Linux Logbook Portfolio 2

Tatenda Manyepa | 2020

Table of Contents

1. Linux Tutorials Basic Linux

1.1. Making Directories

A new directory called backups was created inside the directory unixstuff. The *cd unixstuff* command was used to navigate into the unixstuff directory and the command *mkdir backups* was then used to create the backups directory.

```
root@Bry021:~# cd unixstuff
root@Bry021:~/unixstuff# mkdir backups
root@Bry021:~/unixstuff# ls
backups/  new_file
root@Bry021:~/unixstuff#
```

1.2. ls, pwd and cd Commands

The *ls* command was used to list all the files in the current working directory which in this case is the home directory. However, the files listed did not include hidden files therefore the *ls -a* command was then executed to show all files including the hidden files indicated by (.) at the beginning of file names. *cd unixstuff* was then executed to navigate to the unixstuff directory and the command *cd* was also used to navigate back to the home directory. The *pwd* command was then executed to print the pathname of the current working directory which is /root.

```
root@Bry021:~# ls
Desktop/    Downloads/  Pictures/  Templates/  internal  new_file  unistuff/
Documents/  Music/      Public/    Videos/     ncs/      test.sh*  unixstuff/
root@Bry021:~# ls-a
-su: ls-a: command not found
root@Bry021:~# ls -a
./             .cache/    .xinitrc    Pictures/   ncs/
../            .config/   Desktop/    Public/     new_file
.ICEauthority  .dbus/     Documents/  Templates/  test.sh*
.Xauthority    .kde/      Downloads/  Videos/     unistuff/
.bash_history  .local/    Music/      internal    unixstuff/
root@Bry021:~# cd unixstuff
root@Bry021:~/unixstuff# cd
root@Bry021:~# pwd
/root
root@Bry021:~#
```

The command *cd /etc* was executed to navigate into the /etc directory which is the central location of where all Linux configuration files are located (Surendra, 2016). The image below shows the contents of the /etc directory.

```
gamin/                    my-medium.cnf          syslog.conf
genpowerd.conf            my-small.cnf           sysstat/
gettydefs                 nail.rc                termcap
gimp/                     named.conf             termcap-BSD
gnupg/                    netatalk/              termcap-Linux
gpm-root.conf             netgroup               tin/
gpm-syn.conf              networks               udev/
gpm-twiddler.conf         nfsmount.conf          updatedb.conf
group                     nntpserver             usb_modeswitch.conf
group-                    nscd.conf              usb_modeswitch.d/
gshadow                   nsswitch.conf          uucp/
gshadow-                  nsswitch.conf-nis      vga/
gtk/                      ntp/                   virtuoso.ini
gtk-2.0/                  ntp.conf               vsftpd.conf
hal/                      obex-data-server/      warnquota.conf-sample
hardwareclock             openldap/              wgetrc
host.conf                 openvpn/               wpa_supplicant.conf
hosts                     organization           xdg/
hosts.allow               pango/                 xfce/
hosts.deny                passwd                 xml/
hosts.equiv               passwd-                xpdfrc
hp/                       pcmcia/                yp.conf
htdig/                    pear.conf              ytalkrc
httpd/                    php/                   zprofile@
root@Bry021:/etc#
```

The *cd ntp* command was used inside the /etc directory to view the ntp protocol. The ntp protocol is utilized in synchronizing the computer system clock automatically over networks (Cezar, 2018). The *cd* command was then executed to navigate back to the home directory root as shown in the image below.

```
root@Bry021:/etc# cd ntp
root@Bry021:/etc/ntp# ls
ntp.keys  step-tickers
root@Bry021:/etc/ntp# cd
root@Bry021:~#
```

1.3. ls ~ and ls ~/.. Commands

The ls ~ command lists the contents of the home directory which in this case is root while the ls ~/.. command lists all files in the parent directory under the home directory. This is illustrated in the image below (Saive, 2012).

```
root@Bry021:~# ls ~
Desktop/    Downloads/  Pictures/  Templates/  internal  new_file  unistuff/
Documents/  Music/      Public/    Videos/     ncs/      test.sh*  unixstuff/
root@Bry021:~# ls ~/..
bin/   dev/  home/  lost+found/  mnt/  proc/  sbin/  sys/  usr/
boot/  etc/  lib/   media/       opt/  root/  srv/   tmp/  var/
root@Bry021:~#
```

1.4. Relative and Absolute Pathnames

The *cd* command was executed to navigate to the home directory and *cd unixstuff* was used to navigate to the unixstuff directory. To find the absolute pathname of the unixstuff directory the *pwd* command was executed and the result obtained was that /root/unixstuff is the absolute pathname for the unixstuff directory. From this result we can deduce that the relative pathname of the unixstuff directory relative to the home directory is unixstuff/.

```
root@Bry021:~# cd
root@Bry021:~# pwd unixstuff
/root
root@Bry021:~# cd unixstuff
root@Bry021:~/unixstuff# pwd
/root/unixstuff
root@Bry021:~/unixstuff# _
```

1.5. Creating Files in VI Text Editor

The command *vi new_file* was executed to create a new text editor file and text was added to the in the insert mode as shown below. The file was then saved as new_file1 in the command mode and the command *:q* was used to exit the vi text editor.

```
A controversial theory that the way we smell involves a quantum physics effect h
as recieved a boost, following experiments with human subjects.  It challenges t
he notion that our sense of smell depends only on the shapes of molecules we sni
ff in the air.  Instead, it suggests that the molecules' vibrations are responsi
ble.  A way to test it is with two molecules of the same shape, but with differe
nt vibrations.  A report in plos one shows that humans can distinguish the two.
 Tantalisingly, the idea hints at quantum effects occuring in biological systems
-an idea that is itself driving a new field of science, as the BBC feature artic
le Are birds hijaking quantum physics? points out.  But the theory-first put for
ward by Luca Turin, now of the Fleming Biomedical Research Sciences Centre in Gr
eece-remains contestant and devisive.  the idea that molecules' shapes are the o
nly link to their smell is well entranched, but Dr Turin said there were holes i
n the idea.  He gave an example of molec[ules that include sulphar and hydrogen
atoms bonded together-they may take a wide range of shapes, but all of them smel
l of rotten eggs.

"If you look from the [traditional] standpoint...it is really hard to explain,"
Dr Turin told BBC news.

"If you look from the standpoint of an alternative theory-that what determines t
he smell of a molecule is the vibrations-the sulphar-hydrogen mystery becomes ab
solute clear."

Molecules can be viewed as a collection of atoms on springs, so the atoms can  m
wrote new_file1, 7 lines, 1706 chars                          7,1   Command
```

As shown in the image below the command vi new_file1 was used to open the file in the vi text editor.

```
"If you look from the [traditional] standpoint...it is really hard to explain,"
Dr Turin told BBC news.

"If you look from the standpoint of an alternative theory-that what determines t
he smell of a molecule is the vibrations-the sulphar-hydrogen mystery becomes ab
solute clear."

Molecules can be viewed as a collection of atoms on springs, so the atoms can  m

root@Bry021:~# vi newfile1
```

The following image shows that the vi new file1 still exists and is the result of using the command vi new_file1.

```
A controversial theory that the way we smell involves a quantum physics effect h
as recieved a boost, following experiments with human subjects.  It challenges t
he notion that our sense of smell depends only on the shapes of molecules we sni
ff in the air.  Instead, it suggests that the molecules' vibrations are responsi
ble.  A way to test it is with two molecules of the same shape, but with differe
nt vibrations.  A report in plos one shows that humans can distinguish the two.
 Tantalisingly, the idea hints at quantum effects occuring in biological systems
-an idea that is itself driving a new field of science, as the BBC feature artic
le Are birds hijaking quantum physics? points out.  But the theory-first put for
ward by Luca Turin, now of the Fleming Biomedical Research Sciences Centre in Gr
eece-remains contestant and devisive.  the idea that molecules' shapes are the o
nly link to their smell is well entranched, but Dr Turin said there were holes i
n the idea.  He gave an example of molec[ules that include sulphar and hydrogen
atoms bonded together-they may take a wide range of shapes, but all of them smel
l of rotten eggs.

"If you look from the [traditional] standpoint...it is really hard to explain,"
Dr Turin told BBC news.

"If you look from the standpoint of an alternative theory-that what determines t
he smell of a molecule is the vibrations-the sulphar-hydrogen mystery becomes ab
solute clear."

Molecules can be viewed as a collection of atoms on springs, so the atoms can  m
Read new_file1, 7 lines, 1706 chars                              1,1   Command
```

1.6. Changing File Mode: chmod

The *cd unixstuff* command was used to navigate into the unixstuff directory. The *ls -l* command was then used to view the file permissions of the backups and new_file sub directories in the unixstuff directory. The backups directory had read, write and execute permissions for the user, execute and read permissions for the group and only execute permissions for others on the server while new_file had read and write permissions for the user and only read permissions for both the group and others. The command *chmod go-x backups* was used to add execute permissions to the group and others while the command *chmod a+x new_file* was used to give execute permissions on the new_file to all people on the server. The command *ls -l* was then again used to check if the file permissions had been changed as illustrated in the image below.

```
root@Bry021:~# cd unixstuff
root@Bry021:~/unixstuff# ls -l
total 8
drwxr-xr-x 2 root root 4096 Nov  1 23:07 backups/
-rw-r--r-- 1 root root   39 Oct 31 15:40 new_file
root@Bry021:~/unixstuff# chmod go-x backups
root@Bry021:~/unixstuff# chmod a+x new_file
root@Bry021:~/unixstuff# ls -l
total 8
drwxr--r-- 2 root root 4096 Nov  1 23:07 backups/
-rwxr-xr-x 1 root root   39 Oct 31 15:40 new_file*
root@Bry021:~/unixstuff#
```

2.1. Further File Handling

In the unixstuff directory the command *ls* was executed to view the contents of the directory. The result showed that a file new_file exists in the backups directory. A copy of the file new_file was created by using the command *cp new_file new file.bak* and *ls* was then again executed to see if the backup folder had been created.

```
root@Bry021:~/unixstuff# ls
backups/  new_file*
root@Bry021:~/unixstuff# cp new_file new_file.bak
root@Bry021:~/unixstuff# ls
backups/  new_file*  new_file.bak*
```

2.2. Removing Files

A new directory tempstuff was created in the unixstuff directory by using the command *mkdir tempstuff* and the command *ls* was executed to see if the tempstuff directory had been created. The command *rm -r tempstuff* was then used to remove the tempstuff directory and the command *ls* was again executed to check if the removal had been successful.

```
root@Bry021:~/unixstuff# mkdir tempstuff
root@Bry021:~/unixstuff# ls
backups/  new_file*  new_file.bak*  tempstuff/
root@Bry021:~/unixstuff# rm tempstuff
rm: cannot remove 'tempstuff': Is a directory
root@Bry021:~/unixstuff# rm -r tempstuff
root@Bry021:~/unixstuff# ls
backups/  new_file*  new_file.bak*
root@Bry021:~/unixstuff# _
```

