NCS 490:
Bash Lab

By: Tony Perez

October 14, 2014

1 Abstract

In this lab we learned how to write bash scripts and also schedule then using the cron daemon. We used the program called the crontab to schedule these scripts at certain times. We have already dealt with scripting in our previous lab to run certain commands on start-up. That bash script we edited was called the **bashrc** file which was executed every time we logged on.

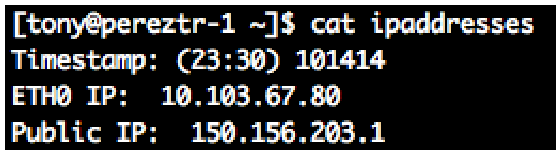
2 Introduction

For this lab we were given three tasks:

- To write a script that writes our system's public and private IP addresses to a file in our home directory called **ipaddresses**.
- Make another script that backs up the contents of our home directory to a new directory at **/var/backup/pereztr**, which I have created.
- The last script will take 10 exam grades as input and returns the average.

3 Get IP Address Script

We had to take our system's public and private IP address and store it into a file. We also had to include a timestamp using the **date** command and be in spcific format. The format was:
Timestamp: (HH:MM) MMDDYYYY
ETH0 IP: XXX.XXX.XXX.XXX
Public IP: XXX.XXX.XXX.XXX
Result:



```
[tony@pereztr-1 ~]$ cat ipaddresses
Timestamp: (23:30) 101414
ETH0 IP:  10.103.67.80
Public IP:  150.156.203.1
```

Figure 1: Using the **cat** command to display the content of the **ipaddresses** file

To do this we first used the **date** command with some flags to format the date and time as shown in Figure 1.

```
date +‘‘Timestamp: (%H:%M) %m%d%y’’ > /home/tony/ipaddresses
```

Here we used the **date** command with the + symbol to indicate we want to format the output. The **date** command makes it easy to manipulate how to display the date and time with text we want, for example the *Timestamp:* text right before the time and date. This all must be put into quotes and to tell where to put time and date we use the %H for hour,

%M for minute, %m for month, %d for date and %y for year. We put parenthesis and colon where we wish to display it. I use the >operand to overwrite the file **ipaddresses** in **/home/tony/** directory.
For the Public and Private IP address I just reused the code from the previous lab.

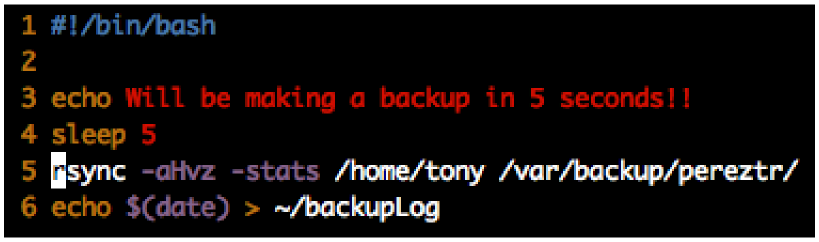
```
echo ''ETH0 IP: '' $(/sbin/ifconfig | grep -m 1 ''inet addr:'' |
awk {'print $2'} | cut -d ':'-f2) >> /home/tony/ipaddresses

echo ''Public IP: '' $(wget -qO- ipchicken.com | grep -Eo -m
''([0-9]{1,3}[\.]){3}[0-9]{1,3}'') >> /home/tony/ipaddresses
```

Instead of using a singe > operand I used two, >>, to append the command results into the **ipaddresses** file. We make sure to make the script executable using the:

```
chmod +x get_ipaddress.sh
```

4 Backup Script

For the backup script we had to back up our home directory which in our case was **/home/tony** to the directory **/var/backup/pereztr**. We used the **rsync** copy tool to do this because it provided some nice options.



```
1 #!/bin/bash
2
3 echo Will be making a backup in 5 seconds!!
4 sleep 5
5 rsync -aHvz -stats /home/tony /var/backup/pereztr/
6 echo $(date) > ~/backupLog
```

