**Class 7 – Jobs & Process Control**

**Introduction**

A **process** is an instance of an executed program. A **job** is a process running in your current shell in your current login session. You can use several shell commands to manipulate jobs. Once you have closed your current shell or exited your current login session, jobs may continue to run. You will have to use UNIX commands explained below to manipulate any processes you may have left behind.

**Starting Jobs**

You can start a job in the foreground or background. If you run a job in the background and then logout or close the current shell, you will not be able to bring the job to the foreground again. The job can, however, continue to produce output to a file once you have logged out or closed the shell.

**Running Jobs in the Foreground**

To start a job in the foreground, simply type in the command at the Unix prompt and then hit return. When you start a job in the foreground, you will notice that you do not get a prompt back, but instead, your cursor is left hanging below your last prompt. You can only have one job running in the foreground at a time.

**Running Jobs in the Background**

In the UNIX prompt, type in the command followed by an ampersand (&) symbol:

**command &**

When you start a job in the background, you will notice that you do get a prompt back and at that point, you can enter more commands.

**Listing Jobs**

To see a list of all the jobs you have running from your current shell, type:

**jobs**

You will see something like the following:

[1] Running

[2] Stopped

[3] Done

[4] Terminated

stop %1

The first column contains the job number, the second column describes whether the job is running or suspended, and the last column gives the name of the job. The '+' symbol indicates the current job (the job most recently stopped or put into the background). The '-' symbol indicates the previous job.

**Suspending a Job**

There are two ways to suspend a job. First, if a job is running in the foreground, you can put your cursor in the window where you started the job and type ^Z (hold down the Ctrl key and press the 'z' key). This will suspend the job that was running in the foreground.

Second, if a job is running in the background, you can type:

**stop %job#**

You get the job# from the "jobs" command.

The following example will suspend job# 3.

Example: stop %3

Note: This worked in Centos 6 or prior.

For Centos 7, use kill instead of stop

**Restarting a Suspended Job**

Once a job has been suspended, you can restart it in the foreground or background. To restart a job in the foreground, type:

**fg %job#**

The following example will restart job# 3 in the foreground. If you restart a job in the foreground, you will not get a prompt back.

Example: fg %3

**To restart a job in the background, type:**

**bg %job#**

The following example will restart job# 4 in the background. If you restart a job in the background, you will get a prompt back and you will be able to execute more commands.

Example: bg %4

**Killing a Job**

To kill a job, type:

**kill %job#**

The following example will kill job# 4.

Example: kill %1

kill %1

**Cron jobs**

**Guide on this: http://www.cyberciti.biz/faq/how-do-i-add-jobs-to-cron-under-linux-or-unix-oses/**

Cron job is a Linux command for scheduling a task to be executed sometime in the future. This is normally used to schedule a job that is executed periodically - for example, to send out a notice every morning. Some scripts, such as Drupal, may require you to set up a cron job to perform certain functions.

For most cron jobs, there are three components present:

1. The script that is to be called or executed.
2. The command that executes the script on a reoccurring basis.
3. The action or output of the script, which depends on what the script being called does.  Frequently, scripts called as a cron job modify files or databases, however they can perform other tasks that do not modify data on the server, like sending out email notifications.

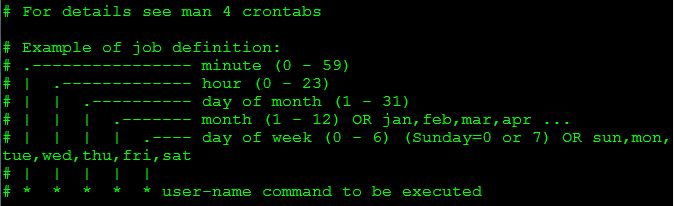
Most scripts that require the use of a cron job will give you specific instructions on what needs to be setup, frequently giving examples.

**crontab –e**

00 09-18 \* \* \* command

* **00** – 0th Minute (Top of the hour)
* **09-18** – 9 am, 10 am,11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm
* **\*** – Every day
* **\*** – Every month
* **\*** – Every day of the week

Examples:<http://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/>

****

**Extra reads on Jobs & Processes in general:** [**http://www.tutorialspoint.com/unix/unix-processes.htm**](http://www.tutorialspoint.com/unix/unix-processes.htm)

**Screen**

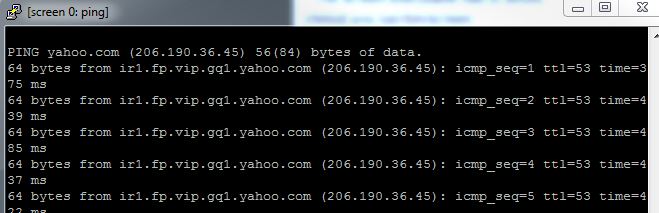
Screen offers the ability to “detach” the result of executing a certain command or script to a different “screen” so that it can be resumed later. Personally I recommend using screen if possible to avoid manipulating the jobs via the built in commands (they just haven’t worked well for the most part in CentOs, feel free to explore for yourself).