2.3. Displaying File Contents

The command *head -5 new_file1* was executed. The head command allows for the viewing of the first 10 lines of your file and adding the -5 option should ideally only allow for the viewing of the first 5 lines of any file.

```
root@Bry021:~# head -5 new_file1
A controversial theory that the way we smell involves a quantum physics effect h
as recieved a boost, following experiments with human subjects.  It challenges t
he notion that our sense of smell depends only on the shapes of molecules we sni
ff in the air.  Instead, it suggests that the molecules' vibrations are responsi
ble.  A way to test it is with two molecules of the same shape, but with differe
nt vibrations.  A report in plos one shows that humans can distinguish the two.
 Tantalisingly, the idea hints at quantum effects occuring in biological systems
-an idea that is itself driving a new field of science, as the BBC feature artic
le Are birds hijaking quantum physics? points out.  But the theory-first put for
ward by Luca Turin, now of the Fleming Biomedical Research Sciences Centre in Gr
eece-remains contestant and devisive.  the idea that molecules' shapes are the o
nly link to their smell is well entranched, but Dr Turin said there were holes i
n the idea.  He gave an example of molecIules that include sulphar and hydrogen
atoms bonded together-they may take a wide range of shapes, but all of them smel
l of rotten eggs.

"If you look from the [traditional] standpoint...it is really hard to explain,"
Dr Turin told BBC news.

"If you look from the standpoint of an alternative theory-that what determines t
he smell of a molecule is the vibrations-the sulphar-hydrogen mystery becomes ab
solute clear."
root@Bry021:~#
```

In order to view the last 15 lines of the file the command *tail -15 new_file1* can the used and the results are shown below.

```
n the idea.  He gave an example of molec[ules that include sulphar and hydrogen
atoms bonded together-they may take a wide range of shapes, but all of them smel
l of rotten eggs.

"If you look from the [traditional] standpoint...it is really hard to explain,"
Dr Turin told BBC news.

"If you look from the standpoint of an alternative theory-that what determines t
he smell of a molecule is the vibrations-the sulphar-hydrogen mystery becomes ab
solute clear."

Molecules can be viewed as a collection of atoms on springs, so the atoms can  m
ove relative to one another.  Energy of just the right frequency-a quantum-can c
ause the springs to vibrate, and in 1996 paper in Chemical Senses Dr Turin said
it was these vibrations that explained smell.
root@Bry021:~# _
```

2.4. Redirection

A file called list2 was created using the command *cat > list2* and four names of fruit were added to that file. In order to read the contents of the file list2 the command *cat list2* was executed as shown below.

```
root@Bry021:~/unixstuff# cat > list2
orange
plum
mango
grapefruit
^D
root@Bry021:~/unixstuff# cat list2
orange
plum
mango
grapefruit
^D
root@Bry021:~/unixstuff#
```

2.5. Wildcards, filename Conventions and Getting Help

The command *apropos copy* was executed, and the results are shown in the screen below. To scroll through the whole file the command *less* can be used because the file contents are larger than the screen and therefore only the bottom contents are being displayed.

```
riod
vfs_shadow_copy2 []  (8)  - Expose snapshots to Windows clients as shadow copies
vga_bitblt []        (3)  - copy pixmap on screen using an accelerator
vga_copytoplanar16 [] (3)  - copy linear pixmap into VGA 16 color mode video mem
ory
vga_copytoplanar256 [] (3)  - copy linear pixmap into Mode X video memory
vga_copytoplane []   (3)  - copy linear pixmap to some planes of VGA 16 color mo
de video memory
vga_imageblt []      (3)  - copy a rectangular pixmap from system memory to vide
o memory
wcpcpy []            (3)  - copy a wide-character string, returning a pointer to
 its end
wcpncpy []           (3)  - copy a fixed-size string of wide characters, returni
ng a pointer to its end
wcscpy []            (3)  - copy a wide-character string
wcsncpy []           (3)  - copy a fixed-size string of wide characters
wmemcpy []           (3)  - copy an array of wide-characters
wmemmove []          (3)  - copy an array of wide-characters
wmempcpy []          (3)  - copy memory area
wxcopy []            (1)  - copy stdin or file into cutbuffer
xdfcopy []           (1)  - Program to copy and format Xdf disks in Linux
xfs_copy []          (8)  - copy the contents of an XFS filesystem
xfs_metadump []      (8)  - copy XFS filesystem metadata to a file
xfs_rtcp []          (8)  - XFS realtime copy command
root@Bry021:~/unixstuff# _
```

The results were then saved into a text file called copy_functions.txt by executing the command *apropos >> copy_function.txt*. To view the amount of different functions available the command *wc -l copy_function.txt* was executed and the results below show that there are 136 functions available.

```
root@Bry021:~/unixstuff# apropos copy >> copy_function.txt
root@Bry021:~/unixstuff# wc -1 copy_function.txt
wc: invalid option -- '1'
Try 'wc --help' for more information.
root@Bry021:~/unixstuff# wc -l copy_function.txt
136 copy_function.txt
root@Bry021:~/unixstuff#
```

2. Linux Systems and Shell Scripting

2.1 Creating an executable bash script

A script called test.sh was created in the home directory using a vi text editor. The command *vi new_file* was executed to create this script. The following image shows the contents of the script in the text editor and the script ends with the extension .sh implying that this is a bash shell script (EDUCBA, 2019). Other console text editors such as nano and emacs could have been used to create the script as they also contain high functionality and are simple to use and easy to navigate in (Blum and Bresnahan, 2015).

```
#!/bin/bash
clear
echo "Hello World"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
writing test.sh                                    1,1   Command
```

The command *ls* was executed to find out if the test.sh script has been saved in the home directory and the results in the image below show that the script now exists in the directory. The symbols ~# suggests that a user is currently operating in the home directory which in this case is root (Soyinka, 2016).To set execute permissions that allow the script to be executable, the command *chmod a+x test.sh* was executed and the script was run by executing the command *./test.sh* as shown in the image below

```
root@Bry021:~# ls
Desktop/    Music/     Templates/  ncs/       test.sh*
Documents/  Pictures/  Videos/     new_file   unistuff/
Downloads/  Public/    internal    new_file1  unixstuff/
root@Bry021:~# chmod a+x test.sh
root@Bry021:~# ls -l
total 56
drwxr-xr-x 2 root root 4096 Mar  2  2012 Desktop/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Documents/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Downloads/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Music/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Pictures/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Public/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Templates/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Videos/
-rw-r--r-- 1 root root    0 Nov  1 20:49 internal
drwxrwxrwx 2 root root 4096 Oct 31 20:41 ncs/
-rw-r--r-- 1 root root   39 Oct 31 19:23 new_file
-rw-rw-rw- 1 root root 1706 Nov  2 02:44 new_file1
-rwxr-xr-x 1 root root   37 Nov  1 07:28 test.sh*
drwxr-xr-x 2 root root 4096 Oct 31 13:50 unistuff/
drwxr-xr-x 3 root root 4096 Nov  2 06:12 unixstuff/
root@Bry021:~#
```

The image below shows the result of running the shell script test.sh. The message “Hello World” was successfully printed on the console screen.

```
Hello World
root@Bry021:~# ls test.sh
test.sh*
root@Bry021:~#
```

2.2. Why giving 777 permissions to a file is a bad idea

777 file permissions indicate that any individual who is a user on a server can read, write and execute any files on that server (Coulter, 2014). This is risky because it gives files and directories low security since the files and directories on the server can altered and/or deleted without the administrator's permission or knowledge (McKinnon, 2017). These files can also be tempered with by having corrupt files added onto them (McDonald, 2012). 777 permissions can also lead to attacks such as cross-site scripting attacks which occur when malicious html and/or JavaScript are copied into an open folder on a server leading to a malicious user stealing the administrator login cookie thereby allowing them to gain full access to the administrator's account (Coulter, 2014). The normal safe permissions for files should be 644 which means that the administrator or user can only have read and write permissions while everyone else will only be able to read the files or scripts (McKinnon, 2017).

2.3. Creating new users

A user account was created for a new user bob using the command *useradd -d/home/bob bob* and a new password bob was assigned to the new user bob. This is shown in the image below:

```
root@Bry021:~# useradd -d/home/bob bob
root@Bry021:~# passwd bob
Changing password for bob
Enter the new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
Re-enter new password:
They don't match; try again.
New password:
Bad password: too short.
Warning: weak password (enter it again to use it anyway).
New password:
Re-enter new password:
passwd: password changed.
```

Another user account for a new user smith was also created using the command *useradd -d/home/smith* and the user was assigned a password smith when prompted by the console as shown below.

```
root@Bry021:~# useradd -d/home/smith smith
root@Bry021:~# passwd smith
Changing password for smith
Enter the new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
Bad password: too simple.
Warning: weak password (enter it again to use it anyway).
New password:
Re-enter new password:
passwd: password changed.
```

There are several other ways by which new users can be added in Linux. The useradd part of the command used to create new users bob and smith can be substituted by the term adduser generating the command *adduser -d/home/newusername newusername*. In Linux, every user has their own specific UID (Unique Identification Number) which is assigned to them when their new account is created, therefore, for example, the command *useradd -u 888 newusername* can be executed to create a new user account assigning them the UID of 888. The option for using a UID to create a new user is -u. In the same way a new user can be created using a UID, a new user can also be created using a specific GID (Group Identification Number) with the -g option generating the command *useradd -u 200 -g 300 newusername*. The figure 100 being the chosen IUD and 300 being the chosen GID. New users can also be created without being assigned to a home directory for security reasons using the option -M create the following command *useradd -M newusername* and a new user can also be given an expiry date with the option -e using the following command, *useradd -e 2019-08-24 newusername*. This means that the account will expire on 24-08-2019 (Saive, 2019).

2.4. Creating a shared executable script

A publicly readable and writeable directory with the path /home/ncs was created using the command *mkdir /home/ncs*. The command *ls -ld /home/ncs* was then executed to view the permissions of the directory. The option -l for the command ls allows for the listing of long formats and the showing of their permissions while the option -d allows for the listing of directories starting with / (RapidTables, 2019). The results in the image below show that the directory has readable, writable and executable permissions for the main user, executable and readable permissions for the group on the server and only executable permissions for all others on the server. To give the directory readable and writable permissions the command *chmod a+rw /home/ncs* was used. As shown below, the directory's permissions changed to all users on the server having read, write and execute permissions.

