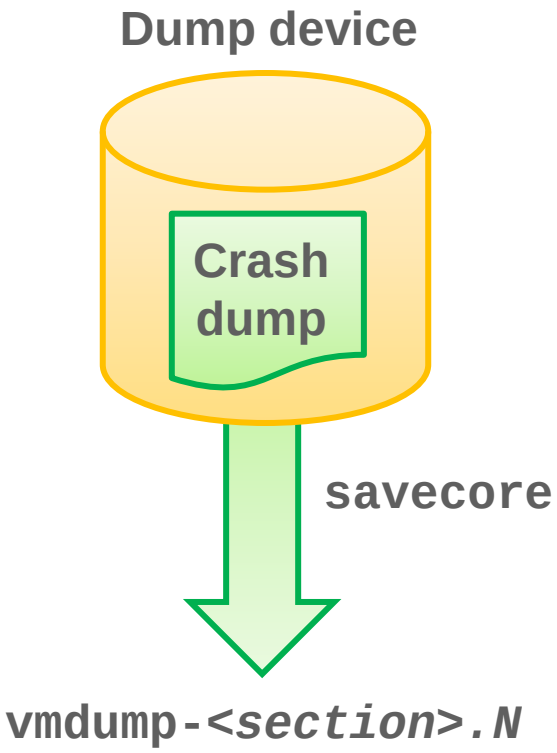
This section covers the following topics:

- Crash dump process: Overview
- How and where crash dump files are saved
- Displaying the current crash dump configuration
- Modifying the crash dump configuration
- Saving the crash dump file
- Uncompressing the crash dump file
- Displaying the crash dump file contents

Crash Dump Process: Overview



How and Where Crash Dump Files Are Saved



/etc/dumpadm.conf File

```
# cat /etc/dumpadm.conf
#
# dumpadm.conf
#
# Configuration parameters for system crash dump.
# Do NOT edit this file by hand -- use dumpadm(1m) instead.
#
DUMPADM_DEVICE=/dev/zvol/dsk/rpool/dump
DUMPADM_SAVDIR=/var/crash/client1
DUMPADM_CONTENT=kernel+zfs
DUMPADM_ENABLE=yes
DUMPADM_CSAVE=on
```

Displaying the Current Crash Dump Configuration

To display the current crash dump configuration, use `dumpadm`.

```
# dumpadm
Dump content      : kernel with ZFS metadata
Dump device       : /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash/client1
Savecore enabled  : yes
Save compressed   : on
```

Modifying the Crash Dump Configuration

To modify the crash dump configuration, use `/usr/sbin/dumpadm [-nuy] [-c content] [-d dump-device][-m mink | minm | min%] [-s savecore-dir] [-r root-dir] [-z on | off].`

```
# dumpadm -y -d /dev/dsk/c0t1d0s1
Dump content      : kernel
Dump device       : /dev/dsk/c0t1d0s1 (dedicated)
Savecore directory: /var/crash/client1
Savecore enabled  : yes
Save compressed   : on
```

Saving the Crash Dump File

To save the crash dump file to the designated dump device, use `savecore -L`.

```
# savecore -L
dumping to /dev/zvol/dsk/rpool/dump, offset 65536, content: kernel
    sections: zfs
 0:02  91% done (kernel)
 0:02 100% done (zfs)
100% done: 142706 (kernel) + 13595 (zfs) pages dumped, dump succeeded
savecore: System dump time: Tue Jan  7 07:59:43 2014

savecore: Saving compressed system crash dump files in directory
    /var/crash/data/3e381450-d324-c553-a88a-a89bd42b6f7e
savecore: Decompress all crash dump files with '(cd
    /var/crash/data/3e381450-d324-c553-a88a-a89bd42b6f7e && savecore -v
    0)' or individual files with 'savecore -vf /var/crash/data/3e381450-
    d324-c553-a88a-a89bd42b6f7e/vmdump{,-<secname>}.0'
```

Uncompressing the Crash Dump File

To uncompress the crash dump file, use `savecore -vf /var/crash/hostname/vmdump.0`.

```
# savecore -vf /var/crash/data/3e381450-d324-c553-a88a-
a89bd42b6f7e/vmdump.0
savecore: System dump time: Tue Jan  7 07:59:43 2014

savecore: saving system crash dump in /var/crash/data/3e381450-d324-
c553-a88a-a89bd42b6f7e/vmcore.0
Constructing corefile /var/crash/data/3e381450-d324-c553-a88a-
a89bd42b6f7e/vmcore.0
 0:18 100% done: 142706 of 142706 pages saved
4874 (3%) zero pages were not written
dump decompression took 0 minutes and 18 seconds
```

