

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
(Estimated time: 50 min.)

Objectives:

- Develop an understanding of UNIX process management
- Review system process concepts
- Review the **ps** command and options
- List processes in the current shell
- List all processes running on the system
- Search for a specific process by command name
- Identify a process to terminate
- Use the **kill** command to terminate a process.
- Find and terminate a process by user
- Terminate a process by command name

Background:

In this lab, you will work with UNIX commands to identify system processes and control them. The UNIX network operating system manages tasks using **processes**. Processes can be initiated by either the operating system or by users. The majority of tasks you perform in the UNIX environment start a process. A process can start or **spawn** a child or subprocess, thus creating a process hierarchy or tree similar to the file system structure with parent / child relationships. You will work with the **ps** (process status) command to monitor system processes and the **kill** command to terminate unwanted process. You will also work with the Solaris commands **pgrep** (process grep) and **pkill** (process kill).

Tools / Preparation:

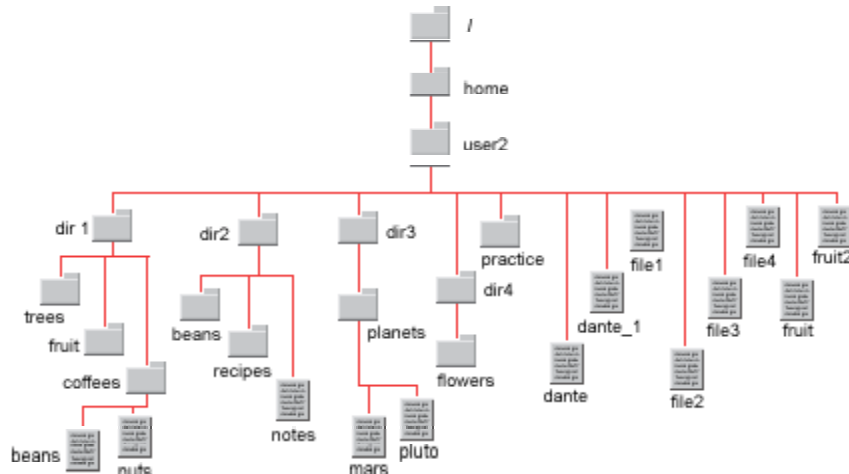
- a) Before starting this lab, review Chapter 13, Section 1 – UNIX Systems Processes, Section 2 – Displaying Processes and Section 3 – Terminating Processes
- b) You will need the following:
 1. A login user ID (e.g. user2) and password assigned by your instructor.
 2. A computer running the UNIX operating system
 3. Networked computers in classroom with class file system installed

Notes:

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
Worksheet

Use the diagram of the sample Class File System directory tree to assist with this lab.

Class File Tree Structure



Step 1. Log in to CDE

Login with the user name and password assigned to you by your instructor in the CDE entry box.

Step 2. Access the Command Line

Right click on the **workspace** backdrop and click on **Tools**. Select **Terminal** from the menu to open a terminal window.

Step 3. Review System Process Concepts

Each program you run creates a process, which is assigned a unique **process identification number (PID)**. The PID is used by the system to identify and track the process until it has completed. The operating system (OS) kernel manages the initiation and termination of all processes. Every process requires system resources such as central processing unit (CPU) time and random access memory (RAM) space to work in. The OS allocates these system resources to each process when it starts and de-allocates them when the process ends. The first two processes started when a UNIX system is booted are the **sched** (scheduler) and **init** (initialization), which manage other processes. There are several different types of processes on a UNIX system. These are summarized below:

Daemon - Daemons are processes that are started by the UNIX kernel and exist for a specific purpose. For instance, the **lpsched** daemon exists for the sole purpose of handling print jobs.

Parent - A process that spawns another process is referred to as its parent. A process called **init** daemon is the first one invoked. Every process except init has a parent process.

Child - A process that is spawned by another process is referred to as a child process.

Orphan – A process whose parent process terminates before it can return its output.

Zombie - A child process that does not return to the parent process with its output. This process becomes "lost" in the system.

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
Worksheet – Cont.

Using the information above, fill in the blanks in the following sentences.

- a. Nearly every process that starts on a UNIX system get assigned a unique _____ by the kernel which is used to track the process from start to finish.
- b. When new processes start the kernel also assign system resources such as **CPU time** and _____.
- c. A process that never returns to the parent with its output is called a _____ process
- d. A process that is spawned by a parent process is called a _____ process.
- e. A _____ process is one that spawns another process.
- f. A UNIX system process that runs to provide services is a: _____
- g. If a parent process ends before the child can finish, it creates an _____ process

Step 4. Review the `ps` Command and Options

The `ps` (process status) command is used to list the processes currently running on the system. This is normally done if a process is taking too long or appears to have stopped as indicated by a terminal window not responding or "**hanging**." By listing the processes, you can see the name of the command or program that initiated the process plus any child processes it may have spawned. By executing the `ps` command more than once, you can see if a process is still running by looking at the **time** for the process, which is the amount of CPU time the process is using. If the amount of time does not increase, then the process may have stopped. You can use the `ps` command to check the process ID (**PID**) of the process and then "**kill**" the process if it is taking too long or has stopped.