To use it, you must first install it with:

**yum install screen**

Sample usage:

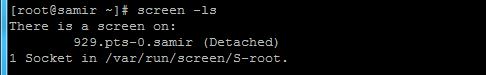
**screen ping yahoo.com** (this will switch the terminal’s view to a “screen” window:



From here, push **CTRL-A** (then let go of keys) then **push d** and it will “**detach**” the screen.



To list the screens that are running, run “**screen –ls**” and you will notice I my detached screen there:



To re-attach that screen the following command is needed:

**screen –r** 929.pts-0.samir

To kill the screen session use:

screen –S 929.pts-0.samir –X quit

More info here (check the comments as well, there is a good one on how to share a session between multiple users):

[**http://www.thegeekstuff.com/2010/07/screen-command-examples/**](http://www.thegeekstuff.com/2010/07/screen-command-examples/)

[**http://www.cyberciti.biz/tips/linux-screen-command-howto.html**](http://www.cyberciti.biz/tips/linux-screen-command-howto.html)

**Introduction to nice and renice**

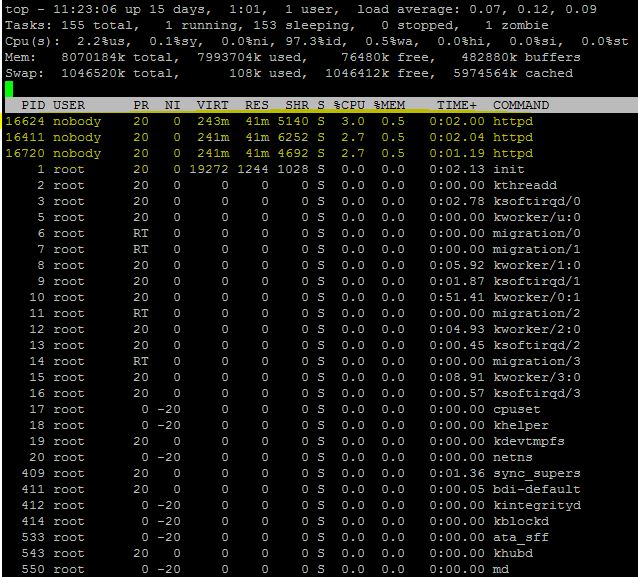
Changing Priority on Linux processes using the **nice** and **renice** commands (manipulate niceness or nice value)

In Linux we can set guidelines for the CPU to follow when it is looking at all the tasks it has to do. These guidelines are called **niceness** or **nice value**. The Linux niceness scale goes from -20 to 19. The lower the number the more priority that task gets. If the niceness value is high number like 19 the task will be set to the lowest priority and the CPU will process it whenever it gets a chance. The default nice value is zero.

By using this scale we can allocate our CPU resources more appropriately. Lower priority programs that are not important can be set to a higher nice value, while high priority programs like daemons and services can be set to receive more of the CPU’s focus. You can even give a specific user a lower nice value for all of his/her processes so you can limit their ability to slow down the computer’s core services.

**Checking the Priority of Running Processes**

The easiest way to get a quick picture of what the current niceness priority for processes is to use the **top** command:



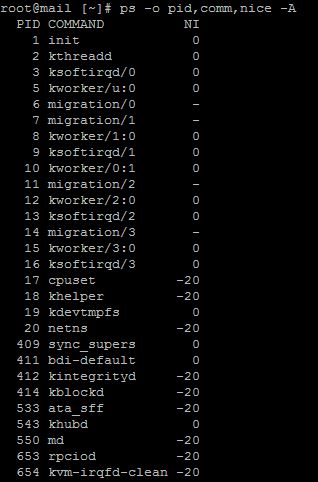
Notice the “NI” column – this shows the niceness value. In this example, there are 3 processes for HTTP (apache web server) which are web site requests which use CPU processing power for a very short amount of time.

Changing the niceness value can come in handy when performing an installation or a backup (basically anything that requires a lot of processing power). On a heavy-duty server, being able to renice (change the niceness value) can be beneficial for admins that want to get the most out of their server.

Another way to get the nice value is to use **ps:**

I used the following command:

**ps -o pid,comm,nice –A**



**Setting priority on new processes**

To change the priority when issuing a new command you use:

**nice –n [nice value] [command]**

Example:

**nice –n 10 yum update**

This will increment the default nice value by a positive 10 for the command “yum update.”

**Setting priority on existing processes**

To alter the nice value of something that is already running, the **renice** command comes into play.

**renice [nice value] –p [process id]:**

renice 10 –p 2123

This will increment the priority of the process with an id 2123 to 10

**Note: only root can apply negative nice values**

**Setting permanent priority on all processes for a specific user**

Sometimes it is helpful to give specific users lower priority than others to keep system resources allocated in the proper places like core services and other programs.

You can set the default nice value of a particular user or group in the **/etc/security/limits.conf** file.

It uses this syntax: **[username] [hard|soft] priority [nice value]**

This is what the /etc/security/limits.conf shows:

