**Processes**

* Whenever you enter a command at the shell prompt, it invokes a program. While this program is running it is called a process. Your login shell is also a process, created for you upon logging in and existing until you logout.
* LINUX is a multi-tasking operating system. Any user can have multiple processes running simultaneously, including multiple login sessions. As you do your work within the login shell, each command creates at least one new process while it executes.
* Process id: every process in a LINUX system has a unique PID - process identifier.
* [ps](http://www.mhpcc.edu/training/vitecbids/UnixIntro/man/ps.txt) - displays information about processes. Note that the ps command differs between different LINUX systems - see the local ps man page for details.

To see your current shell's processes:

**% ps**

PID TTY TIME CMD

26450 pts/9 0:00 ps

66801 pts/9 0:00 -csh

To see a detailed list of all of your processes on a machine (current shell and all other shells):

**% ps uc**

USER PID %CPU %MEM SZ RSS TTY STAT STIME TIME COMMAND

jsmith 26451 0.0 0.0 120 232 pts/9 R 21:01:14 0:00 ps

jsmith 43520 0.0 1.0 300 660 pts/76 S 19:18:31 0:00 elm

jsmith 66801 0.0 1.0 348 640 pts/9 S 20:49:20 0:00 csh

jsmith 112453 0.0 0.0 340 432 pts/76 S Mar 03 0:00 csh

To see a detailed list of every process on a machine:

**% ps ug**

USER PID %CPU %MEM SZ RSS TTY STAT STIME TIME COMMAND

root 0 0.0 0.0 8 8 - S Feb 08 32:57 swapper

root 1 0.1 0.0 252 188 - S Feb 08 39:16 /etc/init

root 514 72.6 0.0 12 8 - R Feb 08 28984:05 kproc

root 771 0.2 0.0 16 16 - S Feb 08 65:14 kproc

root 1028 0.0 0.0 16 16 - S Feb 08 0:00 kproc

{ lines deleted }

root 60010 0.0 0.0 1296 536 - S Mar 07 0:00 -ncd19:0

kdr 60647 0.0 0.0 288 392 pts/87 S Mar 06 0:00 -ksh

manfield 60968 0.0 0.0 268 200 - S 10:12:52 0:00 mwm

kelly 61334 0.0 0.0 424 640 - S 08:18:10 0:00 twm

sjw 61925 0.0 0.0 552 376 - S Mar 06 0:00 rlogin kanaha

mkm 62357 0.0 0.0 460 240 - S Feb 08 0:00 xterm

ishley 62637 0.0 0.0 324 152 pts/106 S Mar 06 0:00 xedit march2

tusciora 62998 0.0 0.0 340 448 - S Mar 06 0:05 xterm -e

dilfeath 63564 0.0 0.0 200 268 - S 07:32:45 0:00 xclock

tusciora 63878 0.0 0.0 548 412 - S Mar 06 0:41 twm

* [kill](http://www.mhpcc.edu/training/vitecbids/UnixIntro/man/kill.txt) - use the kill command to send a signal to a process. In most cases, this will be a kill signal, hence the command name. However, other types of signals are usually supported. Note that you can only kill processes which you own. The command syntax is:

**kill [-signal] process\_identifier(PID)**

Examples:

**kill 63878 *- kills process 63878***

**kill -9 1225 *- kills (kills!) process 1225. Use if***

***simple kill doesn't work.***

**kill -STOP 2339 *- stops process 2339***

**kill -CONT 2339 *- continues stopped process 2339***

**kill -l *- list the supported kill signals***

You can also use CTRL-C to kill the currently running process.

* Suspend a process: Use CTRL-Z.
* Background a process: Normally, commands operate in the foreground - you can not do additional work until the command completes. Backgrounding a command allows you to continue working at the shell prompt.

To start a job in the background, use an ampersand (&) when you invoke the command:

**myprog &**

To put an already running job in the background, first suspend it with CRTL-Z and then use the "bg" command:

**myprog *- execute a process***

**CTRL-Z *- suspend the process***

**bg *- put suspended process in background***

* Foreground a process: To move a background job to the foreground, find its "job" number and then use the "fg" command. In this example, the jobs command shows that two processes are running in the background. The fg command is used to bring the second job (%2) to the foreground.

**jobs**

[1] + Running xcalc

[2] Running find / -name core -print

**fg %2**

* Stop a job running in the background: Use the jobs command to find its job number, and then use the stop command. You can then bring it to the foreground or restart execution later.

**jobs**

[1] + Running xcalc

[2] Running find / -name core -print

**stop %2**

* Kill a job running in the background, use the jobs command to find its job number, and then use the kill command. Note that you can also use the ps and kill commands to accomplish the same task.

**jobs**

[1] + Running xcalc

[2] Running find / -name core -print

**kill %2**

* Some notes about background processes:
  + If a background job tries to read from the terminal, it will automatically be stopped by the shell. If this happens, you must put it in the foreground to supply the input.
  + The shell will warn you if you attempt to logout and jobs are still running in the background. You can then use the jobs command to review the list of jobs and act accordingly. Alternately, you can simply issue the logout command again and you will be permitted to exit