

p6 Getting Started Activity

Learn a few more Linux commands and some steps you will need to complete p6.

1. Launch a new Terminal (referred to here as Terminal 1)

1. Remote connect to any CSL machine.
2. Type **users** to see which (and how many) users are connected to this machine.
3. Make a note of the machine name that you are connected to in Terminal 1.
Tip: the machine name is shown in the user prompt and it is not "best-linux"
4. Write a C program that runs an infinite loop without any output.
 1. Compile and run your program.
Note: if you did the above correctly, the program is just running and there is no visible output.
 2. Type **Ctrl-z** to suspend the currently running program. (Don't use Ctrl-C here)
 3. Type **ps -u** to display the list of your running processes.
 1. You should see your suspended process in the list of running processes.
 2. Note: the **Process ID <pid>** of your running process.
 4. Type **fg** to bring your suspended program back to the *foreground*.
 5. Leave your infinite loop process running and continue with next instructions

2. Launch a second Terminal connected to the same machine as your first (referred to here as Terminal 2)

1. Remote connect to the same machine from a second terminal window.

```
ssh CSLOGIN@machine-01.cs.wisc.edu
```

where *CSLOGIN* is your CS user name and *machine-01* is the name of the machine you connected to in step 1.

2. Resize and reposition your terminal windows so you can see them both at same time.
3. In Terminal 2:
 1. Type **users** to display the users on the machine.
You should see your login name listed twice.
 2. Display the running processes. (Use same command as above).
 1. You should see the process (name of the program) that you started in the other terminal.
 2. And it should have the same <pid> as you noted above.
 3. Type **kill <pid>** where <pid> is the process ID from the infinite loop process started in the other terminal and the angle brackets are not typed as part of the command.

4. Notice that the running process that was started in the other window (on the same machine) has been terminated.

Example Screenshot:

Note: both terminal windows must be connected to same machine to see the same processes on both.

The screenshot shows two terminal windows connected to the same machine (best-linux.cs.wisc.edu). The left window shows a user listing processes and then killing a process. The right window shows a user listing processes and then killing a process.

Left Terminal Window:

```
rockhopper-04[~/private/cs354/w12] (127)%users
ams deppeler hemmila
rockhopper-04[~/private/cs354/w12] (128)%ls
a.out main.c
rockhopper-04[~/private/cs354/w12] (129)%./a.out
^Z
Suspended
rockhopper-04[~/private/cs354/w12] (130)%ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
deppeler 1962739  0.0  0.0 21512 3972 pts/4    Ss   Apr08   0:00 -tcsh
deppeler 2004050 53.6  0.0 2332  580 pts/4    T    00:21   0:04 ./a.out
deppeler 2004130  0.0  0.0 14156 2952 pts/4    R+   00:21   0:00 ps -u
rockhopper-04[~/private/cs354/w12] (131)%fg
./a.out
Terminated
rockhopper-04[~/private/cs354/w12] (132)%
```

Right Terminal Window:

```
rockhopper-04[~/private/cs354/w12] (127)%users
ams deppeler deppeler hemmila
rockhopper-04[~/private/cs354/w12] (128)%ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
deppeler 1962739  0.0  0.0 21512 3972 pts/4    Ss   Apr08   0:00 -tcsh
deppeler 2004050 78.1  0.0 2332  580 pts/4    R+   00:21   0:44 ./a.out
deppeler 2004287  0.0  0.0 18440 3472 pts/3    Ss   00:21   0:00 -tcsh
deppeler 2004342  0.0  0.0 14156 2948 pts/3    R+   00:22   0:00 ps -u
rockhopper-04[~/private/cs354/w12] (129)%kill 2004050
rockhopper-04[~/private/cs354/w12] (130)%ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
deppeler 1962739  0.0  0.0 21512 3972 pts/4    Ss+  Apr08   0:00 -tcsh
deppeler 2004287  0.0  0.0 18440 3472 pts/3    Ss   00:21   0:00 -tcsh
deppeler 2004387  0.0  0.0 14156 3036 pts/3    R+   00:22   0:00 ps -u
rockhopper-04[~/private/cs354/w12] (131)%
```

Now, you are ready to create programs that can signal each other and handle specific signals received.