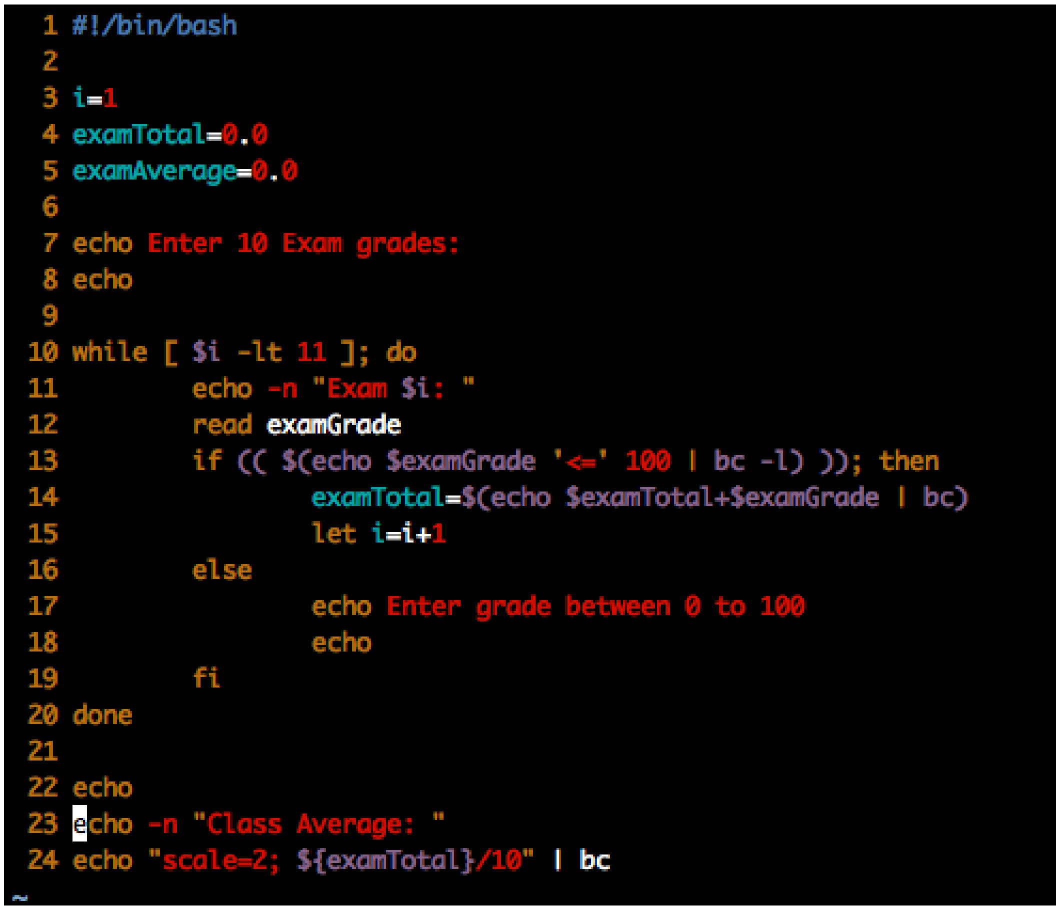
Figure 2: The **echo** command on line 6 was used to verify the scripted ran

The **rsync** command was used with flags **-aHvz**. The **-a** option is used to say we want to recursively transfer the files and also keep all the files the way they are permission and sym-links wise. **-H** means we want to keep hard-links. **-v** means verbose which just displays more information about the current copy job. **z** means we want to compress the file. **-stats** will display file transfer stats. Then we choose a source and destination.
We also made this file executable using the **chmod** command.

```
chmod +x backup.sh
```

5 Grade Average Script

To accomplish this script we had to use the bc command like before to add and divide floating point values.

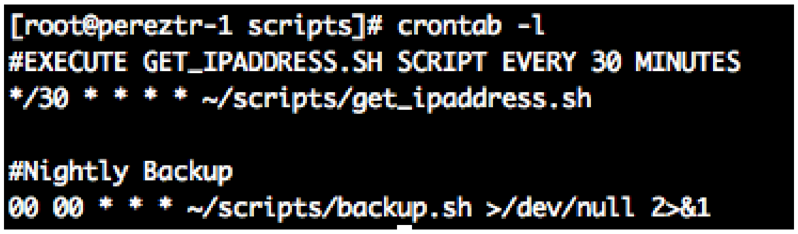


```
#!/bin/bash

i=1
examTotal=0.0
examAverage=0.0

echo Enter 10 Exam grades:
echo

while [ $i -lt 11 ]; do
	echo -n "Exam $i: "
	read examGrade
	if (( $(echo $examGrade '<=' 100 | bc -l) )); then
		examTotal=$(echo $examTotal+$examGrade | bc)
		let i=i+1
	else
		echo Enter grade between 0 to 100
		echo
	fi
done

echo
echo -n "Class Average: "
echo "scale=2; ${examTotal}/10" | bc
~
```

