NCS 490:
CLI Lab

By: Tony Perez

October 1 2014

1 Abstract

In this lab we learned more command-line tools to manage and manipulate files. We also learned how to use certain programs and commands that helped us get information and change how they are displayed. Many of the programs and commands we learned have been useful and seem to be very common. They make certain tasks easier when configuring your Linux box. In general we learned that there are many commands that can help us do tasks easier and in the way we want it. In the end we were assigned a task to produce a custom banner with some information about our machine and environment that must be displayed every time the user logged in.

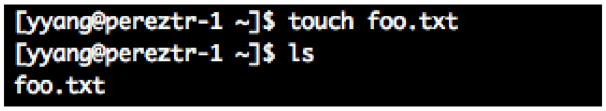
2 Introduction

In this lab we started of by making some temporary files and directories to see how certain commands worked. We used commands to display information and also chained them to get specific things from those outputs. We manipulated files and directories by creating, moving, deleting, and linking them. We used programs and utilities like: find, grep, wget, locate, awk, and sed, to help us get information the way we wanted it. We also worked with the ***.bashrc*** file that our bash, which is our command-line interpreter, reads when you login. We edited this file to show us a custom banner, welcome message, system uptime, LAN IP, WAN IP, and current temperature in a certain color depending on that temperature. We were also assigned pages 122-140 from our Linux Administration book.[1]

3 File Management and Manipulation

The **cp** command is used to copy files and to test it first we used the the **touch** command to create an empty file called **foo.txt** and I will be using the **ls** command to list the directory content:

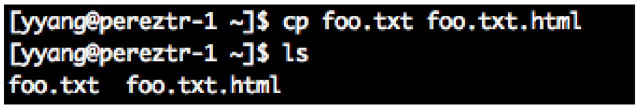
```
touch foo.txt
```



```
[yyang@pereztr-1 ~]$ touch foo.txt
[yyang@pereztr-1 ~]$ ls
foo.txt
```

Then we used the **cp** command to copy **foo.txt** to **foo.txt.html**:

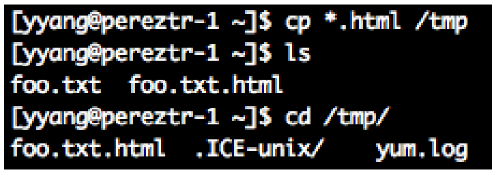
```
cp foo.txt foo.txt.html
```



```
[yyang@pereztr-1 ~]$ cp foo.txt foo.txt.html
[yyang@pereztr-1 ~]$ ls
foo.txt  foo.txt.html
```

To copy all the files ending with .html to the **/tmp** directory we used the wild card * before .html

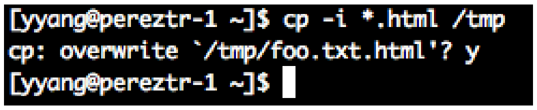
```
cp *.html /tmp
```



```
[yyang@pereztr-1 ~]$ cp *.html /tmp
[yyang@pereztr-1 ~]$ ls
foo.txt  foo.txt.html
[yyang@pereztr-1 ~]$ cd /tmp/
foo.txt.html  .ICE-unix/    yum.log
```

Then we interactively recopied the **foo.txt** to the **/tmp** directory except this time we used the interactive flag **-i** to prompt us whether to overwrite the file since we are coping the same file again.

```
cp -i *.html /tmp
```



```
[yyang@pereztr-1 ~]$ cp -i *.html /tmp
cp: overwrite `/tmp/foo.txt.html'? y
[yyang@pereztr-1 ~]$
```

The next command we test is the **mv**, move, command by moving the **foo.txt.html** file we copied; from the **tmp** directory to the current directory using the dot at the end:

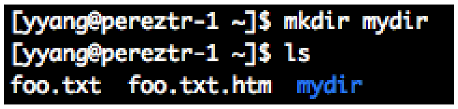
```
mv /tmp/foo.txt.html .
```

The **mv** command can also be used to rename a file like so:

```
mv /tmp/foo.txt.html foo.txt.htm
```

To make a directory we use the **mkdir** command. For a single directory we use **mkdir** with no flags:

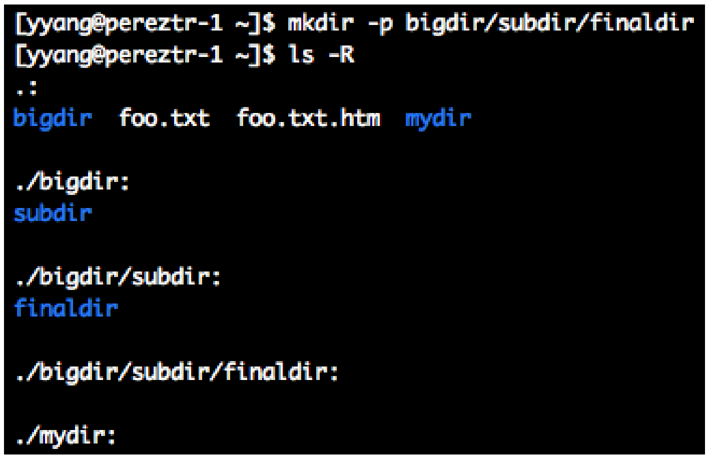
```
mkdir mydir
```



```
[yyang@pereztr-1 ~]$ mkdir mydir
[yyang@pereztr-1 ~]$ ls
foo.txt  foo.txt.htm  mydir
```

We can also use the **-p** flag to force **mkdir** to create parent directories if they don't exist already. In our case we are going to make the directories **bigdir/subdir/finaldir**.
The **ls -R** command is used to display the files and directories with their sub directories. **-R** means it will list the sub directories recursively.:

