7

# Performing Basic Process Control

# Objectives

After completing this lesson, you should be able to:

- Describe a process and its attributes
- Manage processes

# Agenda

- Describing a Process and Its Attributes
- Managing Processes

# Process: Overview

- A process, also known as a task, is the running form of a program or a shell script.
- Programs and scripts are stored on disk and processes run in memory.
- Processes have a parent/child relationship.
- A running process can `spawn` one or more child processes, or `fork` multiple independent processes.
- Multiple processes can run in parallel.

# Attributes of a Process

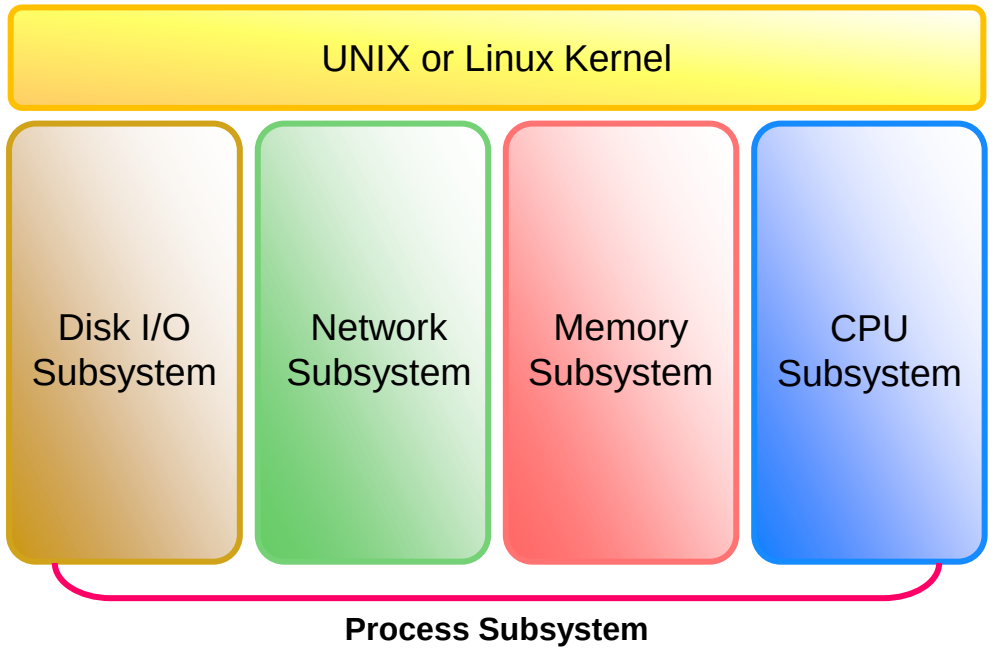
- The kernel assigns a unique identification number to each process called a process ID or `PID`.
  - The kernel uses this `PID` to track, control, and manage the process.
  - Every child `PID` has an owning Parent Process ID (PPID).
- Each process is further associated with a UID and a GID.
  - UIDs and GIDs indicate the process's owner.
  - Generally, the UID and GID associated with a process are the same as the UID and GID of the user who started the process.

# Process States

- The `s`, `stat`, and `state` output specifiers describe the state of a process.
- A process may be in any one of the following states:
  - `D`: Uninterruptible sleep (usually IO)
  - `R`: Running or runnable (on run queue)
  - `S`: Interruptible sleep (waiting for an event to complete)
  - `T`: Stopped, either by a job control signal or because it is being traced
  - `Z`: Defunct (“zombie”) process, terminated but not reaped by its parent
- For more information about the `ps` command process state options, see the `ps` man page.

# Process Subsystems

- Each time you boot a system, execute a command, or start an application, the system activates one or more processes.
- A process, as it runs, uses the resources of the various subsystems:
  - Disk I/O
  - Network
  - Memory
  - CPU



# Agenda

- Describing a Process and Its Attributes
- Managing Processes

# Listing System Processes

- The process status (`ps`) command lists the processes that are associated with your shell.

```
$ ps [options]
```

- For each process, the `ps` command displays the `PID`, the terminal identifier (`TTY`), the cumulative execution time (`TIME`), and the command name (`CMD`).
- List the currently running processes on the system owned by the logon user using the `ps` command.

```
$ ps
 PID    TTY      TIME CMD
 1001   pts/1    0:00 bash
 1004   pts/1    0:00 ps
```

- For more information about the `ps` command options, see the `ps` man page.

# Listing All Processes

Use the `ps -ef` command to list the full-format of all the processes currently scheduled to run on the system.

```
$ ps -ef | less
UID     PID     PPID    C       STIME  TTY      TIME   CMD
root     0       0      0       Feb 13 ?    00:00:18   sched
root     1       0      0       Feb 13 ?    00:00:01   /etc/init -
root     2       0      0       Feb 13 ?    00:00:00   pageout
root     3       0      0       Feb 13 ?    00:17:47   fsflush
root     9       1      0       Feb 13 ?    00:00:00   svc.configd
...(output truncated)
```

# Listing All Processes

**UID:** The username of the owner of the process

**PID:** The unique process identification number of the process

**PPID:** The parent process identification number of the process

**C:** Processor Utilization

**STIME:** The time the process started (hh:mm:ss)

**TTY:** The controlling terminal for the process.

*Note: system processes (daemons) display a question mark (?), indicating the process started without the use of a terminal.*

**TIME:** The cumulative execution time for the process (hh:mm:ss)

**CMD:** The command name, options, and arguments

```
$ ps -ef | less
UID        PID  PPID  C    STIME TTY          TIME CMD
root       216     1  0    Oct 23 ?       00:00:18 /usr/lib/power/powerd
```

# Listing Process Trees in Oracle Linux

- The process tree (`pstree`) command lists the running processes, rooted at either `systemd` or `PID`.

```
$ pstree [-options] [PID | user ]
```

```
$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
oracle      2274  2262  0 14:47 pts/0    00:00:00 /bin/bash
oracle      2308  2274  0 14:47 pts/0    00:00:00 ps -f
$ pstree 2262
gnome-terminal-─┬─bash───pstree
                ├─gnome-pty-helpe
                └─3*[{gnome-terminal-}]
```

- For more information about the `pstree` command options, see the `pstree` man page.

