

Homework Assignment No. 05:

## **RUNNING COMPUTE-INTENSIVE JOBS**

Submitted to:

Professor Joseph Picone  
ECE 3822: Software Tools for Engineers  
Temple University  
College of Engineering  
1947 North 12<sup>th</sup> Street  
Philadelphia, Pennsylvania 19122

9/28/2015

Prepared by:

Devin Trejo  
Email: [devin.trejo@temple.edu](mailto:devin.trejo@temple.edu)

## 1. PROBLEM

The purpose of this assignment is to demonstrate how to run jobs on a remote machine. Running jobs remotely is useful when running long, intensive computing scripts. One may want to offload the script or job to a remote machine which is faster and will continue working even after one closes their laptop for the night.

To demonstrate this remote compute capability we start by writing a 'CPU heavy' script where we print the date every hour. After we start the script we show it still runs in the background after we logout of the remote machine. We also show that we can run the job without ever logging into the remote machine.

## 2. APPROACH

To begin we write a script that will run for an extended period of time. Using a combination of commands *'sleep'* and a loop we run the job for a long time. The command *'sleep'* tells the script to wait for the amount of time specified. The value can be in terms of seconds, minutes or hours. In our script we tell it to sleep for one hour and loop ten times. Each time we loop we output the date to stdout using the command *'date'*. This looping script simulates a script that is 'CPU intensive' where we expect the computation to take hours. Using this script we can test remote computing capabilities.

Next we monitor the status of our job in realtime. We want to output our stdout stream to a file instead of the terminal screen. Nohup allow us to redirect our stdout steam to a file which we will call hw05.out. We also output stderr to a file to catch any errors that may occur as our script runs. Lastly we need to disconnect our processor of running the script from our current shell otherwise when we log out all child processes of our shell will also quit. The following command will let us accomplish all the above concerns.

```
nohup <path/to/script>/hw05.sh > hw05.out 2> hw05.err < /dev/null &
```

**Figure 1:** Command format to run our hw05.sh script in the background.

Nohup allow us run a job that won't be interrupted by hangups. Therefore when we disconnect our shell and it sends a HUP signal to all child processes our processes will not terminate. To have our stdout saved we output to a file called *'hw05.out'*. The second output argument is the stderr stream which we can also redirect to a file. All other outputs we output to *'/dev/null'* which will ensure our script runs successfully. The & at the end of the command allows the script to run in the background.

### 3. RESULTS

First we create our script to loop for a very long time and output the date on each loop iteration. In our script we loop ten times where we sleep for one hour each time. Therefore we expect a date timestamp every hour ten times. The script format is as follows:

```
#!/bin/bash

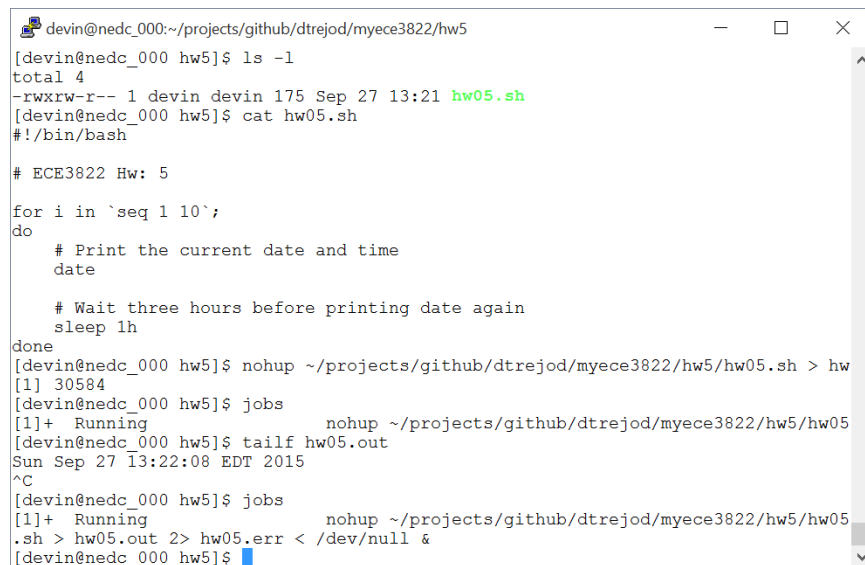
# ECE3822 Hw: 5

for i in `seq 1 10`;
do
    # Print the current date and time
    date

    # Wait one hour before printing date again
    sleep 1h
done
```