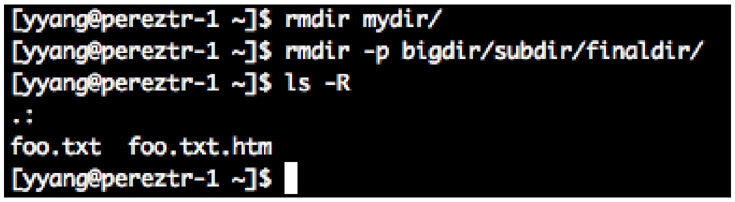
```
mkdir -p bigdir/subdir/finaldir
```



```
[yyang@pereztr-1 ~]$ mkdir -p bigdir/subdir/finaldir
[yyang@pereztr-1 ~]$ ls -R
.:
bigdir  foo.txt  foo.txt.htm  mydir

./bigdir:
subdir

./bigdir/subdir:
finaldir

./bigdir/subdir/finaldir:

./mydir:
```

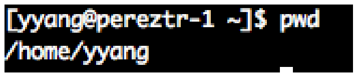
Now I will remove the directories I have made using the **rmdir** command:

```
rmdir mydir
rmdir -p bigdir/subdir/finaldir
```



```
[yyang@pereztr-1 ~]$ rmdir mydir/
[yyang@pereztr-1 ~]$ rmdir -p bigdir/subdir/finaldir/
[yyang@pereztr-1 ~]$ ls -R
.:
foo.txt  foo.txt.htm
[yyang@pereztr-1 ~]$
```

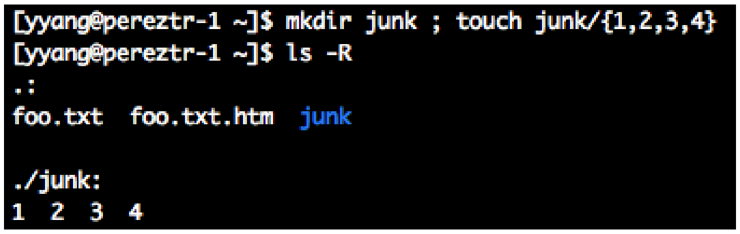
Another useful command is the Present Working Directory or **pwd**. This command lets you know where you are currently are in the file system.



```
[yyang@pereztr-1 ~]$ pwd
/home/yyang
```

The next two commands are compression tool that are used very often. One of them being the **tar** command. This command is used to combine multiple files into a single large file. We create a **junk** directory with some empty files called **1, 2, 3, 4**:

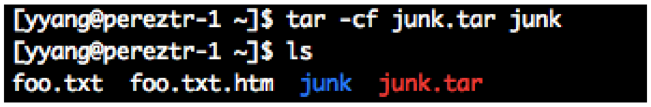
```
mkdir junk ; touch junk/{1,2,3,5}
```



```
[yyang@pereztr-1 ~]$ mkdir junk ; touch junk/{1,2,3,4}
[yyang@pereztr-1 ~]$ ls -R
.:
foo.txt  foo.txt.htm  junk

./junk:
1  2  3  4
```

We then create a archive called **junk.tar** which will contains all the files in the folder **junk**:

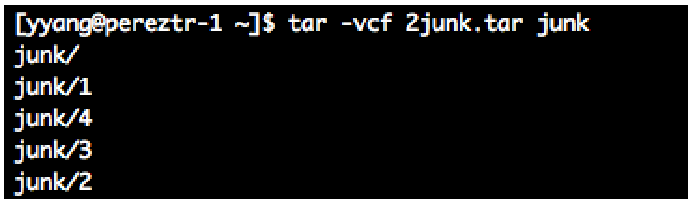
```
tar -cf junk.tar junk
```



```
[yyang@pereztr-1 ~]$ tar -cf junk.tar junk
[yyang@pereztr-1 ~]$ ls
foo.txt  foo.txt.htm  junk  junk.tar
```

Then we make another archive called **2junk.tar** again containing all the files in **junk** except that this time we will be adding the **-v** flag which means verbose and it will show what is happening:

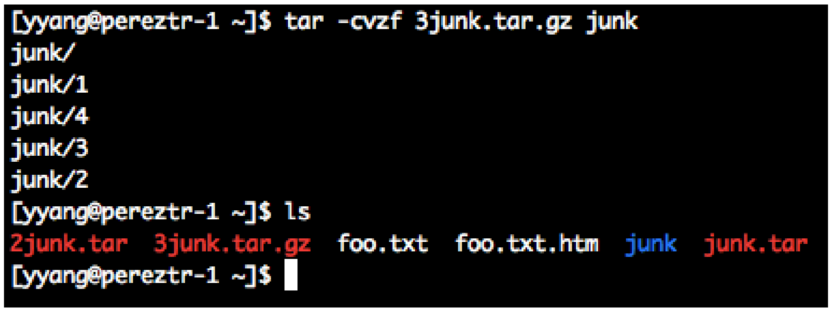
```
tar -vcf 2junk.tar junk
```



```
[yyang@pereztr-1 ~]$ tar -vcf 2junk.tar junk
junk/
junk/1
junk/4
junk/3
junk/2
```

Now that we have put multiple files into one we will compress them using the **-z** flag which filters the archive through the gzip compression utility:

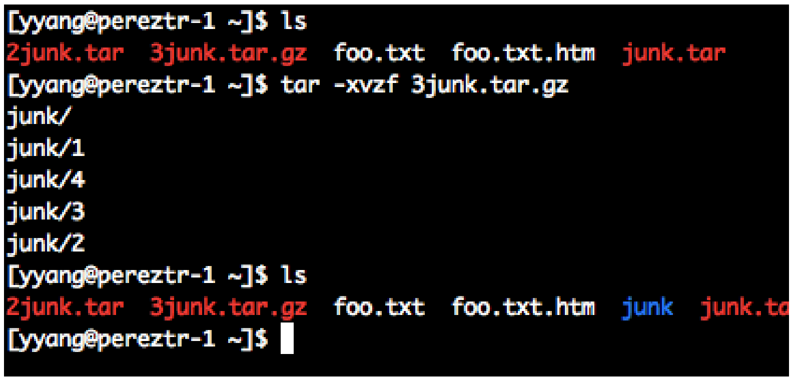
```
tar -cvzf 3junk.tar.gz junk
```



```
[yyang@pereztr-1 ~]$ tar -cvzf 3junk.tar.gz junk
junk/
junk/1
junk/4
junk/3
junk/2
[yyang@pereztr-1 ~]$ ls
2junk.tar  3junk.tar.gz  foo.txt  foo.txt.htm  junk  junk.tar
[yyang@pereztr-1 ~]$
```

Now that we have compressed the file we can extract it using the **-x** flag:

```
tar -xvzf 3junk.tar.gz
```

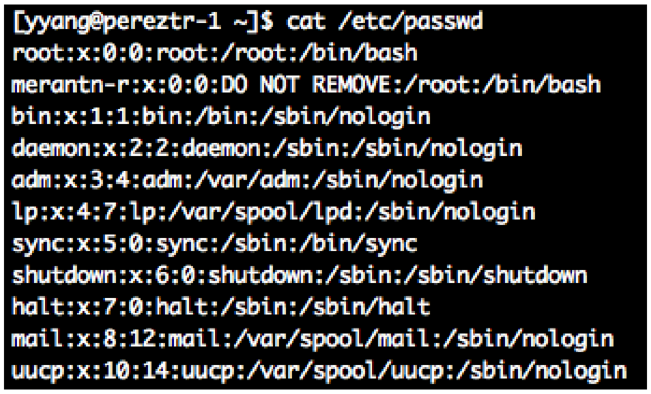


```
[yyang@pereztr-1 ~]$ ls
2junk.tar  3junk.tar.gz  foo.txt  foo.txt.htm  junk.tar
[yyang@pereztr-1 ~]$ tar -xvzf 3junk.tar.gz
junk/
junk/1
junk/4
junk/3
junk/2
[yyang@pereztr-1 ~]$ ls
2junk.tar  3junk.tar.gz  foo.txt  foo.txt.htm  junk  junk.ta
[yyang@pereztr-1 ~]$
```

*Figure 1: We can see how the directory **junk** was extracted from the compressed file **3junk.tar.gz***

The **cat** or concatenate file program will simply display files. We will use it to display the **/etc/passwd** file which contains user's authentication token:

```
cat /etc/passwd
```

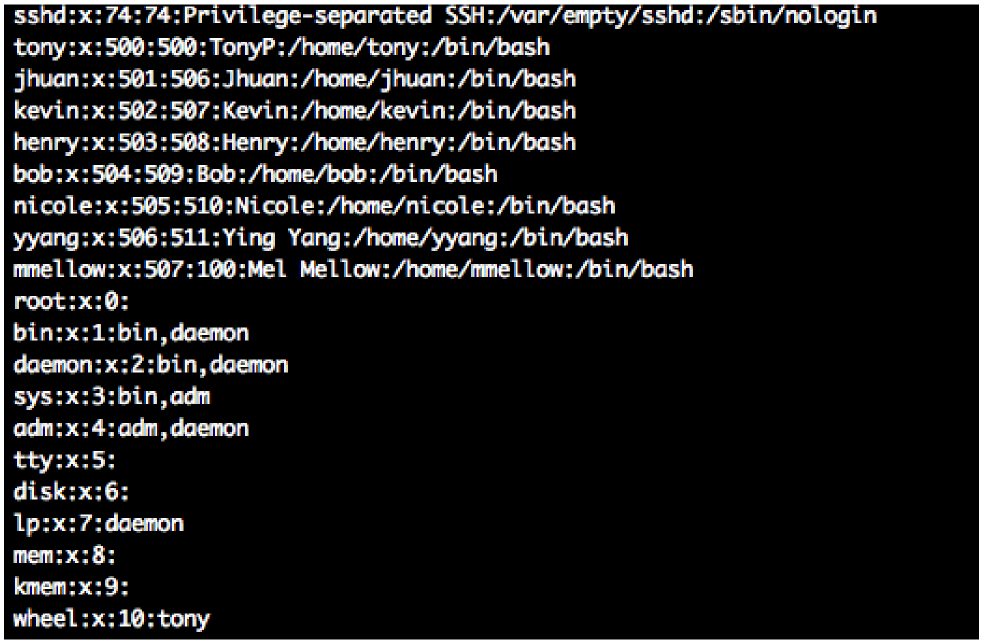


```
[yyang@pereztr-1 ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
merantn-r:x:0:0:DO NOT REMOVE:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

Figure 2: This is only part of the output to keep it short

We can also **cat** out multiple files different files:

