**System processes**

**A process is created in memory when a program or command is executed. A unique identification number, known as process identification (PID), is allocated to it, which is used by the kernel to manage the process until the program or command it is associated with, terminates. When a user logs in to the system, shell is started, which is a process. A process is any program, application or command that runs on the system.**

Several processes are started at system boot up, many of which sit in memory and wait for an event to trigger a request to use their service. These background system processes are called **daemons** and are critical to system functionality.

**Viewing System processes**

There are two commands commonly used to view currently running processes. These are **ps** (process status) ant **top**.

The ps command without any options or arguments, lists processes specific to the terminal where the ps command is run:

**# ps**

PID TTY TIME CMD

4306 pts/1 00:00:00 bash

4319 pts/1 00:00:00 ps

The output has for columns: PID of the process in the first column, terminal the process belongs to in the second column, cumulative time the process given by the system CPU in the third column and actual command being executed in the last column.

Two options –e (every) and –f (full) are popularly used to generate detailed information on every process running in the system. **Check the ps command man pages for more options.**

**# ps -ef**

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 11:29 ? 00:00:01 init [5]

root 2 1 0 11:29 ? 00:00:00 [migration/0]

root 3 1 0 11:29 ? 00:00:00 [ksoftirqd/0]

root 4 1 0 11:29 ? 00:00:00 [events/0]

root 5 1 0 11:29 ? 00:00:00 [khelper]

root 21 1 0 11:29 ? 00:00:00 [kthread]

root 25 21 0 11:29 ? 00:00:00 [kblockd/0]

root 26 21 0 11:29 ? 00:00:00 [kacpid]

root 222 21 0 11:29 ? 00:00:00 [cqueue/0]

Now let’s see what happens when you run a command like find, to search for files ending with .bak in / :

**# find / -type f -name \*.bak**

/etc/mail/submit.cf.bak

…………………………………………………

**# ps -ef**

UID PID PPID C STIME TTY TIME CMD

root 4306 4303 0 11:31 pts/1 00:00:00 bash

root 4322 1 0 11:31 ? 00:00:00 gnome-screensaver

root 4889 4303 0 11:49 pts/2 00:00:00 bash

**root 4917 4306 8 11:49 pts/1 00:00:00 find / -type f -name \*.bak**

root 4918 4889 0 11:49 pts/2 00:00:00 ps –ef

The second method to view process information is the **top** command, which displays additional information including CPU and memory utilization. A sample output from a running **top** session is shown below:

top - 11:53:10 up 23 min, 3 users, load average: 0.11, 0.10, 0.10

Tasks: 117 total, 2 running, 115 sleeping, 0 stopped, 0 zombie

Cpu(s): 8.0%us, 3.3%sy, 0.0%ni, 87.7%id, 0.0%wa, 1.0%hi, 0.0%si, 0.0%st

Mem: 1921784k total, 704436k used, 1217348k free, 76548k buffers

Swap: 2048276k total, 0k used, 2048276k free, 426104k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

4064 root 15 0 218m 11m 6236 S 8.3 0.6 0:15.84 Xorg

4303 root 15 0 262m 15m 8480 R 2.3 0.8 0:01.84 gnome-terminal

5029 root 15 0 12744 1080 808 R 0.3 0.1 0:00.02 top

1 root 15 0 10352 700 588 S 0.0 0.0 0:01.69 init

2 root RT -5 0 0 0 S 0.0 0.0 0:00.00 migration/0

3 root 34 19 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0

4 root 10 -5 0 0 0 S 0.0 0.0 0:00.42 events/0

5 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 khelper

21 root 11 -5 0 0 0 S 0.0 0.0 0:00.00 kthread

25 root 10 -5 0 0 0 S 0.0 0.0 0:00.07 kblockd/0

26 root 20 -5 0 0 0 S 0.0 0.0 0:00.00 kacpid

222 root 11 -5 0 0 0 S 0.0 0.0 0:00.00 cqueue/0

225 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 khubd

227 root 10 -5 0 0 0 S 0.0 0.0 0:00.00 kseriod

292 root 15 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd

293 root 16 0 0 0 0 S 0.0 0.0 0:00.00 pdflush

294 root 15 0 0 0 0 S 0.0 0.0 0:00.09 pdflush

Find out more information on top with the **man top** command.

**Process States**

There are several states that a process can go through in its life. There are five process states: running, waiting, sleeping, stopped and zombie.

* + The running state shows that the process is currently being executed by the system CPU.
  + The sleeping state means shows that the process is currently waiting for input from user or another process.
  + The waiting state means that the process has received input it has been waiting for and is now ready to run as soon as its turn comes.
  + The stopped state indicates that the process is currently halted and will not run even when its turn comes, unless it is sent a signal.
  + The zombie state determines that the process is dead. A zombie process exists in process table just as any other process entry, but takes up no resources. The entry for zombie is retained until the parent process permits it to die.

**Process Priority**

Process priority is determined using the nice value. The system assigns a nice value to a process when it is initiated to establish priority. A total of 40 nice values exist with -20 being the highest and +19 the lowest. Most system started processes use the default nice value of 0. A child process inherits the nice value of its parent process.

Use the ps command and specify the –l option to determine nice values of running processes. See associated nice values for each process under the “NI” column.

A different priority may be assigned to a program or command at the time it is initiated. For example, to run system-config-users with higher priority:

**#nice -2 system-config-users**

The value assigned with nice is relative to the default nice value. The number -2 is added to 0, which means the specified program will run at a higher priority since its nice value is -2.

**Signals and Their Use**

A system runs several processes simultaneously. Sometimes it becomes necessary to pass a notification to a process alerting it of an event. A user or the system uses a signal to pass that notification to a process. A signal contains a signal number and it’s used to control processes. There are a number of signals available for use but most of the time you only deal with only a few of them. A list of available signals can be displayed with the **kill** command using the ***–l*** option.

The commands used to pass a signal to a process are **kill** and **pkill.** These commands are usually used to terminate a process. Ordinary users can kill processes they own, while root can kill any process.

The syntax of the kill command to kill a process is:

**#kill PID**

**#kill –s <signal name or number> PID**

**Homework : read the manual pages of the pkill and kill command.**