Displaying the Crash Dump File Contents

To display the contents of the crash dump file, perform the following steps:

1. Change directories to the `/var/crash` directory.
2. List the files in the crash directory.
3. Use the `file` command to access the crash dump file, either `vmcore.0` or `vmdump.0`.
4. View the contents of the file by using the `string` command.

Displaying the Crash Dump File Contents: Example

```
# cd /var/crash/data/<UID>
root@client1:/var/crash/data/<UID># savecore –v 0
root@client1:/var/crash/data/<UID># ls
vmcore-zfs.0  vmcore.0      vmdump-zfs.0  vmdump.0
root@client1:/var/crash/data/<UID># file vmcore.0
vmcore.0:  SunOS 5.11 11.3 64-bit Intel live dump from ‘client1‘
root@client1:/var/crash/data/<UID># strings vmcore.0 | more
i86pc
3e381450-d324-c553-a88a-a89bd42b6f7e
SunOS
client1
5.11
11.3
i86pc
kernel
core kernel pages
ZFS metadata (ZIO buffers)
.symtab
.strtab
.shstrtab
...
...
```

Quiz

Saving of the crash dump file is enabled by default.

a. True
b. False

Agenda

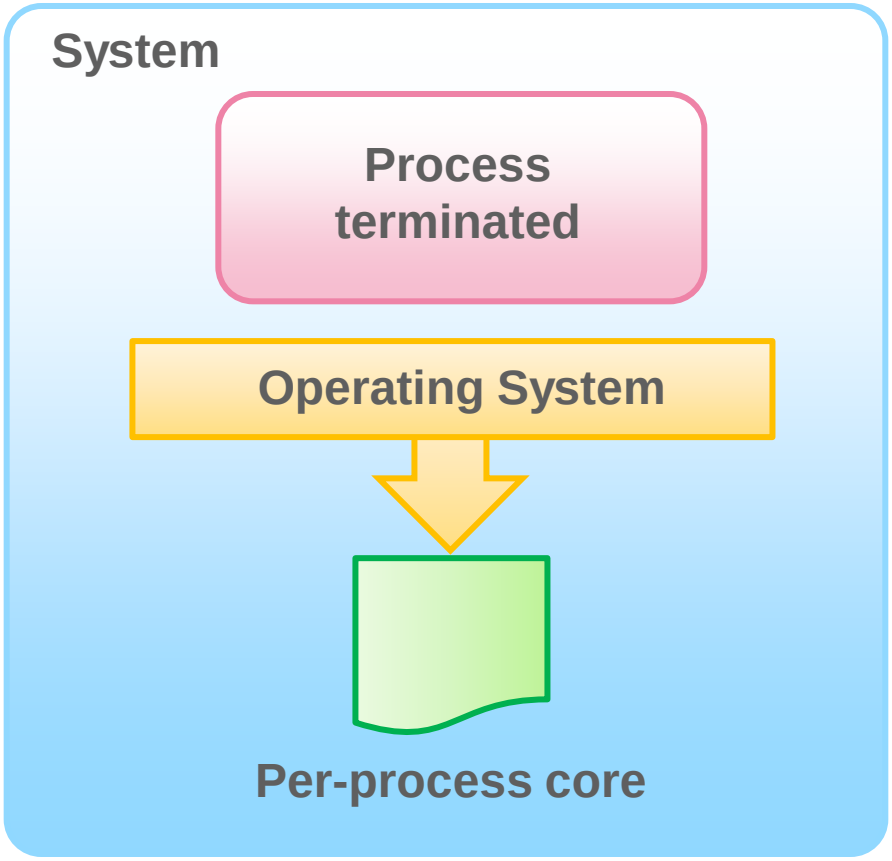
- Managing System Logs
- Configuring System Messaging
- Configuring System Crash Facilities
- Configuring Dump Facilities for Business Application Failure
- Explaining the Role of DTrace in Diagnosing System Issues