```
cat /etc/passwd /etc/group
```

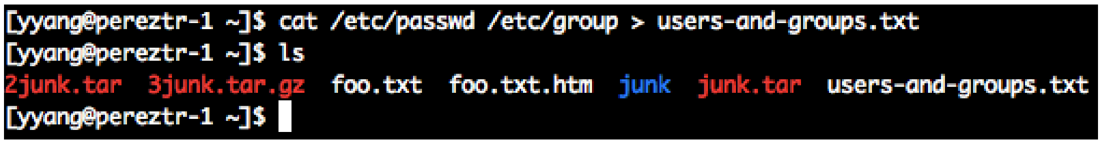


```
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tony:x:500:500:TonyP:/home/tony:/bin/bash
jhuan:x:501:506:Jhuan:/home/jhuan:/bin/bash
kevin:x:502:507:Kevin:/home/kevin:/bin/bash
henry:x:503:508:Henry:/home/henry:/bin/bash
bob:x:504:509:Bob:/home/bob:/bin/bash
nicole:x:505:510:Nicole:/home/nicole:/bin/bash
yyang:x:506:511:Ying Yang:/home/yyang:/bin/bash
mmellow:x:507:100:Mel Mellow:/home/mmellow:/bin/bash
root:x:0:
bin:x:1:bin,daemon
daemon:x:2:bin,daemon
sys:x:3:bin,adm
adm:x:4:adm,daemon
tty:x:5:
disk:x:6:
lp:x:7:daemon
mem:x:8:
kmem:x:9:
wheel:x:10:tony
```

*Figure 3: To keep it short I am showing where the **/etc/passwd** ends and the **/etc/group** starts.*

Using the > key we can send or **cat** an output to a file to save it:

```
cat /etc/passwd /etc/group > users-and-groups.txt
```

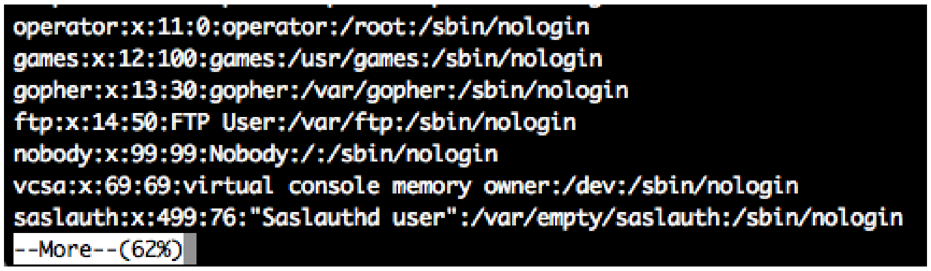


```
[yyang@pereztr-1 ~]$ cat /etc/passwd /etc/group > users-and-groups.txt
[yyang@pereztr-1 ~]$ ls
2junk.tar  3junk.tar.gz  foo.txt  foo.txt.htm  junk  junk.tar  users-and-groups.txt
[yyang@pereztr-1 ~]$ 
```

Figure 4: We can see the new text file was made

Using the **more** we can display a file one screen at a time:

```
more /etc/passwd
```

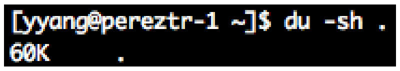


```
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin
--More--(62%)
```

Figure 5: This is based on the window size

To determine where and by whom disk space is being used we use the **du** command. It displays the disk utilization on a directory-by-directory basis:

```
du -sh .
```

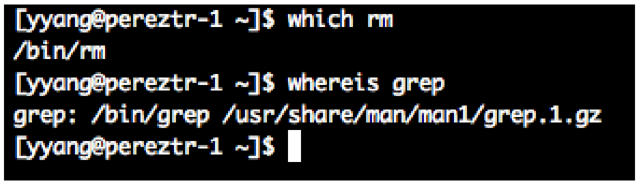


```
[yyang@pereztr-1 ~]$ du -sh .
60K	.
```

Figure 6: Shows the total memory being used in the ***pwd***

To look for the directory location of a file we can use the **which** command and to locate a command we can use the **whereis** tool:

```
which rm
whereis grep
```

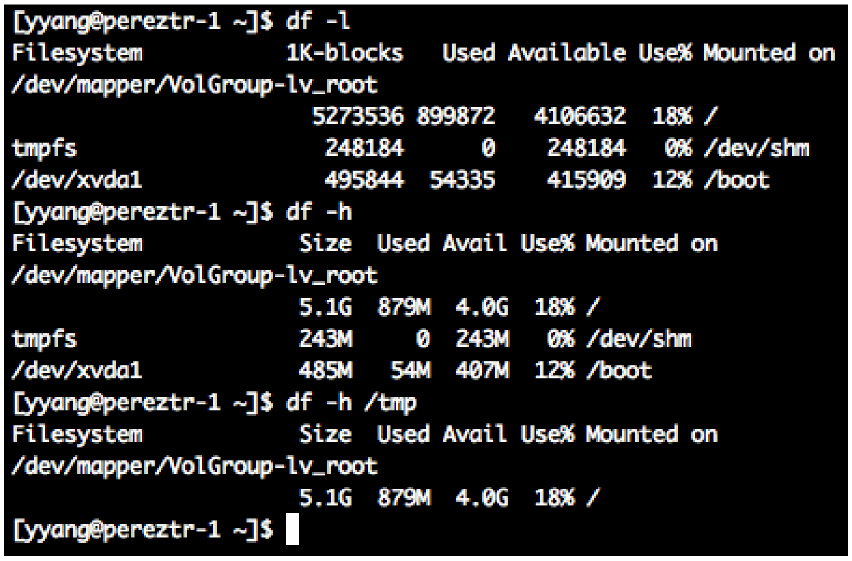


```
[yyang@pereztr-1 ~]$ which rm
/bin/rm
[yyang@pereztr-1 ~]$ whereis grep
grep: /bin/grep /usr/share/man/man1/grep.1.gz
[yyang@pereztr-1 ~]$
```

Figure 7: Here we look for the rm and grep command

The **df** program displays the amount of free space by partition. The **-h** flag displays free-space in human-readable number rather than free blocks. The **-l** lists only the locally mounted file systems. No information about network-mounted file systems will be displayed:

```
df -l
df -h
df -h /tmp
```



```
[yyang@pereztr-1 ~]$ df -l
Filesystem           1K-blocks   Used Available Use% Mounted on
/dev/mapper/VolGroup-lv_root
                       5273536 899872   4106632  18% /
tmpfs                   248184      0    248184   0% /dev/shm
/dev/xvda1              495844  54335    415909  12% /boot
[yyang@pereztr-1 ~]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
                      5.1G  879M  4.0G  18% /
tmpfs                 243M     0  243M   0% /dev/shm
/dev/xvda1            485M   54M  407M  12% /boot
[yyang@pereztr-1 ~]$ df -h /tmp
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
                      5.1G  879M  4.0G  18% /
[yyang@pereztr-1 ~]$
```

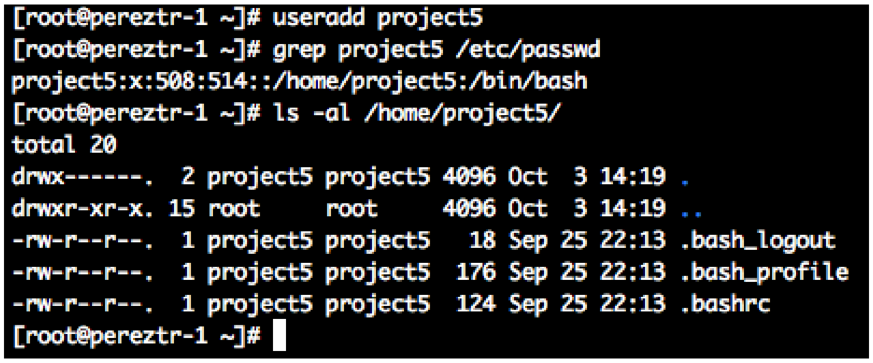
4 Moving a User and Its Home Directory

In this exercise we will be moving a user named project5 from his default home directory **/home/project5** to **/export/home/project5**. "Also we will have to set proper permissions and ownership of the user's files and directories so that the user can access it."[1]
For this we will have to work as root and add a the new user:

```
su -
useradd project5
```

We will **grep** the **/etc/passwd** file to see the new entry we have made and also use the **ls** command to display the user's home directory files:

```
grep project5 /etc/passwd
ls -al /home/project5
```



```
[root@pereztr-1 ~]# useradd project5
[root@pereztr-1 ~]# grep project5 /etc/passwd
project5:x:508:514::/home/project5:/bin/bash
[root@pereztr-1 ~]# ls -al /home/project5/
total 20
drwx------.  2 project5 project5 4096 Oct  3 14:19 .
drwxr-xr-x. 15 root     root     4096 Oct  3 14:19 ..
-rw-r--r--.  1 project5 project5   18 Sep 25 22:13 .bash_logout
-rw-r--r--.  1 project5 project5  176 Sep 25 22:13 .bash_profile
-rw-r--r--.  1 project5 project5  124 Sep 25 22:13 .bashrc
[root@pereztr-1 ~]#
```

To check the total disk space being used by the user we can use the **du** command we tested before:

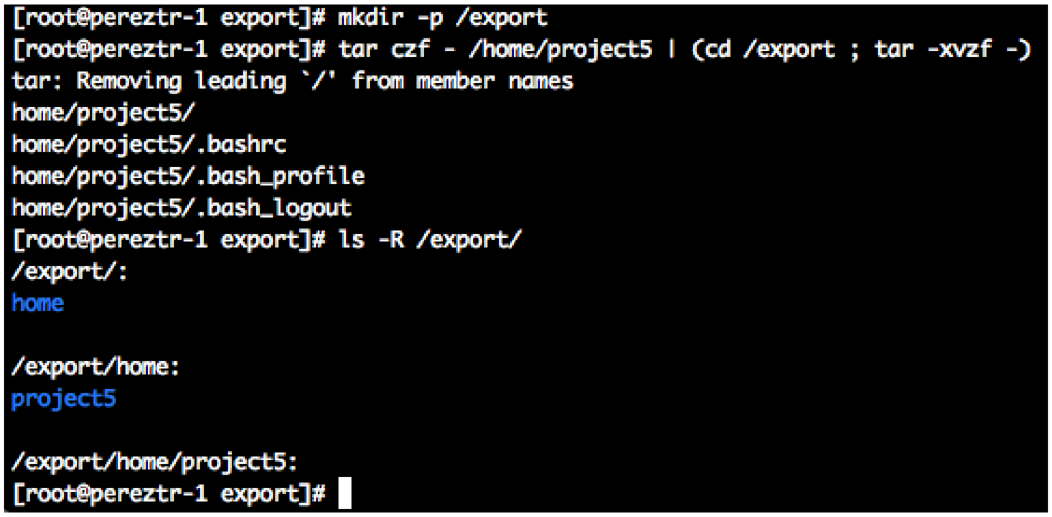
```
du -sh /home/project5
```

We will change to the project5 user and then create some empty files then exit back to root:

```
su - project5
pwd
touch a b c d e
exit
```

We will now create the **/export** directory archive and compress project5's home directory using **tar** the untar and decompress it into the new location **/export**:

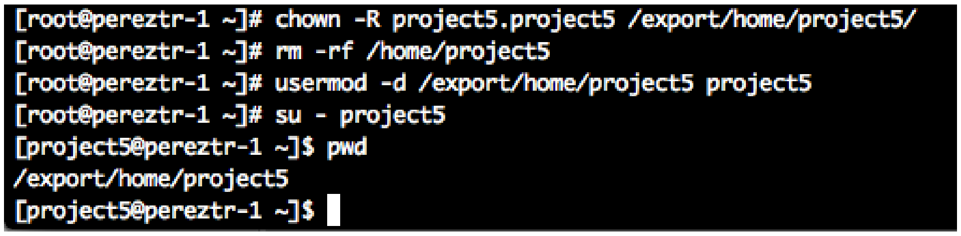
```
mkdir -p /export
tar czf - /home/project5 | (cd /export ; tar -xvzf -)
```



```
[root@pereztr-1 export]# mkdir -p /export
[root@pereztr-1 export]# tar czf - /home/project5 | (cd /export ; tar -xvzf -)
tar: Removing leading `/' from member names
home/project5/
home/project5/.bashrc
home/project5/.bash_profile
home/project5/.bash_logout
[root@pereztr-1 export]# ls -R /export/
/export/:
home

/export/home:
project5

/export/home/project5:
[root@pereztr-1 export]#
```

We then will make sure we give the project5 user complete ownership of all the files and directories in his new home. Then delete the current home directory of project5. Then we will update the /etc/passwd file with the **usermod** command then log in into project5 and do a **pwd** to see if you are in the new home directory:

```
chown -R project5.project5 /export/home/project5/
rm -rf /home/project5
usermod -d /export/home/project5 project5
```

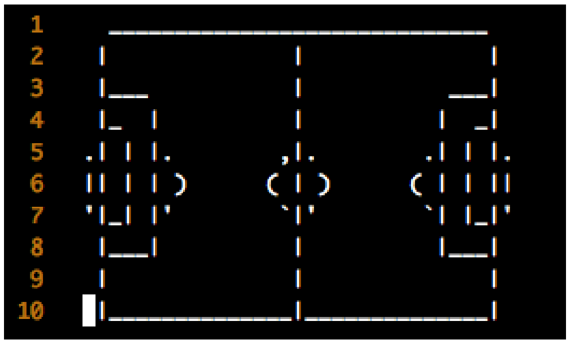


```
[root@pereztr-1 ~]# chown -R project5.project5 /export/home/project5/
[root@pereztr-1 ~]# rm -rf /home/project5
[root@pereztr-1 ~]# usermod -d /export/home/project5 project5
[root@pereztr-1 ~]# su - project5
[project5@pereztr-1 ~]$ pwd
/export/home/project5
[project5@pereztr-1 ~]$
```

5 Making a Custom Banner

We were assigned to make a custom banner that displayed an ASCII image, a welcome message with the user name and hostname, the uptime, LAP IP, WAN IP, and current weather. The weather must be printed in either red when the temperature is greater than 80, orange when it is less than 50 and greater than 50, and blue when it is less than 50. Red is for hot, orange is for warm, and blue is for cold. We used the **.bashrc** to display the information when the user logs in.

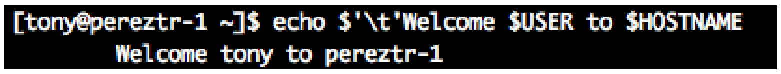
For the ASCII art I just Googled a soccer field image and put it in a text file. I then used the **cat** command to print it out.



```
 1     _____________________________
 2    |              |              |
 3    |___           |           ___|
 4    |_  |          |          |  _|
 5   .| | |.        ,|.        .| | |.
 6   || | | )      ( | )      ( | | ||
 7   '|_| |'        `|'        `| |_|'
 8    |___|          |          |___|
 9    |              |              |
10    |______________|______________|
```

Figure 8: The numbers is just the way I have my vim set up.

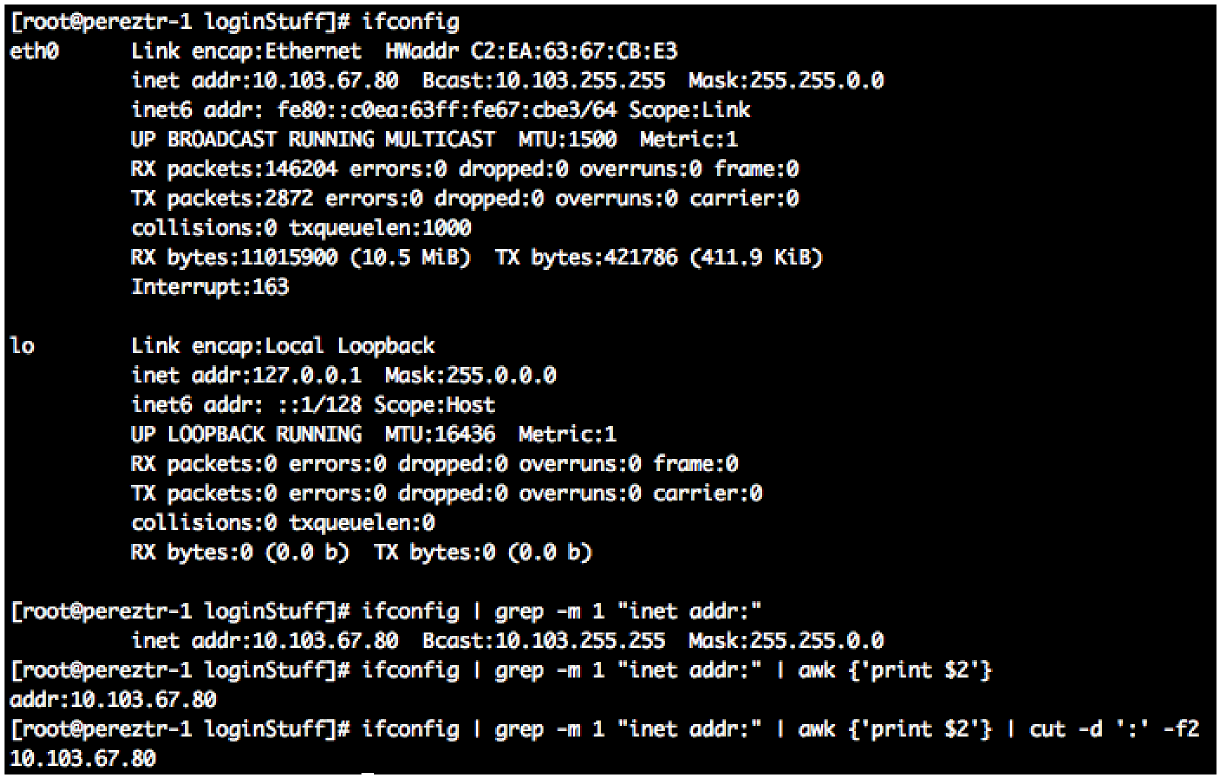
Next for the welcome banner I used **echo** with the variables **USER** and **HOSTNAME** like so:



```
[tony@pereztr-1 ~]$ echo $'\t'Welcome $USER to $HOSTNAME
        Welcome tony to pereztr-1
```

Figure 9: Used tab to center it a bit.

For the LAN IP address I took the output of the command **ifconfig** piped it to **grep** to look for inet addr because that is where the IP address is stored then piped it to the **awk** command with looked for only the second string then **cut** it to only display after the ":"

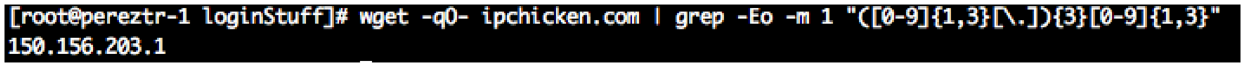


```
[root@pereztr-1 loginStuff]# ifconfig
eth0      Link encap:Ethernet  HWaddr C2:EA:63:67:CB:E3
          inet addr:10.103.67.80  Bcast:10.103.255.255  Mask:255.255.0.0
          inet6 addr: fe80::c0ea:63ff:fe67:cbe3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:146204 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2872 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11015900 (10.5 MiB)  TX bytes:421786 (411.9 KiB)
          Interrupt:163

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@pereztr-1 loginStuff]# ifconfig | grep -m 1 "inet addr:"
          inet addr:10.103.67.80  Bcast:10.103.255.255  Mask:255.255.0.0
[root@pereztr-1 loginStuff]# ifconfig | grep -m 1 "inet addr:" | awk {'print $2'}
addr:10.103.67.80
[root@pereztr-1 loginStuff]# ifconfig | grep -m 1 "inet addr:" | awk {'print $2'} | cut -d ':' -f2
10.103.67.80
```

For the WAN I had to use the **wget** command to get the a webpage that contained my WAN IP. Wget is used to get content from a site and outputs it. I used the site ipchicken to get my WAN IP.

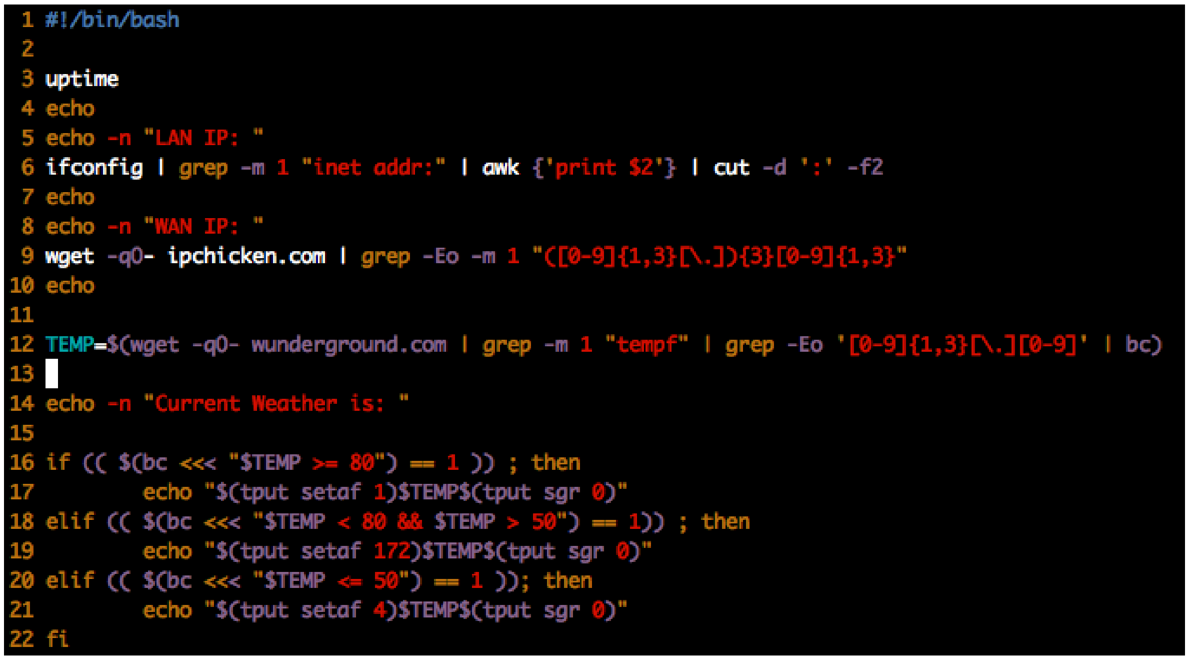
```
wget -qO- ipchicken.com | grep -Eo -m 1 "([0-9]{1,3}[\.]){3}[0-9]{1,3}"
```

I used **grep** with flags -E, this is to use regular expressions, -o to only find the exact match and -m 1 to stop after it has found the first match. The regular expression used was from `http://www.shellhacks.com/en/RegEx-Find-IP-Addresses-in-a-File-Using-Grep`