**Figure 2:** hw05.sh

Next we use our processes for running a script in the background. We can confirm the job is running by using the command `jobs` to see it running.



```
devin@nedc_000:~/projects/github/dtrejod/myece3822/hw5
[devin@nedc_000 hw5]$ ls -l
total 4
-rwxrwx-r-- 1 devin devin 175 Sep 27 13:21 hw05.sh
[devin@nedc_000 hw5]$ cat hw05.sh
#!/bin/bash

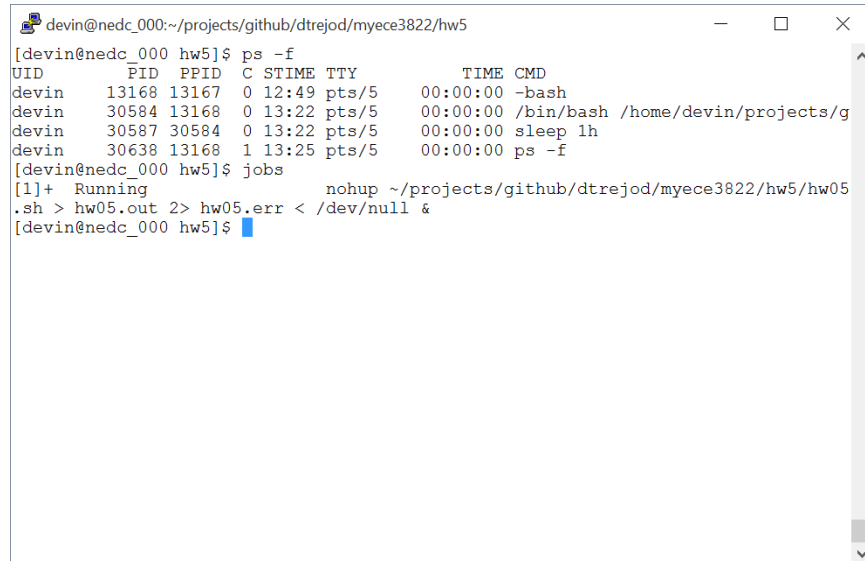
# ECE3822 Hw: 5

for i in `seq 1 10`;
do
    # Print the current date and time
    date

    # Wait three hours before printing date again
    sleep 1h
done
[devin@nedc_000 hw5]$ nohup ~/projects/github/dtrejod/myece3822/hw5/hw05.sh &
[1] 30584
[devin@nedc_000 hw5]$ jobs
[1]+  Running                  nohup ~/projects/github/dtrejod/myece3822/hw5/hw05
[devin@nedc_000 hw5]$ tailf hw05.out
Sun Sep 27 13:22:08 EDT 2015
^C
[devin@nedc_000 hw5]$ jobs
[1]+  Running                  nohup ~/projects/github/dtrejod/myece3822/hw5/hw05
.sh > hw05.out 2> hw05.err < /dev/null &
[devin@nedc_000 hw5]$
```

**Figure 3:** Running the jobs in the background

At this point we can continue using the shell as our intensive job runs in the background. We also can disconnect our current shell and have our jobs still running. Before we disconnect we note the job ID allowing us to find the job after we have disconnected.



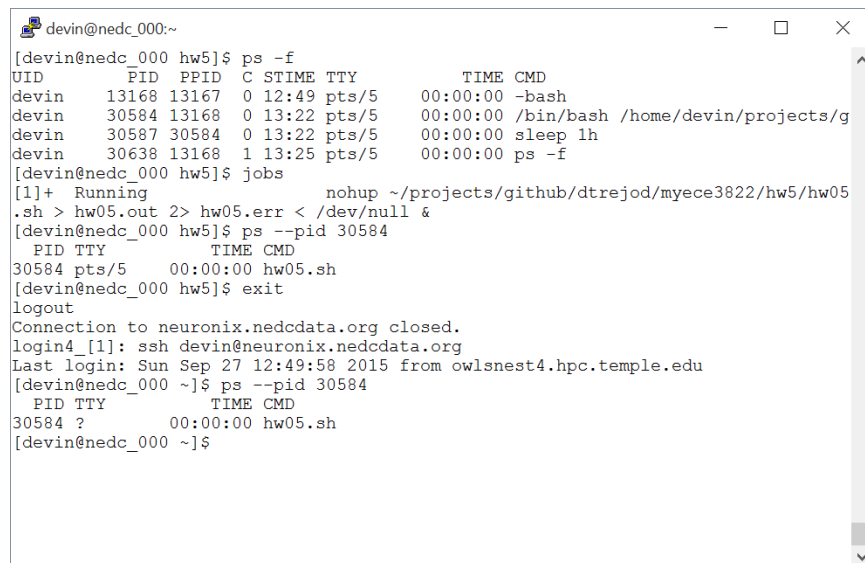
```

devin@nedc_000:~/projects/github/dtrejod/myece3822/hw5
[devin@nedc_000 hw5]$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
devin       13168   13167  0  12:49 pts/5        00:00:00 -bash
devin       30584   13168  0  13:22 pts/5        00:00:00 /bin/bash /home/devin/projects/g
devin       30587   30584  0  13:22 pts/5        00:00:00 sleep 1h
devin       30638   13168  1  13:25 pts/5        00:00:00 ps -f
[devin@nedc_000 hw5]$ jobs
[1]+  Running                  nohup ~/projects/github/dtrejod/myece3822/hw5/hw05
.sh > hw05.out 2> hw05.err < /dev/null &
[devin@nedc_000 hw5]$