# Listing Process Trees in Oracle Solaris

- The process tree (`ptree`) command lists the running processes, rooted at either `svc.startd` or `PID`.

```
$ ptree [PID | user ]
```

```
$ ps -f
   UID    PID   PPID  C STIME TTY      TIME CMD
oracle    1770  1769  0 14:47 pts/1    0:00 ps -f
oracle    1769  1766  0 14:47 pts/1    0:00 /usr/bin/bash
$ ptree 1766
1766 /usr/bin/gnome-terminal -x /bin/sh -c cd '/home/oracle' && exec $S
  1769 /usr/bin/bash
    1771 ptree 1769
```

- For more information about the `ptree` command, see the `ptree` man page.

# Quiz

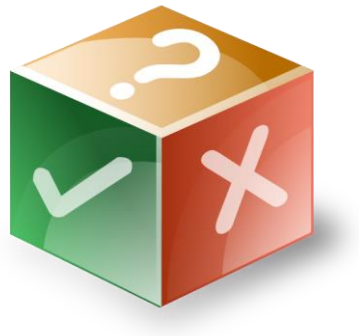
Which of the following is not a process attribute?

a. UID

b. GID

c. PS

d. PID



# Terminating a Process

- There might be times when you need to terminate an unwanted process.
- A process might have entered an endless loop, or it might be hung.
- You can kill or stop any process that you own.
- You can use the following two commands to terminate one or more processes:
  - `kill` *`PID [PID ...]`*
  - `pkill -l [` *`processname | regex_pattern`* `]`
- The `kill` and `pkill` commands send signals to processes directing them to terminate.
- Each signal has a number/value, name, and an associated event.
- For more information about signal values:
  - In Oracle Linux, use `man 7 signal`.
  - In Oracle Solaris, use `man -s3c signal`.

# Terminating a Process: `kill` Command

- You can terminate any process by sending the appropriate signal to the process.
- The `kill` command sends a Termination signal (SIGTERM signal 15) by default to one or more processes.

```
$ kill [-signal] PID [PID ...]
```

**Note:** The `kill` command terminates only those processes that you own.

# Common Signals and Their Uses

- HUP (Hangup - SIGHUP signal 1) is sent to all of the subordinate child processes when the parent process is terminated.
- To terminate a command-line command you usually use a Ctrl + C keyboard command, which sends an Interrupt (SIGINT signal 2), causing it to be interrupted.
- In place of the `exit` command used to exit a terminal session, you can use a Ctrl + D keyboard command, which sends a Quit (SIGQUIT signal 3) to quit the terminal session.
- To stop the command-line command execution, you can use a Ctrl + Z keyboard command to send a Stop (SIGTSTP signal 19) to stop/suspend the foreground execution.

# Terminating a Process: `kill` Command

Use the `kill` command to terminate the `dtmail` process.

```
# ps -e | grep mail
  215 ?       00:00:03 sendmail
12047 ?       00:00:10 dtmail
# kill 12047
# ps -e | grep mail
215 ?       00:00:03 sendmail
```

# Terminating a Process: `pgrep` and `pkill` Commands

- One can use the `pgrep` *`processname`* command to identify the ProcessName or a *regex_pattern* to be killed, and then use the `pkill` command to kill them.

```
$ sleep 500 &
[1] 4378
$ pgrep sleep
4378
$ pgrep -l sleep
4378 sleep
$ pkill sleep
[1]+    Terminated          sleep 500
```

- The `pkill` command requires you to specify the ProcessName or a *regex_pattern* instead of the PID of the process.
- For more information about the `pgrep` and `pkill` command options, see the `pgrep` man page.

# Terminating a Process: `pkill` Command

Use the `pkill` command to terminate the `dtmail` process.

```
# pkill dtmail
# pgrep -l mail
215 ?     00:00:03 sendmail
```

# Forcefully Terminating a Process: Signal 9 (SIGKILL)

- Some processes ignore the default Termination signal (SIGTERM signal 15) that the `kill` command sends.
- If a process does not respond to the Termination signal `15`, you can force it to terminate by sending the Kill signal (SIGKILL signal 9) with either the `kill` or `pkill` commands.

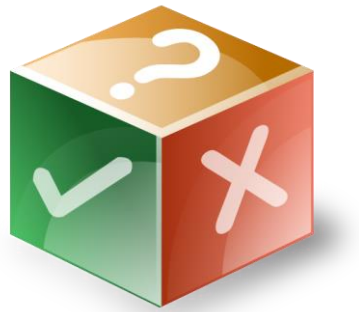
```
$ kill -9 PID
or
$ pkill -9 [ processname | regex_pattern ]
```

**Note:** Sending the Termination signal `15` does not necessarily kill a process gracefully. Only if the signal is caught by the process, it cleans itself up in an orderly fashion and dies. If not, it just dies.

# Quiz

Ordinary users can only `kill` processes they own.

a. True
b. False



# Summary

In this lesson, you should have learned how to:

- Describe a process and its attributes
- Manage system process

# Practice 7: Overview

This practice covers the following topics:

- 7-1: Controlling System Processes