In this script I asked the user for ten exam grades using a while loop. I ad three variables: **i** to keep the counter, **examTotal** to keep the total of the test scores, and **examAverage** to store the average. The while loop ran 10 times each time reading in the exam grade using **read examGrade** then continuously adding it to the total. I also put in a if statement to check in case the input was over 100.

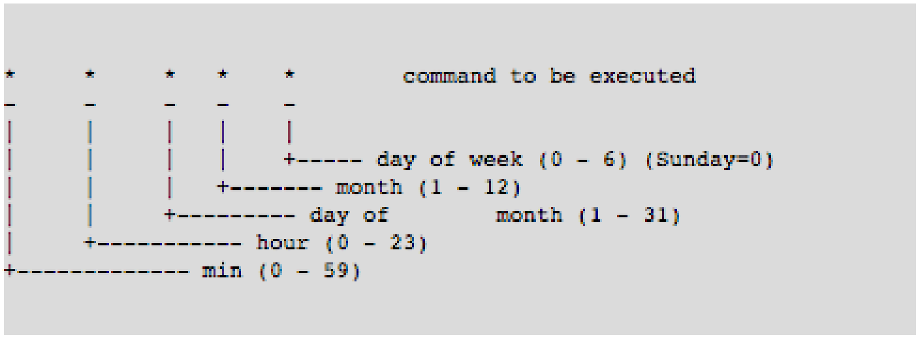
6 Cron Job

We then schedule the **get_ipaddress.sh** script to run every 30 minutes and **backup.sh** script to run every night; I chose it to run at midnight. This shows the result of the command **crontab -l** which displays the current cron jobs.



```
[root@pereztr-1 scripts]# crontab -l
#EXECUTE GET_IPADDRESS.SH SCRIPT EVERY 30 MINUTES
*/30 * * * * ~/scripts/get_ipaddress.sh

#Nightly Backup
00 00 * * * ~/scripts/backup.sh >/dev/null 2>&1
```

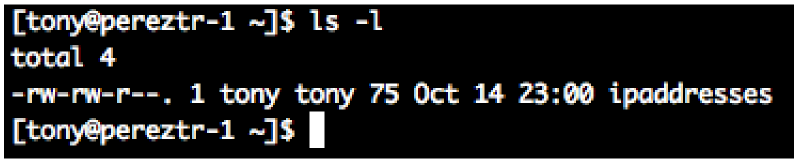
The format to write a cron job is:



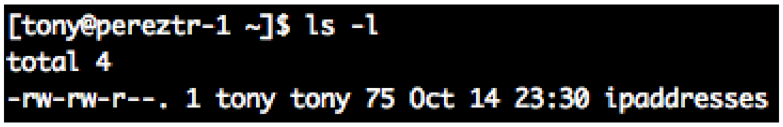
```
*     *     *   *    *        command to be executed
-     -     -   -    -
|     |     |   |    |
|     |     |   |    +----- day of week (0 - 6) (Sunday=0)
|     |     |   +------- month (1 - 12)
|     |     +--------- day of        month (1 - 31)
|     +----------- hour (0 - 23)
+------------- min (0 - 59)
```

Figure 3: Format Meaning
source by:http://www.adminschoice.com/crontab-quick-reference

Proof for IP address script:



```
[tony@pereztr-1 ~]$ ls -l
total 4
-rw-rw-r--. 1 tony tony 75 Oct 14 23:00 ipaddresses
[tony@pereztr-1 ~]$
```



```
[tony@pereztr-1 ~]$ ls -l
total 4
-rw-rw-r--. 1 tony tony 75 Oct 14 23:30 ipaddresses
```

We can see that it was edited exactly 30 minutes from each other.

7 Conclusion

In this lab we have learned how to schedule tasks that we may want to be ran automatically. This is useful because it will help automate things like backups and getting information that we might want to see everyday. Cron can be used to remind or do things if used with scripts we want.