How do I view, kill, or nice processes I have running on a Linux system?

Scope

Much of the information in this document is relevant to all Linux systems. However, there are some commands mentioned (including killpid, killbyname, and killme) that may only be available on the Unified Linux Systems owned and managed by the School of Informatics and Computing.

Viewing Processes

The normal command used to see what processes you have running on Linux system is `ps`. The problem is that, by default, the `ps` command typically does not show you all of your processes so it may be confusing. If you run `ps` with no arguments, it will usually only show you the processes that you own and that have what is called the same "controlling terminal" as your current shell. If you are logged into a GUI on the console (eg. using Gnome or KDE), you have multiple logins to a machine remotely, or you have processes running in the background, then just running `ps` will only show you a subset of the processes you actually have running. To see all of your processes with `ps` you have to use the proper command line flags or, optionally, some other program.

This is further complicated by the fact that the various process display and control programs may behave differently on different variants of Linux. For example, the `ps` command has slightly different usage across platforms which can also be very confusing.

Here is a brief listing of some of the common ways you can see the process you have running.

- **Arguments To `ps`** - There are lots of command line arguments to `ps` and it can get pretty tricky to understand them all (see the `ps` manual page by running "man ps" for all the gory details). However, here are a couple examples of common arguments. You will need to replace `username` with your own username in the examples:

  - `ps -fe` Display a full listing of all processes currently running on the system
  - `ps -feww` Display a full listing of all processes currently running on the system using unlimited width (to display the full process arguments)
  - `ps -fu username` Display a full listing of all processes currently running on the system that are owned by `username`

- **top** - The `top` command is a great tool for displaying processes. Just run `top` and it will show you a periodically updating list of the processes running on the system. The listing will be ordered by CPU usage by default and will display processes from all users, not just you. Just type ‘u’, followed by your username, to display only the processes you own. Type ‘h’ to see the help page and type ‘q’ to quit.

- **pstree** - The `pstree` command displays processes in a tree showing the parent/child relationship between processes. Here are some example usages:

  - `pstree` Display all processes running on the system
  - `pstree -p` Include the process id (pid) of processes
  - `pstree -pl` Include the process id (pid) of processes and don't truncate lines
  - `pstree username` Display only processes owned by `username`

Killing Processes

Now that you have been able to find the processes you have running, you may want to terminate (or `kill`) a process. The normal unix command for killing processes is `kill` but, again, things aren't always as easy as we would like. Once you locate the process using `ps` or `top`, you will be able to determine the Process ID, or PID, of the process and kill it by running:

```
kill PID
```

By default, `kill` sends what is called the SIGTERM signal to the process to tell it to terminate. However, it is possible for the process to ignore this request so this may not actually kill the process. If this fails to kill the process, then you will have to send the process the KILL signal which is a signal the process cannot ignore. You can do this by running:
In these examples, you will replace \textit{PID} with the numeric process id for the process you want to kill.

In addition to the \texttt{kill} command, there are a number of other ways to kill a process that you may find more convenient. These include the following: (note that all of these might not be available on the system you are using)

- \textbf{\texttt{killpid}} - This is a handy little command that tries to kill a process using the default \texttt{TERM} signal first and then, if that doesn't kill it, tries using \texttt{KILL}. For example, to kill \texttt{PID 12345} just run
  
  \texttt{killpid 12345}

- \textbf{\texttt{killbyname}} - This is another handy command that tries to find a named process and kill it. For example, if you had a program called \texttt{example1} running, you could kill it with:
  
  \texttt{killbyname example1}

- \textbf{\texttt{top}} - It is possible to kill processes from within \texttt{top} (mentioned above). While viewing the \texttt{top} output, just hit 'k', followed by the \texttt{PID} of the process you wish to kill. Like \texttt{ps}, \texttt{top} will send the \texttt{TERM} signal to the process which may or may not actually kill it. To send the \texttt{KILL} signal, hit 'k' followed by something like \texttt{-KILL 12345}.

- \textbf{\texttt{killme}} - The \texttt{killme} command finds and kills every process you have running on a machine. You can just run \texttt{killme} with no arguments to kill all your processes. Use with caution since it will kill every process you own on the system, including your current shell, which will also log you out.

Also, keep in mind that not all processes can be killed. There are what are commonly called \textit{zombie processes} that are essentially forked processes that are still hanging around waiting for the parent process to call \texttt{wait()} to clean things up. Such processes commonly show up as \texttt{defu nct} and the only way to get rid of them is for either the parent to call \texttt{wait()} for this process (which you may be able to force by killing the parent process) or just reboot the system. If the parent process is init (pid 1) then your only real option is to reboot.

### Changing Process Priority (nice'ing)

When you are running a cpu-intensive process in the background or on shared systems, you may want to run it with lower priority to limit the impact it has on other users. This is highly encouraged when you are running cpu-intensive, long-running processes on the shared Linux servers. The commands you can use to use to do this are \texttt{nice} and \texttt{renice}.

When you first start a job, you can start it with reduced priority using the \texttt{nice} command. After a process is running, you can then use \texttt{renice} to reduce it's priority on the fly. Below are some example uses of these commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{nice \textit{command}}</td>
<td>Run \textit{command} with reduced priority using the default niceness (level 10)</td>
</tr>
<tr>
<td>\texttt{nice -n 19 \textit{command}}</td>
<td>Run \textit{command} with a niceness of 19 (the highest niceness)</td>
</tr>
<tr>
<td>\texttt{renice 10 1234}</td>
<td>Set the niceness of the running process with pid 1234 to 10</td>
</tr>
<tr>
<td>\texttt{renice 19 1234}</td>
<td>Set the niceness of the running process with pid 1234 to the highest available niceness value, 19</td>
</tr>
</tbody>
</table>

Note that you will not be able to use reduce the niceness value of your processes, only increase it. Only the root user has the ability to lower niceness values.