Configuring Dump Facilities for Business Application Failure

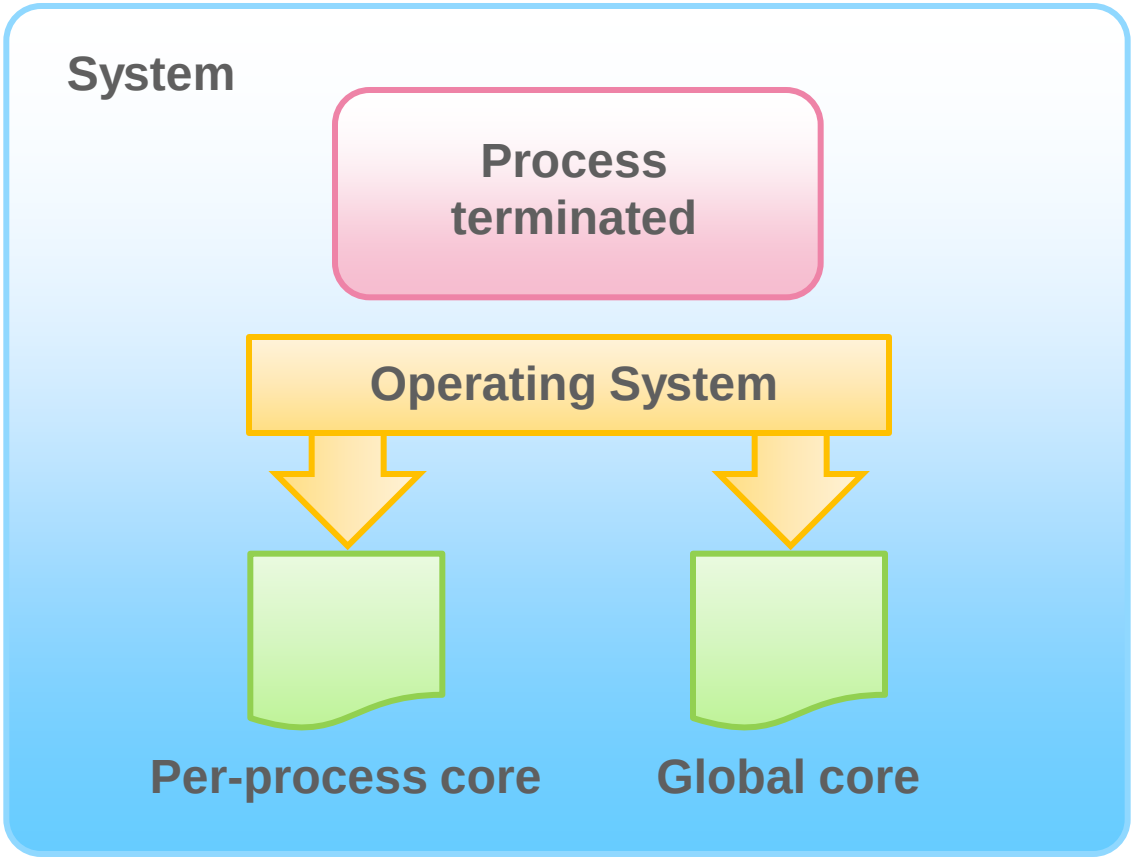
This section covers the following topics:

- Displaying the current core dump configuration
- Modifying the core dump configuration
- Setting a core file name pattern
- Enabling a core file path
- Displaying the contents of the core dump file

Core Dump Process: Overview



How and Where Core Dump Files Are Saved



Core Dump: Example

```
core:  ELF 32-bit LSB core file 80386 Version 1, from 'bash'
```

What Is a Core File?

- A file generated in response to fatal errors.
  - For the kernel, the file is referred to as a “crash dump.”
  - For an application or process, the file is referred to as a “core dump.”
- A fatal kernel error:
  - Results in a system crash (“panic”)
  - Generates a core file that contains a snapshot of the kernel’s memory space when the error occurred
- A fatal application or process error:
  - Results in abnormal termination
  - Generates a core file that contains a snapshot of the process’s memory space when the error occurred

Core File Generation: Advantages and Disadvantages

- Advantages
  - Core files are used to analyze and determine the root cause of a crash or core dump so that the problem can be resolved.
- Disadvantages
  - Core files take up a large amount of disk space.
  - Core files must be managed.