Permissions can also be set in Linux using numeric codes, 0 means no permission, 1 means execute, 2 means write, and 4 means read. For instance, to create the ncs directory and make it publicly readable and writable *chmod 666* could have been used instead. The numbers are added up depending on the permissions being given (Rafacz, 2019).

```
root@Bry021:~# mkdir /home/ncs
root@Bry021:~# ls -ld /home/
drwxr-xr-x 4 root root 4096 Nov  2 07:44 /home//
root@Bry021:~# ls -ld /home/ncs
drwxr-xr-x 2 root root 4096 Nov  2 07:44 /home/ncs/
root@Bry021:~# chmod a+rw /home/ncs
root@Bry021:~# ls -ld /home/ncs
drwxrwxrwx 2 root root 4096 Nov  2 07:44 /home/ncs/
root@Bry021:~#
```

2.5. Creating a Bash Script

A bash script called hello.sh was created in the vi text editor using the command vi new_file in the ncs directory. The following image shows the contents of the script.

```
#!/bin/sh

echo "Hello World"

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                    5*1   Command
```

The command *ls* was used to ensure that the new bash script was in the ncs directory. The script was then run by executing *./hello.sh* but permission was denied. To find out why execute permission was denied *ls -l* was executed to check the permissions on the script and the results showed that there were no execute permissions for all users on the server including the owner root. To rectify this the command *chmod a+x hello.sh* was used to give execute permissions to all users. The script was run again by executing *./hello.sh* and the message "Hello World" was successfully printed out on the console window. This can be attributed to the new file permissions which allow the user root to read, write and execute the script, the group to read and execute the script and all others to only execute the script. However, allowing all users to execute the script might not have been the most secure option and the command *chmod u+x hello.sh* could have been used instead only giving execute permissions to only the administrator root (McKinnon, 2017).

```
root@Bry021:/home/ncs# ls
hello.sh
root@Bry021:/home/ncs# ./test.sh
-su: ./test.sh: No such file or directory
root@Bry021:/home/ncs# ./hello.sh
-su: ./hello.sh: Permission denied
root@Bry021:/home/ncs# ls -1
hello.sh
root@Bry021:/home/ncs# ls -l
total 4
-rw-r--r-- 1 root root 33 Nov  2 08:34 hello.sh
root@Bry021:/home/ncs# chmod a+x hello.sh
root@Bry021:/home/ncs# ./hello.sh
Hello World
root@Bry021:/home/ncs# ls -l
total 4
-rwxr-xr-x 1 root root 33 Nov  2 08:34 hello.sh*
root@Bry021:/home/ncs#
```

2.6. Accessing files from different user accounts

The command *su - bob* (McDonnel, 2020) was used to login as bob on the server as shown in the image below. The command *cd /home/ncs* was executed to access the ncs directory and in the ncs directory a $ sign is present suggesting that the current user accessing the directory is not the administrator. The command *./hello.sh* was executed and the message "Hello World" was printed on the screen. This execution success can be attributed to the ncs file having permissions that allow other users other than the administrator on the server to execute the ncs directory (Saive, 2012).

```
root@Bry021:~# su - bob
No directory, logging in with HOME=/

FLASH!
Intelligence of mankind decreasing.
Details at ... uh, when the little hand is on the ....

bob@Bry021:/$
bob@Bry021:/$ cd /home/ncs
bob@Bry021:/home/ncs$ ./hello.sh
Hello World
bob@Bry021:/home/ncs$ _
```

The command *vi new_file* was used to create the script bob.sh in the ncs directory as shown below.

```
#!/bin/sh

echo "Hello this is Bob"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
Read bob.sh, 3 lines, 36 chars                          1,1   Command
```

The command *./bob.sh* was used to execute the script bob.sh. However, the permission to execute this script was denied. The command *ls -l* (McKinnon, 2017) was then executed to check the file permissions on the bob.sh script and as shown in the image below there were no execute permissions for all users including the administrator root. In order to rectify this the command *chmod a+x bob.sh* was executed to add execute permissions on the script. The script was run again and the message "Hello this is Bob" was successfully printed on the console. Another option would have been to use the command *chmod 111* to give execute permissions to all users on the bob.sh script (Rafacz, 2019).

```
bob@Bry021:/home/ncs$ ./bob.sh
-su: ./bob.sh: Permission denied
bob@Bry021:/home/ncs$ ls -l
total 8
-rw-r--r-- 1 bob  bob  36 Nov  2 11:51 bob.sh
-rwxr-xr-x 1 root root 33 Nov  2 08:34 hello.sh*
bob@Bry021:/home/ncs$ chmod a+x bob.sh
bob@Bry021:/home/ncs$ ./bob.sh
Hello this is Bob
bob@Bry021:/home/ncs$ _
```

To log in as smith the command *su – smith* (McDonnell, 2020) was used together with the password smith. The log in was successful and the command *cd /home/ncs* was used to navigate into the /home/ncs directory. The command *ls* was then executed to view the contents of this directory and the command *ls -l* was executed to view the permissions on the files in this directory. Two scripts bob.sh and hello.sh can be found in the /home/ncs directory. Both scripts, bob.sh and hello.sh have read and execute permissions for all users on the server and write permissions for just the administrator root (McKinnon, 2017).

```
bob@Bry021:/$ su - smith
Password:
No directory, logging in with HOME=/

Q:      How many Bell Labs Vice Presidents does it take to change a light bulb?
A:      That's proprietary information.  Answer available from AT&T on payment
        of license fee (binary only).

smith@Bry021:/$ cd /home/ncs
smith@Bry021:/home/ncs$ ls
bob.sh*  hello.sh*
smith@Bry021:/home/ncs$ ls -l
total 8
-rwxr-xr-x 1 bob  bob  36 Nov  2 11:51 bob.sh*
-rwxr-xr-x 1 root root 33 Nov  2 08:34 hello.sh*
smith@Bry021:/home/ncs$ _
```

When *./hello.sh* was executed the message "Hello World" was printed and when *./bob.sh* was executed the message "Hello this is Bob" was printed on the console screen. This shows that both scripts contain execute permissions by other users on the server as both smith and bob could execute the scripts when they are both not administrators.

```
smith@Bry021:/home/ncs$ ./hello.sh
Hello World
smith@Bry021:/home/ncs$ ./bob.sh
Hello this is Bob
smith@Bry021:/home/ncs$ #
```

2.7. Creating Groups and Changing Group Ownerships

3.5.a1. The command *su – root* (McDonnell, 2020) was used to log on the server as root. The command groupadd -g 1010 sysadmins was then used to create the group sysadmins as shown below (Linuxise, 2019).

```
root@Bry021:~# groupadd -g 1010 sysadmins
root@Bry021:~# usermod -g sysadmins bob
root@Bry021:~# usermod -g sysadmins smith
root@Bry021:~# ls -l
total 56
drwxr-xr-x 2 root root 4096 Mar  2  2012 Desktop/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Documents/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Downloads/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Music/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Pictures/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Public/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Templates/
drwxr-xr-x 2 root root 4096 Mar  2  2012 Videos/
-rw-r--r-- 1 root root    0 Nov  1 20:49 internal
drwxrwxrwx 2 root root 4096 Oct 31 20:41 ncs/
-rw-r--r-- 1 root root   39 Oct 31 19:23 new_file
-rw-rw-rw- 1 root root 1706 Nov  2 02:44 new_file1
-rwxr-xr-x 1 root root   37 Nov  1 07:28 test.sh*
drwxr-xr-x 2 root root 4096 Oct 31 13:50 unistuff/
drwxr-xr-x 3 root root 4096 Nov  2 06:12 unixstuff/
root@Bry021:~# getent group sysadmins
sysadmins:x:1010:
root@Bry021:~# _
```

The commands *usermod -G sysadmins bob* and *usermod -G sysadmins smith* were used to add users bob and smith to the group sysadmins and the command *getent group*

sysadmins was executed to verify that the users had been added to the group and the results below show that the users were successfully added to the group (IBM, 2019).

```
root@Bry021:~# usermod -G sysadmins bob
root@Bry021:~# usermod -G sysadmins smith
root@Bry021:~# getent group sysadmins
sysadmins:x:1010:bob,smith
root@Bry021:~#
```

To change the group ownership of /home/ncs, /home/ncs/hello.sh and /home/ncs/bob.sh the *chgrp* command was used as shown below (Fancher, 2018). The *commands ls -l /home/ncs, ls -l /home/ncs/hello.sh* and *ls -l /home/ncs/bob.sh* were executed respectively to ensure that the group ownership of the above files had been successfully changed (McDonnell, 2020). The results below indicate that the group ownership was successfully changed to sysadmins being the owner of /home/ncs, /home/ncs/hello.sh and /home/ncs/bob.sh (Fancher, 2018).

```
root@Bry021:~# chgrp sysadmins /home/ncs
root@Bry021:~# chgrp sysadmins /home/ncs/hello.sh
root@Bry021:~# chgrp sysadmins /home/ncs/bob.sh
root@Bry021:~# ls -l /home/ncs
total 8
-rwxr-xr-x 1 bob  sysadmins 36 Nov  2 11:51 bob.sh*
-rwxr-xr-x 1 root sysadmins 33 Nov  2 08:34 hello.sh*
root@Bry021:~# ls -l /home/ncs/hello.sh
-rwxr-xr-x 1 root sysadmins 33 Nov  2 08:34 /home/ncs/hello.sh*
root@Bry021:~# ls -l /home/ncs/bob.sh
-rwxr-xr-x 1 bob sysadmins 36 Nov  2 11:51 /home/ncs/bob.sh*
root@Bry021:~# _
```

The command *su – bob* (McDonnell,2020) was used to log in as bob and the command *cd /home/ncs* was used to navigate into the ncs directory. Commands *./hello.sh* and *./bob.sh* were used to run hello.sh and bob.sh scripts and the results below show the that both scripts were run successfully.

```
root@Bry021:~# su - bob
No directory, logging in with HOME=/

Q:      Why did the chicken cross the road?
A:      To see his friend Gregory peck.

Q:      Why did the chicken cross the playground?
A:      To get to the other slide.

bob@Bry021:/$ cd /home/ncs
bob@Bry021:/home/ncs$ ./hello.sh
Hello World
bob@Bry021:/home/ncs$ ./bob.sh
Hello this is Bob
bob@Bry021:/home/ncs$ _
```

The command *su – smith* (McDonnell, 2020) was used to log in as smith and the command *cd /home/ncs* was used to navigate into the ncs directory. Commands *./hello.sh* and *./bob.sh* were used to run hello.sh and bob.sh scripts and the results below show the that both scripts were run successfully.

```
root@Bry021:~# su - smith
No directory, logging in with HOME=/

Whenever I feel like exercise, I lie down until the feeling passes.

smith@Bry021:/$ cd /home/ncs
smith@Bry021:/home/ncs$ ./hello.sh
Hello World
smith@Bry021:/home/ncs$ ./bob.sh
Hello this is Bob
smith@Bry021:/home/ncs$ cd
smith@Bry021:/$ _
```

Both scripts were run successfully by users bob and smith because they both have execute permissions on the scripts as shown below.

```
bob@Bry021:/$ su - root

Some men are discovered; others are found out.

root@Bry021:~# /home/ncs
-su: /home/ncs: is a directory
root@Bry021:~# cd /home/ncs
root@Bry021:/home/ncs# ls -l
total 8
-rwxr-xr-x 1 bob  sysadmins 36 Nov  2 11:51 bob.sh*
-rwxr-xr-x 1 root sysadmins 33 Nov  2 08:34 hello.sh*
root@Bry021:/home/ncs# _
```

To disable the user smith's account the command *passwd smith -l* was used. Most Linux systems utilize the /etc/shadow file to retain any encrypted user passwords therefore the method used only changes this shadow file by adding "!" in front of the user smith's password. This allows the administrator to keep the account active without allowing the user smith to use the smith account. To re-enable the account for smith the command *passwd smith -u* can be used which removes the "!" character from the user smith's password line in /etc/shadow (MDLog:/sysadmin, 2007).

```
root@Bry021:~# passwd smith -l
passwd: password expiry information changed.
root@Bry021:~# su - smith
No directory, logging in with HOME=/

better !pout !cry
better watchout
lpr why
santa claus <north pole >town
```

The image below confirms that the account smith was disabled as smith can no longer login into his account using the password smith which was created for him when the account was created.

```
bob@Bry021:/$ su - smith
Password:
su: Authentication failure
bob@Bry021:/$ _
```