```

**Figure 4:** Running 'ps' to find process ID

Next we disconnect and reconnect and see that our job is still running.



```

devin@nedc_000:~
[devin@nedc_000 hw5]$ ps -f
UID          PID    PPID  C  STIME TTY          TIME CMD
devin       13168   13167  0  12:49 pts/5        00:00:00 -bash
devin       30584   13168  0  13:22 pts/5        00:00:00 /bin/bash /home/devin/projects/g
devin       30587   30584  0  13:22 pts/5        00:00:00 sleep 1h
devin       30638   13168  1  13:25 pts/5        00:00:00 ps -f
[devin@nedc_000 hw5]$ jobs
[1]+  Running                  nohup ~/projects/github/dtrejod/myece3822/hw5/hw05
.sh > hw05.out 2> hw05.err < /dev/null &
[devin@nedc_000 hw5]$ ps --pid 30584
   PID TTY          TIME CMD
30584 pts/5        00:00:00 hw05.sh
[devin@nedc_000 hw5]$ exit
logout
Connection to neuronix.nedcdata.org closed.
login4 [1]: ssh devin@neuronix.nedcdata.org
Last login: Sun Sep 27 12:49:58 2015 from owlsnest4.hpc.temple.edu
[devin@nedc_000 ~]$ ps --pid 30584
   PID TTY          TIME CMD
30584 ?            00:00:00 hw05.sh
[devin@nedc_000 ~]$

```

**Figure 5:** Job is still running after closing the ssh session.

In the end we show that our script ran successfully over its entire 10 hour course.

```

devin@nedc_000:~/projects/github/dtrejod/myece3822/hw5
"\"This research was supported in part by the National Science Foundation
through major research instrumentation grant number CNS-09-58854.\""
=====
The Owl's Nest User Guide can be found at:
http://www.hpc.temple.edu/owlsnest/OwlsnestUserGuide.html

Please send questions and reports of problems to hpc@temple.edu
NEWS: 0 news articles, use 'news -a' to read them all
login4_[1]: ssh devin@neuronix.nedccdata.org
Last login: Mon Sep 28 11:50:06 2015 from owlsnest.hpc.temple.edu
(reverse-i-search) '^C
[devin@nedc_000 ~]$ cd ~/projects/github/dtrejod/myece3822/hw5/
[devin@nedc_000 hw5]$ ls
hw05.err  hw05.out  hw05.sh
[devin@nedc_000 hw5]$ cat hw05.out
Sun Sep 27 13:22:08 EDT 2015
Sun Sep 27 14:22:08 EDT 2015
Sun Sep 27 15:22:08 EDT 2015
Sun Sep 27 16:22:08 EDT 2015
Sun Sep 27 17:22:08 EDT 2015
Sun Sep 27 18:22:08 EDT 2015
Sun Sep 27 19:22:08 EDT 2015
Sun Sep 27 20:22:08 EDT 2015
Sun Sep 27 21:22:08 EDT 2015
Sun Sep 27 22:22:08 EDT 2015
[devin@nedc_000 hw5]$ cat hw05.err
[devin@nedc_000 hw5]$