The output of the `ps` command will display the PID number and the command or program associated with it. The PID number is normally used to terminate a process. There are three main options with the `ps` command as shown in the table:

Command Format: `ps` ***[-options]***

ps Command Options

<code>ps</code> Option	Meaning	Function or Purpose
<code>ps</code>	No Options	Display information for current user processes in current shell or terminal window
<code>ps -e</code>	Every	Display information about every process on the system.
<code>ps -f</code>	Full	Generate a full listing with all available information on each process.
<code>ps -u userid</code>	User	Display all processes for a particular user

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
Worksheet – Cont.

The basic **ps** command displays the information about process in you current shell only. You will only see processes that have been initiated with this terminal window.

PID	TTY	TIME	CMD
785	pts/6	0:45	dbprog
742	pts/6	0:00	csch
689	pts/6	0:00	/bin/ksh

- a. From your current terminal window, practice using the **ps** command with each of the options shown.

The **ps -ef** command displays all information about every process running on the system.

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	0	0	80	16:46:41	?	0:01	sched

The following table defines the Column Headings for the **ps -ef** Command

ps -ef Column Headings

Value	Description
UID	The user ID of the user that initiated the process.
PID	The process identification number of the process. The PID is used to kill a process.
PPID	The parent process identification number of the process
C	The priority of the process
STIME	Start time for the process
TTY	Terminal type - the controlling terminal for the process
TIME	The amount of CPU time used by the process
CMD	The command name or daemon (name of the program executed)

Step 5. List Processes in the Current Shell

- a. In your current terminal window issue the **ps** command with no options. What information is displayed? _____
- b. How many processes were displayed? _____
- c. What was the process ID (PID) ? _____
- d. What was the command (CMD) that started the process ? _____

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
Worksheet – Cont.

Step 6. List All Processes Running on the System

The `ps -ef` command will list all processes and can produce a fairly long listing.

- a. In your current terminal window issue the `ps -ef` command. What headings are displayed?
(Tip: You may want to pipe the `ps -ef` command to the `more` command to view the headings.)

- b. How many processes were displayed? _____
- c. Count the number of processes by running the `ps -ef` command again and then pipe the output to the `wc` (word count) command). The first number is the number of lines displayed, which is also the number of processes. What command did you use? _____ How many processes were running? _____
- d. Display the output a page at a time by piping it through the `more` command. What command did you use? _____
- e. What is the command that has process ID number 1? _____

Step 7. Search for a specific Process by Command Name

In order to stop a process you must find the Process ID. On most systems there are hundreds of processes running and the `ps -ef` listing can be quite long. If you know the name of the executable program that started the process, you can find the PID faster. By piping the output of the `ps` command through `grep`, you can search for the specific process you want to terminate and determine the correct PID. As you will recall, the `grep` command can search for any type of character string in the output of another command. Specific to Solaris, is the `pgrep` (process grep) command used to search for a specific process. The `-l` (long output) option will display the names of the processes associated with the PID found. The `-e` option displays the PID and the name of the initiating command, which allows `grep` to search on this information.

- a. In your current terminal window issue the `ps -e | grep lp` command to look for all processes that are related to the line printer scheduler daemon.
- b. How many processes were displayed? _____
- c. What is the lowest process ID number of the processes displayed? _____
- d. In your current terminal window issue the `pgrep -l lp` command to look for all processes that are related to the line printer scheduler daemon. What is the difference in output between `ps` and `pgrep`? _____

Step 8. Identify a Process to Terminate.

The `ps -ef` command displays a full listing of every process, including the **Process ID (PID)** and its **Parent Process ID (PPID)**. When trying to terminate a program or release a hung terminal window, it may not be enough to kill the process ID that is associated with the unresponsive application. It may be necessary to kill the Parent of that process and on rare occasions even the Parent of the Parent. It is important to be able to look at a PID and PPID to be able to trace from the child up the hierarchy to the parent processes that spawned them.

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
Worksheet – Cont.

To do this, you must first identify the PID of the lowest level unresponsive process. Normally you would try to kill that processes PID. If this does not stop the process, you may need to kill its parent. Killing a parent process will kill all child processes spawned by it. It is also much quicker to kill a parent process rather than killing perhaps several child processes.

- a. From the current terminal window in CDE, enter the `ps` command.
- b. How many processes were running? One. Why are there so few processes?

- c. What is the name of the process running and what does it represent? _____
- d. What is the Process ID (PID) of this process? _____
- e. Enter the command `csh` to open a **C Shell** session under the Korn Shell. What does you prompt look like now? _____
- f. Enter the command to display **full** information on processes running in the shell. What command did you use? _____ What processes are running now _____
- g. Is the Process ID of the Korn Shell (/bin/ksh) the Parent Process ID (PPID) of the C Shell (csh)?