3. Daemons and Processes

3.1. Exploring currently running processes

The *ps aux* command is used to identify the processes currently running on the Linux system by producing a process listing such as the one shown below:

```
root      1888  0.0  1.0  21524  2644 ?        Sl   Oct31   0:00 /usr/libexec/po
root      1909  0.0  0.4   3672  1208 ?        S    Oct31   0:00 hald-runner
root      1941  0.0  0.4   3768  1024 ?        S    Oct31   0:00 hald-addon-inpu
root      1954  0.0  0.4   3772  1028 ?        S    Oct31   0:03 hald-addon-stor
82        1958  0.0  0.4   3576  1144 ?        S    Oct31   0:00 hald-addon-acpi
root      1959  0.0  0.4   3772  1028 ?        S    Oct31   0:00 hald-addon-stor
root      1979  0.0  0.2   2080   664 ?        Ss   Oct31   0:00 /usr/sbin/crond
daemon    1981  0.0  0.1   2072   316 ?        Ss   Oct31   0:00 /usr/sbin/atd -
root      2006  0.0  0.1   2068   376 ?        Ss   Oct31   0:04 /usr/sbin/gpm -
root      2009  0.0  0.2   1852   520 tty2     Ss+  Oct31   0:00 /sbin/agetty 38
root      2010  0.0  0.2   1852   520 tty3     Ss+  Oct31   0:00 /sbin/agetty 38
root      2011  0.0  0.2   1852   520 tty4     Ss+  Oct31   0:00 /sbin/agetty 38
root      2012  0.0  0.2   1852   516 tty5     Ss+  Oct31   0:00 /sbin/agetty 38
root      2013  0.0  0.2   1852   524 tty6     Ss+  Oct31   0:00 /sbin/agetty 38
root      2383  0.0  0.7   3368  1808 tty1     Ss   Oct31   0:00 -bash
root      2885  0.0  0.4   2768  1136 ?        S<   Oct31   0:00 /sbin/udevd --d
bob       2958  0.0  0.7   3352  1792 tty1     S    Oct31   0:00 -su
root      2998  0.0  0.0      0     0 ?        S<   Oct31   0:00 [hci0]
root      3006  0.0  0.4   2768  1128 ?        S<   Oct31   0:00 /sbin/udevd --d
root      3009  0.0  0.4   3768  1028 ?        S    Oct31   0:00 /usr/libexec/ha
smith     3031  0.0  0.7   3356  1792 tty1     S    Oct31   0:00 -su
root      3208  0.0  0.0      0     0 ?        S    01:16   0:00 [flush-8:0]
root      3209  0.0  0.7   3360  1796 tty1     S    01:20   0:00 -su
root      3226  0.0  0.3   2732   964 tty1     R+   01:23   0:00 ps aux
root@Bry021:~# _
```

To count the number of processes being run on the Linux system by any user the command *ps aux | wc -l* can be executed (Linux Tweaks for You, Anon., 2015). The total number of the processes running in the above image are 24. There is a vast amount of *ps* options, *ps ax* when executed also provides a listing of currently running processes. However, the listing displayed will only provide essential minimal information such as command names and PID (Process ID) values of the processes but adding the u to make it *ps aux* enhances the listing by displaying other variables such as CPU loads and usernames as shown above (Smith, 2012).

The *ps* command sorts the process listing by the PID values and only presents the list at a point in time (Blum and Bresnahan, 2015). To view the listing by CPU use and observe the most CPU-intensive processes in real time which are frequently swapped in and out of the system's memory the *top* command can be used. The top command also allows you to employ numerous single character commands

such as the *r* command which can change a process's priority and the *h* command which displays help information (Smith, 2012). The following image shows the screen displayed by the Linux system after executing the *top* command.

```
top - 02:59:24 up 14:10,  1 user,  load average: 0.00, 0.01, 0.05
Tasks:  99 total,   1 running,  98 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    242128k total,   228980k used,    13148k free,    93444k buffers
Swap:  2815388k total,        0k used,  2815388k free,   103624k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      20   0   824  276  236 S  0.0  0.1   0:02.92 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0     0    0    0 S  0.0  0.0   0:00.18 ksoftirqd/0
    5 root      20   0     0    0    0 S  0.0  0.0   0:03.38 kworker/u:0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    7 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 cpuset
    8 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 khelper
    9 root      20   0     0    0    0 S  0.0  0.0   0:03.39 kworker/u:1
   12 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 netns
  274 root      20   0     0    0    0 S  0.0  0.0   0:00.39 sync_supers
  276 root      20   0     0    0    0 S  0.0  0.0   0:00.02 bdi-default
  278 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kblockd
  280 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpid
  281 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  282 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 kacpi_hotplug
  445 root       0 -20     0    0    0 S  0.0  0.0   0:00.00 ata_sff
  453 root      20   0     0    0    0 S  0.0  0.0   0:00.19 khubd
  456 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kseriod
```

The command *ps -eo pid,comm,%cpu,%mem –sort=-%cpu | head -n 10* was executed to identify the ten most CPU intensive processes. This command allows you to specify the output format with *-eo* such as showing the PIDs and %CPU. The sort part allows you to sort the commands either by %CPU or %MEM while the *head -n 10* tells the system to show the top 10 processes currently running (Canepa, 2016).

```
root@Bry021:~# ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -n 10
  PID COMMAND         %CPU %MEM
 1372 kworker/0:2      0.0  0.0
 1253 jbd2/sda1-8      0.0  0.0
 2006 gpm              0.0  0.1
    9 kworker/u:1      0.0  0.0
 1954 hald-addon-stor  0.0  0.4
    5 kworker/u:0      0.0  0.0
 1667 dhcpcd           0.0  0.2
 1097 scsi_eh_3        0.0  0.0
    1 init             0.0  0.1
root@Bry021:~# _
```

The following are the top 10 processes currently running on Linux:

i. Kworker/2.0 is a placeholder process for kernel worker threads and it carries out most of the processing for the kernel particularly in places where there are interrupts, timers and I/O (StackExchange, 2019).
ii. Jbd2/sda1-8 is a kernel process associated with journaling on the ext4 filesystem.
iii. Gpm is a process that provides mouse support to text-based Linux applications such as vi text and Emacs (nixCraft, 2006)
iv. Kworker/u:1 a variation of kworker with the same function of being a placeholder process for kernel worker threads.
v. Hald-addon-stor is function of the hal daemon that allows you to mount CDs and DVDs.
vi. Kworker/u:0 another variation of kworker with the function of being a placeholder process for kernel worker threads (StackExchange, 2019).
vii. Dhcpcd is a dynamic host configuration protocol and a network protocol which is used on IP networks where a DCP server automatically assigns an IP address and other information to each host on the network

so they can communicate efficiently with other endpoints (Keravala, 2018).

viii. Scsi_eh_3 is an error handling process and a high CPU usage time by the process could be an indicator of issues on your hard drives (StackExchange, 2019).

ix. Init is responsible for initializing the system in the required way and it can resist signal 9 which normally kills the process when started by the kernel (StackExchange, 2019).

x. Sync_super is a process that is responsible for writing to the disk (StackExchange, 2019).

3.2. Exploring network processes

The command *nmap localhost* was executed to view the network hosts and services being run on the Linux Server and the results are shown in the image below:

```
root@Bry021:~# nmap localhost

Starting Nmap 5.51 ( http://nmap.org ) at 2019-11-01 15:29 GMT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000040s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
37/tcp  open  time
113/tcp open  auth

Nmap done: 1 IP address (1 host up) scanned in 0.52 seconds
root@Bry021:~# _
```

The following processes where returned:

i. Ssh is utilized when logging into the remote machine, executing commands on the machine and transferring files between two machines (Henry-Stocker, 2017).

ii. Time synchronizes the software clock of a GNU/Linux system with internet time server mitigating the effects of variable network latency also maintaining time within tens of milliseconds over the public internet (Cezar, 2018).

iii. Auth is used to authenticate a user and set up user credentials (Kili, 2018).

3.3. Exploring UNIX Signals

The command kill -l was executed and the following results were obtained:

```
root@Bry021:~# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

The command *kill* is capable of transmitting signals to running processes on the server. This command contains various options and is manly used to terminate program executions. For this command to be used it is useful to always obtain the process PID first. The *-l* option of the command kill giving the command *kill -l* allows you to view a list of signals that can sent to either halt, kill or start processes. Another variation of the kill command which *killall* can be used to terminate all programs running with a particular name or the same name (Nooning, 2003).

Table 1 Results of the command kill -1 together with signal numbers and functions of the signals

Signal Name	Signal Number	Description
SIGHUP	1	Hangs up the process
SIGINT	2	Interrupts the process
SIGQUIT	3	Stops the process
SIGILL	4	An illegal signal which is sent when a process carries out an unknown or faulty process.
SIGTRAP	5	Used for debugging purposes when a condition that a debugger is waiting for has been met.
SIGABRT	6	A kill signal that is initiated by the process as an abort signal.
SIGBUS	7	Sent to processes with a bus errors such an inaccurately set memory alignment.
SIGFPE	8	Used to kill processes that divide by zero.
SIGKILL	9	Unconditionally terminates the process
SIGTERM	15	Terminates the process if possible
SIGSTOP	17	Unconditionally stops but does not terminate the process
SIGTSTP	18	Stops or pauses the process without terminating it
SIGCONT	19	Continues a stopped process
SIGVTALRM	26	Sent to process whose CPU time usage elapsed.

Blum and Bresnahan, 2015.

3.4. Processes and Networking

In order to edit /etc/inetd.conf the command *chmod 600 /etc/inetd.conf* was first used to change the permissions on the file allowing the user root to be able to read the file and edit it. The command *stat /etc/inetd.conf* was then executed to view the properties of the file etc/inetd.conf and the command *vi /etc/inetd.conf* was executed to access the file in the vi test editor (IBM, 2019).

```
root@Bry021:~# chmod 600 /etc/inetd.conf
root@Bry021:~# stat /etc/inetd.conf
  File: '/etc/inetd.conf'
  Size: 4599            Blocks: 16         IO Block: 4096   regular file
Device: 801h/2049d      Inode: 146251      Links: 1
Access: (0600/-rw-------)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-11-01 20:48:20.973008071 +0000
Modify: 2007-06-30 21:02:57.000000000 +0100
Change: 2019-11-02 17:48:39.973022946 +0000
 Birth: -
root@Bry021:~# vi /etc/inetd.conf_
```

In the vi text editor ftp and telnet were enabled by removing the "#" symbol at the beginning of their lines as shown below (IBM, 2019).

```
# daytime       stream  tcp     nowait  root    internal
# daytime       dgram   udp     wait    root    internal
# chargen       stream  tcp     nowait  root    internal
# chargen       dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
#
# These are standard services:
#
# Very Secure File Transfer Protocol (FTP) server.
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  vsftpd
#
# Professional File Transfer Protocol (FTP) server.
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  proftpd
#
 Telnet server:
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
#
# The comsat daemon notifies the user of new mail when biff is set to y:
comsat          dgram   udp     wait    root    /usr/sbin/tcpd  in.comsat
#
# Shell, login, exec and talk are BSD protocols
#
#shell  stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -L
                                                                41,1    Input
```

The command *kill -HUP 1741* was executed to restart inetd. To connect to ftp and telnet localhost *ftp localhost* and *telnet localhost* was executed and there were successful connections to both hosts as shown below.

```
wrote /etc/inetd.conf, 109 lines, 4595 chars
root@Bry021:~# kill -HUP 1741
root@Bry021:~# ftp localhost
Connected to localhost.
220 ProFTPD 1.3.3e Server (ProFTPD Default Installation) [127.0.0.1]
Name (localhost:root): telnet localhost
331 Password required for telnet
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
root@Bry021:~# telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Bry021 login:
```

The command *su – bob* was used to login as bob and the commands ftp localhost and telnet localhost were again executed. Successful connections to telnet and ftp were again obtained.