Core File Paths

- Per-process core file path:
  - Defaults to `core`
  - Is enabled by default
  - If enabled, produces a core file when a process terminates abnormally
  - Is inherited by a new process from its parent process
- Per-process files are owned and can be viewed only by the process owner.
- Global core file path:
  - Defaults to `core`
  - Is disabled by default
  - If enabled, produces an additional core file with the same content as the per-process core file
- Global core files are owned by the superuser. Nonprivileged users cannot read these files.

Displaying the Current Core Dump Configuration

To display the current core dump configuration, use `coreadm`.

```
# 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: disabled
       per-process core dumps: enabled
      global setid core dumps: disabled
 per-process setid core dumps: disabled
     global core dump logging: disabled
```

Modifying the Core Dump Configuration

To modify the core dump configuration, use `coreadm [-g pattern] [-i pattern] [-d option ... ] [-e option ... ]`.

```
# coreadm -e log
# 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: enabled
```

Setting a Core File Name Pattern

- To set a per-process file name pattern, use `coreadm -p $HOME/corefiles/%f.%p $$`.

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

- To set a global file name pattern, use `coreadm -g /var/core/%f.%p`.

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

Enabling a Core File Path

- To enable the per-process core file path, use `coreadm -e process`.
- To enable the global core file path, use `coreadm -e global -g /var/core/core.%f.%p`.
- To verify the configuration, use `coreadm`.

```
# 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: enabled
```

Displaying the Contents of the Core Dump File

To display the contents of the core dump file:

1. Change directories to the `/var/core` directory.
2. List the files in the core directory.
3. Use the `file` command to access the core file.
4. View the contents of the file by using the `string` command.

Displaying the Core Dump File Contents: Example

```
# cd /var/core
root@client1:/var/core# ls /var/core
core.bash.1554
root@client1:/var/core# file core*
core.bash.1554:ELF 32-bit LSB core file 80386 Version 1, from 'bash'
root@client1:/var/core# strings core.bash.1554 | more
CORE
bash
-bash
CORE
i86pc
CORE
CORE
CORE
CORE
bash
-bash
…
…
```

Quiz

In which directory are the system messages stored?

a. /var/lib

b. /var/tmp

c. /var/adm

d. /var/log

Quiz

What is the facility in the following `syslog` entry?

```
Aug 10 05:40:03 client1 genunix: [ID 936769
kern.info] fssnap0 is /pseudo/fssnap@0
```

a. genunix
b. kern
c. info

Quiz

What is the severity level for the following message?

```
Aug 10 05:40:00 client1 mac: [ID 469746
kern.info] NOTICE: net1 registered
```

a. alert
b. warning
c. notice
d. info

Quiz

A core dump is generated when the system experiences a fatal error.

a. True
b. False

Quiz

What is the result of a fatal application or process error? (Choose two.)

a. System panic
b. Abnormal termination of the application or process
c. A core dump being created on the designated dump device

Quiz

You can separately enable or disable two configurable `core` file paths: per-process and global.

a. True
b. False

Quiz

The global core file path is enabled by default.

a. True
b. False

Quiz

What is the default target destination for the following `syslog` message type?

```
*.err;kern.debug;daemon.notice;mail.crit
```

a. /dev/sysmsg

b. /var/adm/messages

c. operator

d. root

Quiz

You must always restart the `syslogd` daemon after you modify the `etc/syslog.conf` file.

a. True

b. False

Practice 11-7 Overview: Configuring System and Application Crash Facilities

This practice covers configuring:

- System crash facilities
- Dump facilities for business application failure

Agenda

- Managing System Logs
- Configuring System Messaging
- Configuring System Crash Facilities
- Configuring Dump Facilities for Business Application Failure
- Explaining the Role of DTrace in Diagnosing System Issues

DTrace: Overview

- A comprehensive, dynamic tracing facility that is built into the Oracle Solaris operating system
- An observability technology that helps you examine the behavior of user programs as well as the operating system in development and in production
- Features:
  - Examines the entire software stack
  - Determines the root cause of performance problems
  - Tracks the source of aberrant behavior

DTrace: Capabilities

- Analysis and observability
  - Provides a powerful new system and a process-centric framework for real-time analysis and observability
- Safety and comprehensive monitoring
  - Provides over 50,000 prebuilt data-monitoring points, inspection kernels, and user space levels
- Flexibility
  - Enables you to create custom programs to dynamically instrument the system
    - There is no need to instrument, stop, or restart your applications.
    - DTrace enables users to dynamically create as many monitoring points as required.