- h. Enter the command `sleep 1000 &` to create a process that suspends execution for 1000 seconds (Appx 15 min.). The **ampersand (&)** runs the command in the background and returns the shell prompt so you can continue working.
- i. Enter the `ps -f` command again. Is the Process ID of the C Shell (csh) the Parent Process ID (PPID) of the sleep command? _____. Which process ID is the child of the C Shell?

- j. Exit the **C shell** and type the `ps -f` command again. What process ID (PID) is the **parent** of the `sleep` command? _____ What type of process is sleep now? _____

Step 9. Use the `kill` Command to Terminate a Process.

Signals are used to terminate, suspend, and continue processes. Using **Ctrl-c** can sometimes terminate a process that is not responding. This sends an interrupt (INT) signal to the process, terminating it and any child processes it might have spawned.

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
Worksheet – Cont.

The **kill** command provides a direct way to terminate unwanted command processes. It is useful when you want to stop a command that takes a long time to run, or when you need to terminate a process that you cannot quit in the normal way. Specifying their process id normally kills processes.

Command Format: ***kill [-signal] process-id***

To terminate a process with the **kill** command, you would first type **ps** to find out the PID(s) for the process(es) and then type **kill** followed by the PID(s). If you use the **kill** command without specifying a signal, **signal 15** (SIGTERM) is sent to the process with the specified PID number. This is referred to as a **soft kill** and usually causes the process to terminate. It is best to soft kill a process, if possible, since it closes files properly and terminates the process(es) gracefully.

If you need to forcibly terminate a process, you can use the **-9** option to the **kill** command. This option is referred to as a **sure kill** and is necessary for killing shells that will not respond to any other signal to terminate.

Command Format: ***kill -9 Process-id***

Note - For processes other than shells, use the **kill -9 (SIGKILL)** command as a last resort because it is an abrupt method and does not allow for proper process termination.

- a. Enter the command to display **full** information on processes running in the shell. What command did you use? _____ What processes are running now? _____
- b. Since the sleep process is now an orphan and has been adopted by the init process (PID #1), Enter the command to perform a soft kill on the PID for sleep. If sleep has ended (more than 15 minutes) repeat steps 7h through 7j again and then soft kill the sleep program. Enter the **ps -f** command again. What processes are running now? _____
- c. Enter the command to display **full** information on processes running in the shell. What command did you use? _____. What processes are running now? _____
- d. Enter the command **csh** to open another **C Shell** session under the Korn Shell.
- e. Enter the command to display **full** information on processes running in the shell. What processes are running now? _____
- f. Is the Process ID (PID) of the Korn Shell (/bin/ksh) the Parent Process ID (PPID) of the C Shell (csh)? _____
- g. Enter the command **sleep 1000 &** again.
- h. Enter the **ps -f** command again. Is the Process ID of the C Shell (csh) the Parent Process ID (PPID) of the sleep command? _____
- i. Use the **soft kill** command to kill the C shell process ID (PID). Use **ps -f** again to find out if you killed the shell. Did it die? _____ Why or why not? _____
- j. Use the **sure kill** command to kill the C Shell PID. What was the response from the **kill** command? _____

Fundamentals of UNIX
Lab 13.3.2 – Managing System Processes
Worksheet – Cont.

Step 10. Find and Terminate a Process by User

The **ps** command can be used with the **-u** (user) option to find processes for a specific user. You may find processes for users by their **login name** or **UID** number. A user can only terminate their processes, but the **superuser** can terminate any process running on the system

Command Format: **ps -u login-ID or UID**

- a. Open another C Shell and run the **sleep 500 &** command again.
- b. Use the **id** command to determine your numeric user ID (UID) is. What is your numeric UID?

- c. Use **ps** command with the **-u** option to find all processes running for your login ID (e.g.: userX) or your numeric UID (e.g. 1004). What is the process ID for the **sleep** command? _____
- d. Use a **soft kill** to terminate the **sleep** process. Use the **ps -f** command again. Is it Dead?
_____ Exit the C Shell.

Step 11. Terminate a Process by Command Name

The **kill** command is specific to Solaris and works exactly like the **pgrep** command, except that it terminates the process by matching process or processes command name (CMD) and sending a kill signal.

Command Format: **kill CMD name**

- a. Open another **C Shell** and run the **sleep 500 &** command again.
- b. Use the **kill** command to terminate the **sleep** process by its command name. Use the **ps -f** command again. Is it Dead? _____
- c. Exit the C Shell.

Step 11 – Remove Files and Directories Created in this Lab

Remove all files and directories created in you home directory during this lab.

Step 12. Close the Terminal Window and Logout

Double click on the dash button in the upper left corner of the screen, then click the **EXIT** icon on the front panel.