```
root@Bry021:~# su - bob
No directory, logging in with HOME=/

Humility is the first of the virtues -- for other people.
                -- Oliver Wendell Holmes

bob@Bry021:/$ ftp localhost
Connected to localhost.
220 ProFTPD 1.3.3e Server (ProFTPD Default Installation) [127.0.0.1]
Name (localhost:root): quit
331 Password required for quit
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quit
221 Goodbye.
bob@Bry021:/$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Bry021 login:
```

To edit /etc/inetd.conf again *vi /etc/inetd.conf* was executed to access the inetd file in the vi text editor. The ftp and telnet daemons were disabled by adding the "#" symbol at the beginning of the lines containing the daemons (IBM, 2019).

```
# chargen       stream  tcp     nowait  root    internal
# chargen       dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
#
# These are standard services:
#
# Very Secure File Transfer Protocol (FTP) server.
#ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  vsftpd
#
# Professional File Transfer Protocol (FTP) server.
#ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  proftpd
#
# Telnet server:
#telnet stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
#
# The comsat daemon notifies the user of new mail when biff is set to y:
comsat          dgram   udp     wait    root    /usr/sbin/tcpd  in.comsat
#
# Shell, login, exec and talk are BSD protocols
#
#shell  stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -L

wrote /etc/inetd.conf, 109 lines, 4599 chars
root@Bry021:~# _
```

The command *kill -HUP 1741* was used to restart inetd and ftp localhost and telnet localhost were executed but no connections were obtained illustrating that ftp and telnet were successfully disabled.

```
wrote /etc/inetd.conf, 109 lines, 4599 chars
root@Bry021:~# kill -HUP 1741
root@Bry021:~# ftp localhost
ftp: connect: Connection refused
ftp> quit
root@Bry021:~# telnet localhost
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
root@Bry021:~#
```

Secure Shell (SSH) is a transport layer that is used in securing logins and information moving from one end to another. It uses both asymmetric, which is public and private key, and symmetric cryptology to provide excellent performance and strong encryption. SSH is also capable of providing secure communications between remote servers and an organization (GoAnywhere, 2019). SHH works in conjunction with a file transfer protocol called SFTTP (SSH File Transfer protocol) to provide secure communications. SFTP is built upon the SSH transport layer and is used to securely transfer enormous amounts of data over an internet connection (Vincent, 2016). It works by implementing a secure authenticated connection providing organizations with a higher level of protection over file transfers and uses the SSH authentication and cryptogenic capabilities to keep files safe during any transfer processes (GoAnywhere, 2019).

The use of Telnet is discouraged because all its data is transferred unencrypted over the network. Usernames and passwords are always sent in plain text meaning that anyone at any point can capture a user's login information or nay data being managed over the Telnet connection (SSH.COM, 2019).

5. Email Under Linux

4.1. Sending Email Using Mail

6.1.1 The command *ls -l /etc/rc.d/rc.sendmail* was executed to check the permissions on the sendmail process. The results showed that there were read and write permissions for the user root but only read permissions for all others on the server. To give execute permissions to all users the *chmod a+x /etc/rc.d/rc.sendmail* was executed followed *by ls -l /etc/rc.d/rc.sendmail* to check if the chmod execution had been successful. The results show that execute permissions were successfully added on to the sendmail process. A new file was then created in the vi text editor using the *vi new_file* command.

```
root@Bry021:~# ls -l /etc/rc.d/rc.sendmail
-rw-r--r-- 1 root root 687 Jun  4  2002 /etc/rc.d/rc.sendmail
root@Bry021:~# chmod a+x /etc/rc.d/rc.sendmail
root@Bry021:~# ls -1 /etc/rc.d/rc.sendmail
/etc/rc.d/rc.sendmail*
root@Bry021:~# ls -l /etc/rc.d/rc.sendmail
-rwxr-xr-x 1 root root 687 Jun  4  2002 /etc/rc.d/rc.sendmail*
root@Bry021:~# vi new_file_
```

The image below shows the contents of the new file saved as message.txt in the vi text editor.

```
#!/bin/bash
clear
echo "Hello Bob when you are able to maintain your own highest standards of inte
grity-regardless of what others may do-you are destined for greatness"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
Read message.txt, 4 lines, 172 chars                          1,1   Command
```

The command *mail -s hellobob-mail bob@localhost < message.txt* was used to send an email to the user bob as shown below. There are other commands that can be used to send mail on the Linux server. For instance, the sendmail command can be used instead of the mail command as follows, *sendmail bob@localhost < message.txt*. It is also possible to send mail with an attachment or attachments on the Linux server using the mutt command. For instance, the following command could have been used to send mail to bob which contains the speakers attachment as shown in the command *mutt -s "hellobob-mail" bob@localhost -a speakers < message.txt* using the *-a* option (Henry-Stocker, 2019).

```
root@Bry021:~# mail -s hellobob-mail bob@localhost < message.txt
root@Bry021:~# _
```

The command below is an alternative that can be used to send an email to other users. The first part echo is followed by a message or messages to be sent to the recipients. The *-c* represents the carbon copy (Cc) which is where you put email addresses if you are sending information to more than one recipient and would like the email addresses to be visible to everyone receiving the email while the *-b* which represents Blind carbon copy (Bcc) also allows you to send information to various recipients without allowing them to view which other individuals have also received the same information(Baydan, 2017) .

```
root@Bry021:~# echo "Welcome to Network Computer Systems" | mail -s "Hello world
" bob@anglia.bryant -c smith@anglia.bryant -b root@anglia.bryant
root@Bry021:~# Invalid operation mode -
```

The command *su – bob* was used to login as bob on the server and the command *mail* was used to check for mail from root as shown below. The message returned by the server was that there was no mail for bob.

```
root@Bry021:~# su - bob
No directory, logging in with HOME=/

The very ink with which all history is written is merely fluid
prejudice.
                -- Mark Twain

bob@Bry021:/$ mail
No mail for bob
```

To rectify the reason why bob could not access the mails sent the command *su – root* was used to log back into root's account. The command *chmod a+x /etc/rc.d/rc.sendmail* was then executed to make sure all users could execute the sendmail process and the command */etc/rc.d/rc.sendmail start* was executed to start the sendmail process.

```
root@Bry021:~# ls /etc/rc.d
init.d/         rc.cups         rc.messagebus*          rc.sysstat
rc.0@           rc.dnsmasq      rc.modules@             rc.sysvinit*
rc.4*           rc.font         rc.modules-2.6.37.6*    rc.udev*
rc.6*           rc.fuse*        rc.modules-2.6.37.6-smp* rc.wireless*
rc.K*           rc.gpm*         rc.mysqld               rc.wireless.conf
rc.M*           rc.hald*        rc.nfsd                 rc.yp*
rc.S*           rc.httpd        rc.ntpd                 rc0.d/
rc.acpid*       rc.inet1*       rc.pcmcia               rc1.d/
rc.alsa*        rc.inet1.conf   rc.rpc                  rc2.d/
rc.atalk        rc.inet2*       rc.samba                rc3.d/
rc.autofs       rc.inetd*       rc.saslauthd            rc4.d/
rc.bind         rc.ip_forward   rc.sendmail*            rc5.d/
rc.bluetooth    rc.keymap*      rc.serial               rc6.d/
rc.cgconfig     rc.local*       rc.snmpd
rc.cgred        rc.loop*        rc.sshd*
rc.consolekit*  rc.mcelog*      rc.syslog*
root@Bry021:~# chmod a+x /etc/rc.d/rc.sendmail
root@Bry021:~# /etc/rc.d/rc.senmail start
-su: /etc/rc.d/rc.senmail: No such file or directory
root@Bry021:~# /etc/rc.d/rc.sendmail start
Starting sendmail MTA daemon:  /usr/sbin/sendmail -L sm-mta -bd -q25m
Starting sendmail MSP queue runner:  /usr/sbin/sendmail -L sm-msp-queue -Ac -q25
m
root@Bry021:~#
```

The command su – bob was then again used to log back into bob's account and command mail was used vies the mail messages sent to bob as shown below.

```
root@Bry021:~# su - bob
No directory, logging in with HOME=/

Hire the morally handicapped.

bob@Bry021:/$ mail
Heirloom mailx version 12.4 7/29/08.  Type ? for help.
"/var/spool/mail/bob": 7 messages 7 new
>N  1 root@Bry021.bryant Sun Nov  3 02:57   20/777   Hello-Bob
 N  2 root@Bry021.bryant Sun Nov  3 02:57   20/781   hellobob-mail
 N  3 root@Bry021.bryant Sun Nov  3 02:57   21/715   testing-mail
 N  4 root@Bry021.bryant Sun Nov  3 02:57   20/839   Hello-Bob
 N  5 root@Bry021.bryant Sun Nov  3 02:57   23/857   Test
 N  6 root@Bry021.bryant Sun Nov  3 02:57   23/862   Hello-Bob
 N  7 root@Bry021.bryant Sun Nov  3 02:57   23/866   hellobob-mail
?
```

4.2. Exploring Mail

Reading Mail: To read any mail the command *print* followed by the massage number can be executed as shown below.

```
? print 7
Message  7:
From root@Bry021.bryant  Sun Nov  3 02:57:09 2019
Return-Path: <root@Bry021.bryant>
From: root@Bry021.bryant
Date: Sun, 03 Nov 2019 00:45:36 +0000
To: bob@Bry021.bryant
Subject: hellobob-mail
User-Agent: Heirloom mailx 12.4 7/29/08
Content-Type: text/plain; charset=us-ascii
Status: R

#!/bin/bash
clear
echo "Hello Bob when you are able to maintain your own highest standards of inte
grity-regardless of what others may do-you are destined for greatness"
~

?
```

Replying Mail: To reply to *mail* the command reply followed by the number of the message you want to reply to can be used as shown below.

```
? reply 7
To: bob@Bry021.bryant root@Bry021.bryant
Subject: Re: hellobob-mail

root@Bry021.bryant wrote:

> #!/bin/bash
> clear
> echo "Hello Bob when you are able to maintain your own highest standards of in
tegrity-regardless of what others may do-you are destined for greatness"
> ~
Thank you
.
EOT
? _
```

Sending Mail: The command *mail root@slackware-lab.slackware .com* can be used to send a message as shown below.

```
? mail root@slackware-lab.slackware.com
Subject: Thank You
Thank you for the message.
.
.
EOT
New mail has arrived.
Loaded 1 new message
 N  8 bob@Bry021.bryant  Sun Nov  3 04:12   28/1003  Re: hellobob-mail
? _
```

Saving Mail: To save mail the command *save* can be executed followed by the number of the mail you want to save as shown below.

```
.
EOT
? save 1
"1" [New file] 22/858
?
```

Listing Messages: To list messages various commands can be used. The commands *f** and *h* can be used to list all the messages in the inbox. The *f* command can also be varied by adding numbers to it for instance the *f 1-8* command can be used to only list emails 1-8.