```

Figure 6: Checking on the stdout/error the next day

Alas we conclude by showing we can actually run a script on a remote machine without ever logging into the remote machine. As one knows to login into a remote machine we use the secure shell command 'ssh'. Well one can pass an argument to the 'ssh' command and have it executed on the remote machine. We demonstrate the concept by starting out from our local machine. After we run the code we then login into the remote machine and grep for the process. The processes we started from the local machine is shown to be indeed running.

```

devin@nedc_000:~
=====
For any publication that is based in whole or in part on calculations performed
on this cluster, you are required to include the following acknowledgment:
"\"This research was supported in part by the National Science Foundation
through major research instrumentation grant number CNS-09-58854.\""
=====
The Owl's Nest User Guide can be found at:
http://www.hpc.temple.edu/owlsnest/OwlsnestUserGuide.html

Please send questions and reports of problems to hpc@temple.edu
NEWS: 0 news articles, use 'news -a' to read them all
login4_[1]: ssh devin@neuronix.nedccdata.org '^C
login4_[1]: ssh devin@neuronix.nedccdata.org '^C
login4_[1]: ssh devin@neuronix.nedccdata.org 'nohup /home/devin/projects/github/d
trejod/myece3822/hw5/hw05.sh > hw05_remote.out 2> hw05_remote.err < /dev/null &'

login4_[1]: ssh devin@neuronix.nedccdata.org
Last login: Mon Sep 28 12:21:19 2015 from owlsnest4.hpc.temple.edu
[devin@nedc_000 ~]$ ps -ef | grep hw05.sh
devin    10694 10670   0 11:50 pts/1    00:00:00 vim hw05.sh
devin    11668   1 0 12:23 ?        00:00:00 /bin/bash /home/devin/projects/g
ithub/dtrejod/myece3822/hw5/hw05.sh
devin    11700 11676   0 12:24 pts/4    00:00:00 grep hw05.sh
[devin@nedc_000 ~]$

```

Figure 7: Running a script remotely without every logging into the remote machine.

We can monitor the realtime output of our script by running 'tailf' command on our stdout steam file 'hw05\_remote.out'. Note that since I did not specify the full path for our stdout steam it saved the files in home directory.



```

devin@nedc_000:~
[devin@nedc_000 ~]$ ls -l
total 8
drwxrwxr-x 4 devin devin 31 Sep  2 15:17 cluster-jobs
drwxr-xr-x 3 devin devin 28 Aug 27 13:54 Desktop
drwxr-xr-x 2 devin devin  6 Aug 20 16:36 Documents
drwxr-xr-x 6 devin devin 4096 Sep 14 12:36 Downloads
-rw-rw-r-- 1 devin devin  0 Sep 28 12:23 hw05_remote.err
-rw-rw-r-- 1 devin devin 29 Sep 28 12:23 hw05_remote.out
drwxr-xr-x 2 devin devin  6 Aug 20 16:36 Music
drwxr-xr-x 2 devin devin  6 Aug 20 16:36 Pictures
drwxrwxr-x 5 devin devin 43 Aug 26 15:34 projects
drwxr-xr-x 2 devin devin  6 Aug 20 16:36 Public
drwxr-xr-x 2 devin devin  6 Aug 20 16:36 Templates
drwxr-xr-x 2 devin devin  6 Aug 20 16:36 Videos
[devin@nedc_000 ~]$ tailf hw05_remote.out
Mon Sep 28 12:23:41 EDT 2015

```

**Figure 8:** Monitoring the real-time status of our output.

#### 4. ANALYSIS

What we have shown in this homework task is the powerful capabilities of running jobs remotely. Today the emphasis on computing power is being reserved to big computing farms located in some remote location. The local machine one works from is simply a terminal or script editing station. The real computation of the script is reserved for more powerful machines loaded with faster processors and large amounts of RAM. One may never see the actual machine the remote machine is running off of. However as long as it is accessible via a remote connection, use the remote machines as if it were a local machine.

The task asked was successfully accomplished in that we ran a script remotely in the background for an extended period of time. We showed the processes of redirecting the output/error streams, which would typically populate the terminal screen, to a file. Having a file that separates the output and errors is valuable when one is trying to find where their script broke down. The error would typically be buried in the terminal output stream.

Next we showed that the script is disconnected from our shell allowing the script to run even after closing the 'ssh' connection. This idea comes handy so that one does not need to have their local machine running just to execute a command on a remote machine. In fact we showed that one does not even need to login to the remote machine and instead can start and execute a script all from their local machine. Depending on a user's flow either approach is shown to be viable.