```
[root@pereztr-1 loginStuff]# wget -qO- ipchicken.com | grep -Eo -m 1 "([0-9]{1,3}[\.]){3}[0-9]{1,3}"
150.156.203.1
```

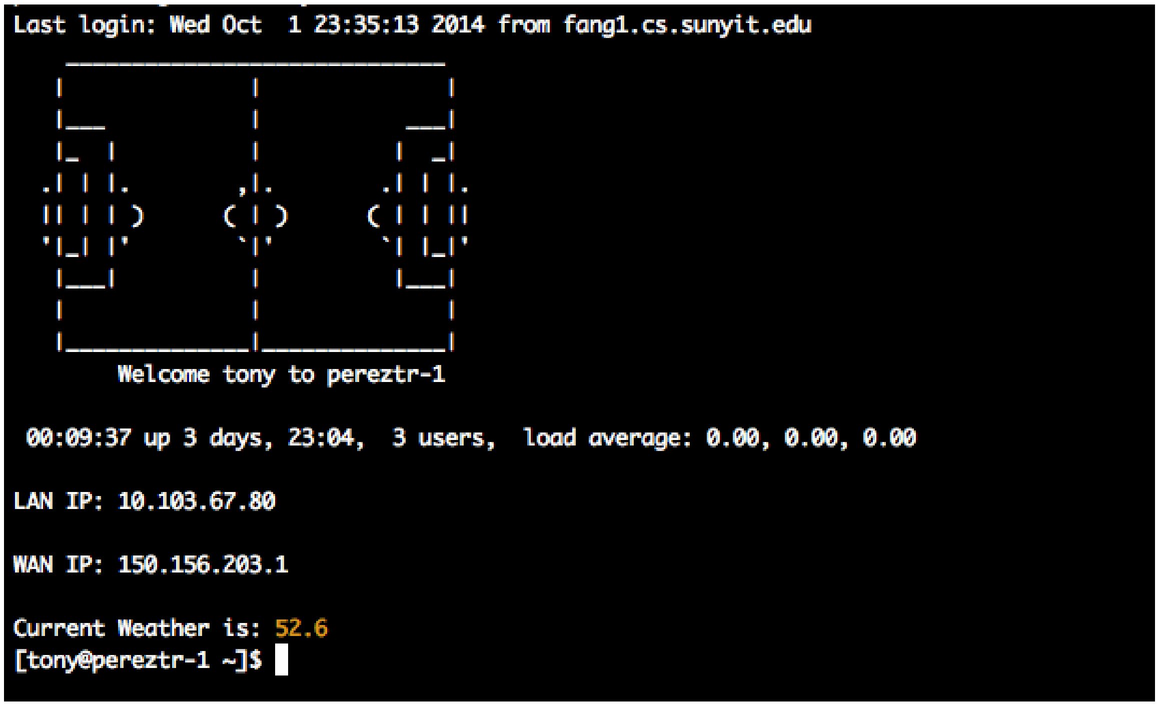
For the current weather I used the site `wuderground.com` and did almost the same steps to get the WAN IP. I used **grep** to find the word *temf* because that will give me the temperatures for the first day and then grep with regular expression to find a number from three digit number max deciaml then and two digit number with the flag -m 1 to only give me the first one I then put the output to **bc** to compare it in an if else statement to see which color it should be based on the tempurature.



```
#!/bin/bash

uptime
echo
echo -n "LAN IP: "
ifconfig | grep -m 1 "inet addr:" | awk {'print $2'} | cut -d ':' -f2
echo
echo -n "WAN IP: "
wget -qO- ipchicken.com | grep -Eo -m 1 "([0-9]{1,3}[\.]){3}[0-9]{1,3}"
echo

TEMP=$(wget -qO- wunderground.com | grep -m 1 "tempf" | grep -Eo '[0-9]{1,3}[\.][0-9]' | bc)

echo -n "Current Weather is: "

if (( $(bc <<< "$TEMP >= 80") == 1 )) ; then
        echo "$(tput setaf 1)$TEMP$(tput sgr 0)"
elif (( $(bc <<< "$TEMP < 80 && $TEMP > 50") == 1)) ; then
        echo "$(tput setaf 172)$TEMP$(tput sgr 0)"
elif (( $(bc <<< "$TEMP <= 50") == 1 )); then
        echo "$(tput setaf 4)$TEMP$(tput sgr 0)"
fi
```

6 Conclusion

I learned how to get information the way I wanted it and so it makes it easier to look for things. I also learned how to better use the CLI and many of these commands I will using on a daily basis.
This was my final product:



```
Last login: Wed Oct  1 23:35:13 2014 from fang1.cs.sunyit.edu
    _____________________________
   |              |              |
   |___           |           ___|
   |_  |          |          |  _|
  .| | |.        ,|.        .| | |.
  || | | )     ( | )     ( | | ||
  '|_| |'        `|'        `| |_|'
   |___|          |          |___|
   |              |              |
   |______________|______________|
        Welcome tony to pereztr-1

 00:09:37 up 3 days, 23:04,  3 users,  load average: 0.00, 0.00, 0.00

LAN IP: 10.103.67.80

WAN IP: 150.156.203.1

Current Weather is: 52.6
[tony@pereztr-1 ~]$
```

References

[1] Wale Soyinka. *Linux Administration: A Beginners Guide*. McGraw-Hill Osborne Media, 6 edition, 2012.