```
? f*
>U  1 root@Bry021.bryant Sun Nov  3 02:57   21/787   Hello-Bob
 O  2 root@Bry021.bryant Sun Nov  3 02:57   21/792   hellobob-mail
 U  3 root@Bry021.bryant Sun Nov  3 02:57   22/725   testing-mail
 U  4 root@Bry021.bryant Sun Nov  3 02:57   21/849   Hello-Bob
 U  5 root@Bry021.bryant Sun Nov  3 02:57   24/867   Test
 U  6 root@Bry021.bryant Sun Nov  3 02:57   24/872   Hello-Bob
 O  7 root@Bry021.bryant Sun Nov  3 02:57   24/877   hellobob-mail
 U  8 bob@Bry021.bryant  Sun Nov  3 04:12   29/1013  Re: hellobob-mail
 N  9 Mail Delivery Subs Sun Nov  3 04:16   68/2341  Returned mail: see transc
? h
 U  1 root@Bry021.bryant Sun Nov  3 02:57   21/787   Hello-Bob
 O  2 root@Bry021.bryant Sun Nov  3 02:57   21/792   hellobob-mail
 U  3 root@Bry021.bryant Sun Nov  3 02:57   22/725   testing-mail
 U  4 root@Bry021.bryant Sun Nov  3 02:57   21/849   Hello-Bob
 U  5 root@Bry021.bryant Sun Nov  3 02:57   24/867   Test
 U  6 root@Bry021.bryant Sun Nov  3 02:57   24/872   Hello-Bob
 O  7 root@Bry021.bryant Sun Nov  3 02:57   24/877   hellobob-mail
 U  8 bob@Bry021.bryant  Sun Nov  3 04:12   29/1013  Re: hellobob-mail
>N  9 Mail Delivery Subs Sun Nov  3 04:16   68/2341  Returned mail: see transc
? _
```

/var/spool/mail/ contains flat text files that serve as the user's mailbox (IBM, 2018). Sendmail uses mail spools to do this and a spool is any file that saves the mail header such the sender's address or time of delivery as well as the message body for every mail (SimplyWebHosting.com, 2014). Sendmail mailboxes are owned by mail and not users therefore the nature of sendmail can allow mail attackers to flood the server with mail easily leading to denial of the service. Having said that, the effectiveness of such attacks is limited (RedHat, 2020).

The SMTP server is responsible for outgoing messages. For a non-encrypted SMTP server Port 25 or 587 can be used but for secure servers such as TLS and SSL ports 587 and 465 can be used respectively. On the other hand,, POP3 is a server responsible for incoming messages and port 110 is commonly used for a non-encrypted POP3 server while port 995 is commonly used for a secure POP3 server (ARCLAB, 2020).

4.3. Optional Exercises

IMAP and POP3 are both ways of connecting to the mail server so that one can read emails through a mail client. IMAP is short for Internet Message Access Protocol. This protocol does not allow messages to be saved on a computer device therefore the messages remain on the server. However, POP3 on the other hand works by allowing mail to be saved or kept on the computer device or other output device. The table below describes the differences between IMAP and POP3 (Name com, 2019).

Table 2 Differences between IMAP and POP3

IMAP	POP3
Sent messages are saved on the server.	Sent messages are kept on a single device.
Messages can be synched and accessed across multiple devices.	Emails can only be accessed from a single device.
Emails are stored on the server	The emails are kept on a single device.
	Messages can be kept on the server, however, the setting "keep email on server" must be enabled for this to be possible. Having said that, once the messages are downloaded they are then instantly deleted from the server.

Aol.Help, 2019.

PGP which is an abbreviation of Pretty Good Privacy is a command line tool used in transferring sensitive data or files securely between two systems. In order to encrypt a file or data, a public key which is also shared with end users is needed. This key has to be generated on a source system using the command *gpg –gen-key*. After the public key has been installed it is then used to encrypt a file either using a passphrase or without using a passphrase. To encrypt a file using a passphrase the following command is used, *pg -s --no-tty --always-trust --passphrase "passphrase@test" -u <Key_Name>-pub-sub.asc "data_file.txt"* and to encrypt a file without a passphrase the following command is used, *gpg --encrypt --recipient '<Key_Name>-pub-sub.asc' data_file.txt* (Chauhan, 2019).

5. Apache HTTP Server and PHP

5.1. Configuring Apache

The following file was opened in the vi text editor using the command *vi /etc/httpd/httpd.conf*. To browse through the file the syntax */search_string* was used.

```
#
# This is the main Apache HTTP server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.2> for detailed information.
# In particular, see
# <URL:http://httpd.apache.org/docs/2.2/mod/directives.html>
# for a discussion of each configuration directive.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "/var/log/httpd/foo_log"
# with ServerRoot set to "/usr" will be interpreted by the
# server as "/usr//var/log/httpd/foo_log".

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# Do not add a slash at the end of the directory path.  If you point
Read /etc/httpd/httpd.conf, 484 lines, 17661 chars              1,1   Command
```

Lines that start with # which are comments have a general purpose of temporarily disabling a specific code meaning that anything that comes after the # will not be executed. The comments disable specific http commands and /or options in httpd.conf (Buzdar, 2019).

The default value of ServerName is www.example .com:80 while that of DocumentRoot is “/srv/httpd/htdocs”. Default values allow you to specify a file for Apache which can be used for specific error events (Ubuntu Documentation, 2020). .

```
# This can often be determined automatically, but we recommend you specify
# it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
#
#ServerName www.example.com:80

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/srv/httpd/htdocs"
```

The line Include /etc/httpd/mod_php.conf was uncommented by removing the # at the beginning of the line as shown in the image below.

```
  GNU nano 2.3.0              File: httpd.conf                          Modified

#       starting without SSL on platforms with no /dev/random equivalent
#       but a statically compiled-in mod_ssl.
#
<IfModule ssl_module>
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
</IfModule>

# Uncomment the following line to enable PHP:
#
Include /etc/httpd/mod_php.conf

# Uncomment the following lines to enable svn support:
#
#LoadModule dav_svn_module lib64/httpd/modules/mod_dav_svn.so
#LoadModule authz_svn_module lib64/httpd/modules/mod_authz_svn.so
```

5.2. Running Apache

The command *ls -l /etc/httpd/httpd.conf* was executed to view the permissions on the Apache HTTP daemon. There were no execute permissions on the daemon for that reason the command *chmod a+x /etc/rc.d/rc.httpd* was executed to add execution permissions on the daemon. These changes were confirmed by executing *ls -l /etc/rc/d/rc.httpd.*

```
wrote /etc/httpd/httpd.conf, 484 lines, 17649 chars
root@Bry021:~# ls -l /etc/rc.d/rc.httpd
-rw-r--r-- 1 root root 703 Feb 12  2011 /etc/rc.d/rc.httpd
root@Bry021:~# chmod a+x /etc/rc.d/rc.httpd
root@Bry021:~# ls -l /etc/rc.d/rc.httpd
-rwxr-xr-x 1 root root 703 Feb 12  2011 /etc/rc.d/rc.httpd*
root@Bry021:~#
```

It is essential to restart httpd after you make changes to the configuration because it enables the changes made to take effect when it restarts and starts running again.

The *ps aux* command is a command used to display all process on the server and can be broken down in parts in order to understand its function. The *ps* command lists processes running under the logged in user account from the current terminal. The *a* option prints any running processes from all users on the server, the *u* option shows the user or owner column in the output and the *x* option prints the processes that have not been executed from the terminal (ComputerNetworkingNotes, 2020).

The *grep* command searches files for specific words or patterns therefore *grep httpd* searches for any files, processes or daemons containing the word or phrase httpd. From this we can deduce that the *ps aux | grep httpd* command lists all processes, daemons or files on the server that contain the word/phrase httpd.

The expected result from the command ps aux | grep httpd is the return of the details of the daemon httpd as shown below.

```
root@Bry021:~# ps aux | grep httpd
root      7922  0.0  0.3   2440   788 tty1     S+   07:34   0:00 grep httpd
root@Bry021:~# /etc/rc.d/rc.httpd start
Syntax error on line 156 of /etc/httpd/httpd.conf:
ServerName takes one argument, The hostname and port of the server
root@Bry021:~# ps aux | grep httpd
root      7928  0.0  0.3   2440   788 tty1     S+   07:37   0:00 grep httpd
```

If the daemon is not running the expected result will be file not running or no process found as shown below.

```
root@Bry021:~# /etc/rc.d/rc.httpd stop
Syntax error on line 156 of /etc/httpd/httpd.conf:
ServerName takes one argument, The hostname and port of the server
httpd (no pid file) not running
httpd: no process found
root@Bry021:~# ps aux | grep httpd
root      7936  0.0  0.3   2440   788 tty1     S+   07:38   0:00 grep httpd
root@Bry021:~#
```

The command ps axl | egrep “httpd|PID” was executed and the following results were obtained. The image below shows that the PPID of the parent httpd process in 1.

```
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
1     0  3309     1  20   0 379612 13328 poll_s Ss   ?          0:00 /usr/sbin/h
ttpd -k start
5    80  3310  3309  20   0 379612  7064 inet_c S    ?          0:00 /usr/sbin/h
ttpd -k start
5    80  3311  3309  20   0 379612  7064 inet_c S    ?          0:00 /usr/sbin/h
ttpd -k start
5    80  3312  3309  20   0 379612  7064 inet_c S    ?          0:00 /usr/sbin/h
ttpd -k start
5    80  3313  3309  20   0 379612  7064 inet_c S    ?          0:00 /usr/sbin/h
ttpd -k start
5    80  3314  3309  20   0 379612  7064 inet_c S    ?          0:00 /usr/sbin/h
ttpd -k start
0     0  3316  3194  20   0   6756   984 pipe_w S+   tty1       0:00 egrep httpd
|PPID
```

5.3. Creating HTML Files

The following image shows the results of executing the command ls -l inside the /srv/httpd/htdocs file. The file named index.html can be seen inside the file.

```
total 12
drwxr-xr-x  2 root root 4096 Oct 15  2008 htdig/
-rw-r--r--  1 root root   44 Nov 20  2004 index.html
drwxr-xr-x 14 root root 4096 Feb 12  2011 manual/
```

Index.html and index.htm files are special because they are local files or URLs that automatically load when a web browser starts. They are default ages shown on a website if no other page is specified when a visitor requests a site (Kyrnin, 2019).

As indicated in the image below the file permissions for the index.html file are read and write permissions for root and only read permissions for the group on the server and any other users. Root has the group ownership of this file.