DTrace: Components

- Probes
- Providers
- Consumers
- D programming language

Probes

- *Probes* are programmable sensors or points of instrumentation placed all over the Oracle Solaris system.
- DTrace provides thousands of probes.
- Each probe is associated with an action.
- When the probe fires, certain defined actions are executed.
- A four-tuple (`provider:module:function:name`) uniquely identifies every probe.
  - **Example:** `fbt:zfs:arc_read:entry`

Providers

- Libraries of probes are called *providers*.
- Providers make probes available to the DTrace framework.
- These libraries instrument a specific area of the system or a mode of tracing.
- New providers are added with every release of the operating system.

Consumers

- *Consumers* are user mode programs that call in to the underlying DTrace framework.
- Four consumers are available in the current version of Oracle Solaris:
  - `DTrace(1M)`
  - `lockstat(1M)`
  - `plockstat(1M)`
  - `intrstat(1M)`

D Language

- Tracing programs, also referred to as *scripts*, are written in the D programming language.
- The language is a subset of C with added functions and variables that are specific to tracing.
- Additional capabilities of the D language include the following:
  - Supports ANSI C operators, strings, pointers, struct, and unions
  - Consists of expressions based on built-in variables: `pid`, `execname`, `timestamp`, and `curthread`
  - Performs arithmetic only on integers in D programs (floating-point arithmetic is not permitted in D)
  - Consists of CLI and scripting modes

DTrace Toolkit

- The DTrace Toolkit is a collection of over 230 DTrace scripts and one-liners for performance observability and troubleshooting.
- The toolkit contains:
  - Scripts
  - Man pages
  - Sample documentation
  - Notes files
  - Tutorials
- To install the DTrace toolkit:

```
# pkg install dtrace-toolkit
```

DTrace Toolkit: Important Scripts

Script Folder	Function
`Apps`	For certain applications (such as Apache and NFS)
`CPU`	Measuring CPU activity
`Disk`	Analyzing I/O activity
`Extra`	For other categories
`Kernel`	For kernel activity
`Locks`	Analyzing locks
`Mem`	Analyzing memory and virtual memory
`Net`	Analyzing activity of network interfaces and the TCP/IP stack
`Proc`	Analyzing activity of a process
`System`	Measuring systemwide activity
`User`	Monitoring activity by UID
`Zones`	Monitoring activity by zone

Before Using DTrace

Consider the following options before using DTrace:

- Perform a sanity check first.
  - `/var/adm/messages`
- Start with the “Big 5” stat tools.
  - `vmstat`, `mpstat`, `iostat`, `prstat`, and `netstat` for memory leaks
- Answer the following high-level questions for a quick initial diagnosis:
  - Is there a significant number of cache misses?
  - How much time is spent in user mode compared with system mode?
  - Is the system short on memory or other critical resources?
  - What are the system’s I/O characteristics?

Launching DTrace

Before launching DTrace, consider the following facts:

- Only `root` is allowed to run DTrace by default.
- Non-`root` users need one or more of the following privileges to access DTrace:
  - `DTrace_kernel`
  - `DTrace_proc`
  - `DTrace_user`

```
% ppriv -l | grep Dtrace
DTrace_kernel DTrace_proc DTrace_user
```

DTrace: Example

Scenario: Your system processes are causing a massive number of system calls and subsequently having a negative effect on system performance. You want to investigate the number of system calls that the processes are making.

```
# dtrace –n 'syscall::read:entry (@[execname] = count()}'
dtrace: description 'syscall::read:entry ' matched 1 probe
^C
gnome-settings-d                                  1
in.mpathd                                         2
gnome-terminal                                    3
init                                              4
hald                                              72
hald-addon-acpi                                   72
Xorg                                              248
#
```

Summary

In this lesson, you should have learned how to:

- Monitor system logs
- Identify a crash dump file
- Identify a core dump file
- Troubleshoot a script execution issue
- Troubleshoot a software update failure
- Troubleshoot a network connectivity issue
- Troubleshoot a directory access issue
- Troubleshoot a default shell issue
- Configure the following:
  - System messaging
  - System crash facilities
  - Dump facilities for business application failure
- Explain the role of DTrace in diagnosing system issues