```
total 12
drwxr-xr-x  2 root root 4096 Oct 15  2008 htdig/
-rw-r--r--  1 root root   44 Nov 20  2004 index.html
drwxr-xr-x 14 root root 4096 Feb 12  2011 manual/
```

The command *vi new_file* was executed to create a new html file called test.html and the contents of this file are shown below:

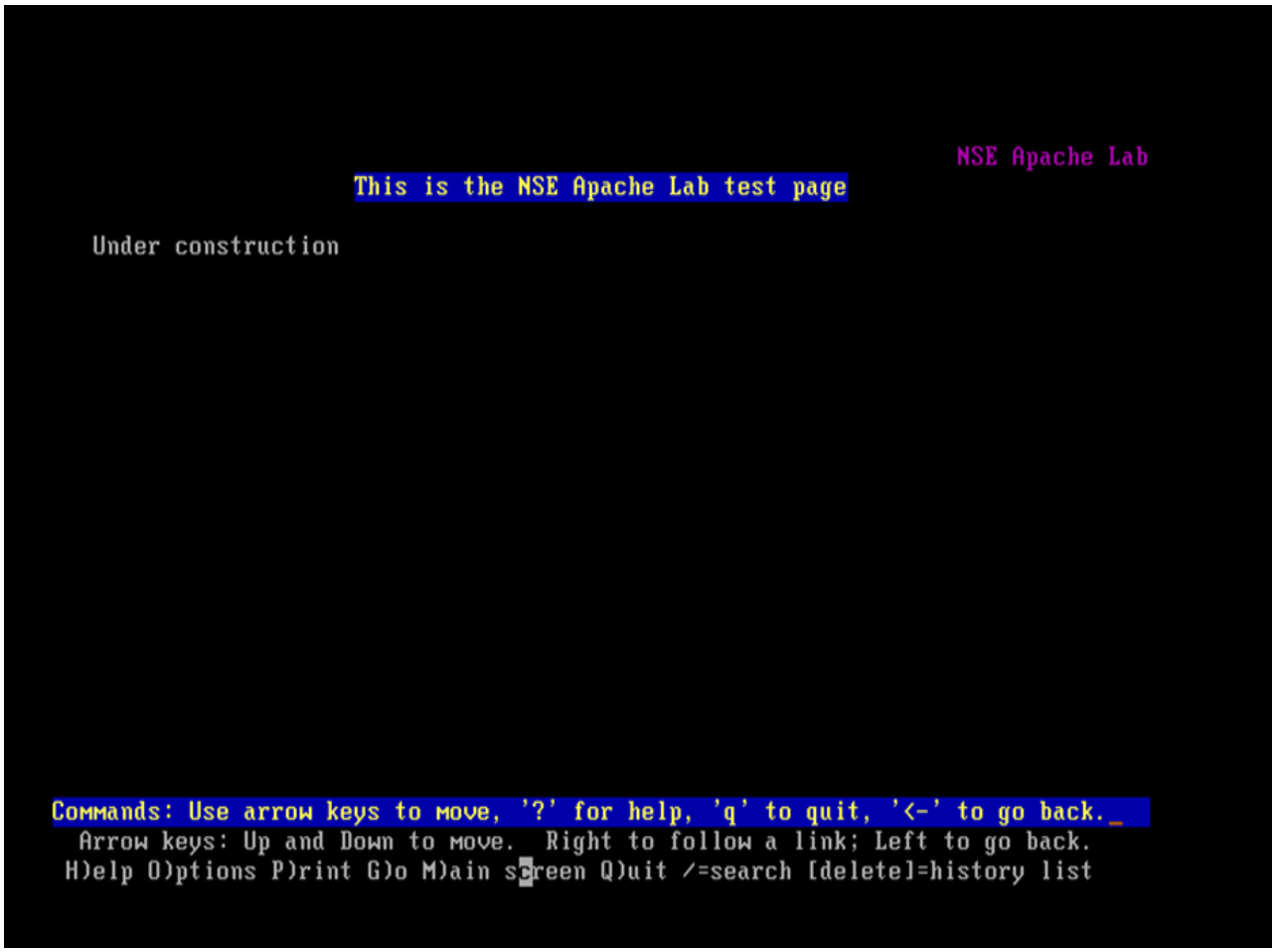
```
<html>
<head><title>NSE Apache Lab</title></head>
<body>
<h1>This is the NSE Apache Lab test page</h1>
<p>Under construction!</p>
</body>
</html>
~
~
~
~
~
~
~
~
~
~
~
~
~
~
wrote test.html, 10 lines, 150 chars                    9,7   Command
```

5.4. Viewing HTML Files Using a Terminal Interface

5.4.1. CLI is a command line interface while GUI is a graphical user interface. The main distinction between CLI and GUI is that GUI allows a user to interct with a system using graphical elements such as windows and icons while CLI alows a user to interact with a system using commands instead (Lithmee, 2018).

5.4.2. The IP address 127.0.0.1 is special because it is used by all computers as their own special-purpose IPv4 address but, however, does not allow the computers to communicate with devises using this address (Mitchell, 2019).

5.4.3. The image below shows the result of executing the command lynx 127.0.0.1/test.html to view the HTML file in lynx.



The image shows that the IP address for the virtual machine is 192.168.31.161. This is being shown in a browswe of the windows system.

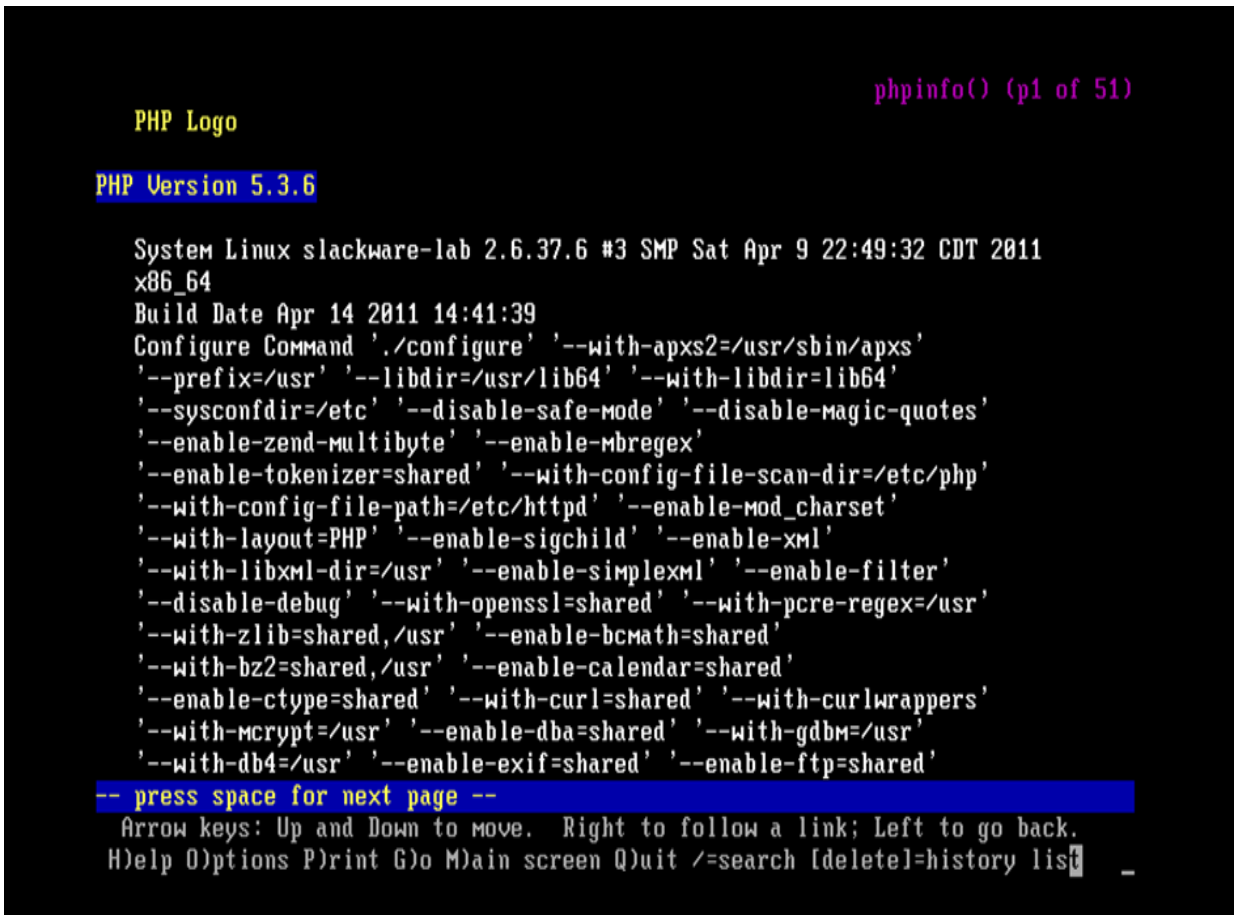
This is the NSE Apache Lab test page

Under construction

5.5. Creating and Viewing PHP Files Using the Terminal Interface

5.5.1. PHPinfo is a function of PHP which is responsible for returning compiled information concerning any PHP environment on your server. The information returned includes data on the PHP version, the PHP environment on the server, master and local values of configuration options, PHP compilation extentions and options as well as data on the PHP license and HTTP headers (xneelo, 2020).

5.5.2. The following image shows the results obtained from executing the command *lynx 127.0.0.1/nse.php* which allows the phpinfo() file to be loaded in lynx. The results show that the phpinfo() file was successfully loaded in lynx after executing the command.



5.6. Exploring and Adding An Entry to the Hosts File

The file /etc/hosts was opened via the vi text editor by using the command *vi /etc/hosts*. In the text editor, the line 127.0.0.1 www.example .com was added as pointed out in the image below. The information was saved using the command *:wq* before exiting the text editor.

```
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.  Just add the names, addresses
#               and any aliases to this file...
#
# By the way, Arnt Gulbrandsen <agulbra@nvg.unit.no> says that 127.0.0.1
# should NEVER be named with the name of the machine.  It causes problems
# for some (stupid) programs, irc and reputedly talk. :^)
#

# For loopbacking.
127.0.0.1               localhost
127.0.0.1               Bry021.bryant Bry021
127.0.0.1               www.example.com
# End of hosts.
```

The command *lynx /etc/hosts* was then executed and the results obtained are shown in the image below.

```
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.  Just add the names, addresses
#               and any aliases to this file...
#
# By the way, Arnt Gulbrandsen <agulbra@nvg.unit.no> says that 127.0.0.1
# should NEVER be named with the name of the machine.  It causes problems
# for some (stupid) programs, irc and reputedly talk. :^)
#

# For loopbacking.
127.0.0.1               localhost
127.0.0.1               Bry021.bryant Bry021
127.0.0.1               www.example.com
# End of hosts.


Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

6. Network Traffic Analysis

6.1. Creating a bash script called ping.sh

A bash script called ping.sh was created through the nano text editor by using the command *nano new_file*. The following image displays the contents of the ping.sh bash script. To save the script as ping.sh *ctrl + O* was executed and to exit the script *ctrl + X* was executed (Gentoo Linux, 2019).

The command *ls -l ping.sh* was executed to check if the script ping.sh was executable and the results showed that there were no execute permissions for users on the system on this script. To add these permissions the command *chmod a+x ping.sh* was executed as demonstrated below.

```
root@Bry021:~# bash ping.sh
ping.sh: line 3: syntax error near unexpected token 'newline'
ping.sh: line 3: '<html>'
root@Bry021:~# ls -l ping.sh
-rw-r--r-- 1 root root 151 Nov  3 10:48 ping.sh
root@Bry021:~# chmod a+x ping.sh
root@Bry021:~# ls -l ping.sh
-rwxr-xr-x 1 root root 151 Nov  3 10:48 ping.sh*
root@Bry021:~# bash ping.sh_
```

The script ping.sh was run by using the command *./ping.sh*. The network 192.168.31.0 was also used to test the script by connecting it to the virtual Slackware machine and the results obtained are shown in the image below.

```
{192.168.31.2} is alive
{192.168.31.161} is alive
```

7. Further Unix Tools

7.1. User and System Information

7.1.1. The command *last b* was executed to find out the number failed, or unsuccessful login attempts, and the results show that there were none in the last 48 hours.

```
root@Bry021:~# last b

wtmp begins Fri Mar  2 14:35:53 2012
root@Bry021:~#
```

7.1.2. The image below shows the number of system reboots in the last 48 hours. The command *last reboot* was executed, and the results below show that six system reboots have occurred.

```
root@Bry021:~# last reboot
reboot   system boot  2.6.37.6-smp     Thu Oct 31 12:49         (2+22:38)
reboot   system boot  2.6.37.6-smp     Fri Mar  2 15:24          (00:44)
reboot   system boot  2.6.37.6-smp     Fri Mar  2 15:16          (00:02)
reboot   system boot  2.6.37.6-smp     Fri Mar  2 15:12          (00:02)
reboot   system boot  2.6.37.6-smp     Fri Mar  2 14:43          (00:27)
reboot   system boot  2.6.37.6-smp     Fri Mar  2 14:35          (00:07)

wtmp begins Fri Mar  2 14:35:53 2012
root@Bry021:~#
```

7.2. Symbolic and Hard Links

7.2.1. The command *cd unixstuff* was used to navigate into the unixstuff directory. A file ~/unixstuff/extra_file was created by using the command *touch extra_file* and a symlink ~/unixstuff/links/extra_file_link was created by using the command *mkdir links*. The two were linked by using the command *ln -s extra_file links/extra_file_link* and to see whether the extra_file file and links directory were successfully created the command *ls -l* was executed in the unixstuff directory.

```
root@Bry021:~# ls
Desktop/    Pictures/   hello.sh     new_file   test.sh*
Documents/  Public/     internal     new_file1  unistuff/
Downloads/  Templates/  message.txt  ping.sh*   unixstuff/
Music/      Videos/     ncs/         test.html
root@Bry021:~# cd unixstuff
root@Bry021:~/unixstuff# ls
backups/  copy_function.txt  list1  list2  new_file*  new_file.bak*
root@Bry021:~/unixstuff# touch extra_file
root@Bry021:~/unixstuff# mkdir links
root@Bry021:~/unixstuff# ln -s extra_file links/extra_file_link
-su: ln: command not found
root@Bry021:~/unixstuff# ln -s extra_file links/extra_file_link
root@Bry021:~/unixstuff# ls -l
total 36
drwxr--r-- 2 root root 4096 Nov  1 23:07 backups/
-rw-r--r-- 1 root root 9762 Nov  2 06:12 copy_function.txt
-rw-r--r-- 1 root root    0 Nov  3 11:39 extra_file
drwxr-xr-x 2 root root 4096 Nov  3 11:40 links/
-rw-r--r-- 1 root root   21 Nov  2 05:54 list1
-rw-r--r-- 1 root root   32 Nov  2 05:56 list2
-rwxr-xr-x 1 root root   39 Oct 31 15:40 new_file*
-rwxr-xr-x 1 root root   39 Nov  2 04:38 new_file.bak*
root@Bry021:~/unixstuff#
```

7.2.2. A file extra_file was opened using the vi text editor and contents were added to this file as shown below.

```
 Hi what is your name?
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
writing extra_file                                        1,22  Command
```

The *cat ~/unixstuff/links/extra_file_link* command was executed to try and view the added contents to the extra_file_link but this was unsuccessful. The command *ls extra_file* was then used to see if the file exists and the results show that the file exists. Another command *ls links* was also executed to see if the extra_file_link and the results show that it exists. If the execution had been successful the added contents should have been printed on the console window.

```
root@Bry021:~/unixstuff# cat ~/unixstuff/links/extra_file_link
cat: /root/unixstuff/links/extra_file_link: No such file or directory
root@Bry021:~/unixstuff# ~/unixstuff/links$ cat ../links
-su: /root/unixstuff/links$: No such file or directory
root@Bry021:~/unixstuff# chmod a+x ~/unixstuff/links/extra_file_link
chmod: cannot operate on dangling symlink '/root/unixstuff/links/extra_file_link
'
root@Bry021:~/unixstuff# ls extra_file
extra_file
root@Bry021:~/unixstuff# ls links
extra_file_link@
root@Bry021:~/unixstuff# _
```

The extra_file was successfully moved into the backups directory as shown in the image below using the *mv extra_file backups* command. To check if the file had been successfully moved into the backups directory the command *cat extra_file_link* was executed, and the message returned showed that the file had been successfully moved.

```
root@Bry021:~/unixstuff# mv extra_file backups/
root@Bry021:~/unixstuff# cd links/
root@Bry021:~/unixstuff/links# cat extra_file_link
cat: extra_file_link: No such file or directory
root@Bry021:~/unixstuff/links# ls -l
total 0
lrwxrwxrwx 1 root root 10 Nov  3 11:40 extra_file_link -> extra_file
root@Bry021:~/unixstuff/links# _
```

The file extra_file was moved back into the links directory by executing the command *mv ../backups/extra_file ../ as shown below*.

```
oot@Bry021:~/unixstuff/links# mv ../backups/extra_file ../
oot@Bry021:~/unixstuff/links# ls -l
otal 0
rwxrwxrwx 1 root root 10 Nov  3 11:40 extra_file_link -> extra_file
oot@Bry021:~/unixstuff/links# cat extra_file_link
```

The command *rm extra_file_link* was executed to delete extra_file_link. The result obtained shows the message or text of the original file which has remained the same.

```
root@Bry021:~/unixstuff/links# rm extra_file_link
root@Bry021:~/unixstuff/links# cat ../extra_file
 Hi what is your name?
root@Bry021:~/unixstuff/links#
```

The file extra_file_link was recreated using the command *ls -s ../extra_file extra_file_link* and the command *rm extra_file* was used to delete the file again. The command *../extra_file* was executed to view the file, but the results obtained show that the file no longer existed and did not have any content.

```
root@Bry021:~/unixstuff/links# ln -s ../extra_file extra_file_link
root@Bry021:~/unixstuff/links# cat extra file link
cat: extra: No such file or directory
cat: file: No such file or directory
cat: link: No such file or directory
root@Bry021:~/unixstuff/links# rm ../extra_file
root@Bry021:~/unixstuff/links# cat ../extra_file
cat: ../extra_file: No such file or directory
root@Bry021:~/unixstuff/links# cat extra_file-link
cat: extra_file-link: No such file or directory
root@Bry021:~/unixstuff/links#
```

A hard link is a mirror copy of an original file while a symbolic or soft link is an actual link to an original file. The differences between a hard link and a symbolic link are stated in the table below.

Table 3 Differences between hard link and symbolic link

Hard Link	**Symbolic Link**
Does not have the ability to cross file system boundaries	Has the ability to cross the file system.
Contains the same permissions and inode number of the original file.	Contains different file permissions and inode number compared to the original file.
Inhibits the linking between different directories.	Allows one to link between different directories.
If the permissions of a source file are changed, the permissions will be updated.	Does not update permissions
Consists of the actual contents of the original file, so that you still can view the contents, even if the original file moved or removed	Holds only one path of the original file and not the contents.

SK, 2019.

References

1. ARCLAB, 2020. *A list of SMTP and POP3 server knowledge base // email*. [online] Available at: https://www.arclab.com/en/kb/email/list-of-smtp-and-pop3-servers-mailserver-list.html [Assessed 11/12/2019].
2. Aol.com, 2019. *What is the difference between POP3 and IMAP?.* [online] Available at: https://help.aol.co.uk/articles/what-is-the-difference-between-pop3-and-imap?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvLnVrLw&guce_referrer_sig=AQAAABAlL_iOus7JrFzd-4FD7szYEh0w-vL7W_F0DYYv1eaJc86r215GkBGyFnpeTI6H3KkksOjkUr-mouNfQAfvOugjepFErlxoh6Id0K6tYqxbvjWykoN8zto0t_oZnPR7KTCNBmLT6VDqfN9b92ReQNC1WlrR-1KfrEwMWFgcuTa3 [Assessed 04/12/2019].
3. Bayden, I., 2017. *Linux mail and mailx commands tutorial with examples and send email from command line*. [online] Available at: https://www.ibm.com/support/pages/var-filesystem-full-due-varspoolmail [Assessed 26/12/2019].
4. Blum, R., Bresnahan, C., 2o15. *Linux Command Line and Shell Scripting Bible*. 3rd ed. Indiana: John and Wiley Sons.
5. Buzdar, 2019. *How to comment out lines in configuration files on Linux*. [online] Available at: https://vitux.com/how-to-comment-out-lines-in-configuration-files-on-linux/ [Assessed 28/12/2019].
6. Canepa, G., 2016. *Find Top Running Processes by Highest Memory and CPU Usage in Linux*. [online] Tecmint: Linux Howtos, Tutorials & Guides. Available at: https://www.tecmint.com/find-linux-processes-memory-ram-cpu-usage/.
7. Cezar, A., 2018. *How to Synchronize Time with NTP in Linux*. [online] Tecmint: Linux Howtos, Tutorials & Guides. Available at: https://www.tecmint.com/synchronize-time-with-ntp-in-linux/ [Accessed 06/11/2019].
8. Chauhun, A., 2018. *Introduction to PGP encryption and decryption*. [online] Available at: https://developer.rackspace.com/blog/introduction-to-pgp-encryption-and-decryption/ [Assessed, 04/12/2019].
9. ComputerNetworkingNotes, 2020. *ps aux command and ps command explained*. [online] Available at: https://www.computernetworkingnotes.com/linux-tutorials/ps-aux-command-and-ps-command-explained.html [Assessed 30/12/2019].
10. Coulter, F., 2014. *Why 777 folder permissions are a security risk*. [online] Available at: https://www.spiralscripts.co.uk/Blog/why-777-folder-permissions-are-a-security-risk.html [Assessed 04/11/2019].
11. EDUCBA, 2019: *4 Important Shell Script Types for Linux Newbies (Helpful)*. [online] Available at: https://www.educba.com/shell-scripting-in-linux/ [Assessed 04/11/2019].
12. Fancher, P, J., 2018. *How to Change File Ownership & Groups in Linux*. [online] Available at: https://www.hostingadvice.com/how-to/change-file-ownershipgroups-linux/ [Assessed 12/11/2019].
13. Gen too Linux, 2019. *Nano/Basics Guide*. [online]. Available at: https://wiki.gentoo.org/wiki/Nano/Basics_Guide [Assessed 04/01/2010].

14. Henry-Stocker, S., 2017. *Examining network connections on Linux systems*. [online] Available at https://www.networkworld.com/article/3230519/examining-network-connections-on-linux-systems.html [Assessed 14/11/2019].
15. Henry-Stocker, S., 2019. *How to send email from the Linux command line.* [online] Available at: https://www.networkworld.com/article/3402027/how-to-send-email-from-the-linux-command-line.html [Assessed 24/12/2019].
16. IBM, 2019. *Creating and managing groups on Linux*. [online] Available at: https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.sec.doc/q011110_.htm [Assessed 12/11/2019].
17. IBM, 2018. */var Filesystem Is Full (due to /var/spool/mail/*).* [online] Available at: https://www.ibm.com/support/pages/var-filesystem-full-due-varspoolmail [Assessed 23/12/2019].
18. Kerravala, Z., 2018. *DHCP defined and how it works*. [online] Available at: https://www.networkworld.com/article/3299438/dhcp-defined-and-how-it-works.html [Assessed 12/11/2019].
19. Kyrnin, J., 2019. *Understanding the Index.html Page on a Website*. [online] Available at: https://www.lifewire.com/index-html-page-3466505 [Assessed 30/12/2019].
20. Lithmee, 2018. *Difference Between GUI and CLI*. [online] Available at: https://pediaa.com/difference-between-gui-and-cli/ [Assessed 01/01/2010].
21. Linuxize, 2019. *How to Create Groups in Linux (groupadd Command*). [online] Available at: https://linuxize.com/post/how-to-create-groups-in-linux/ [Assessed 12/11/2019].
22. McDonald, 2012. *The "chmod 777" trap: How and why to avoid it.* [online] Available at: https://www.anchor.com.au/blog/2012/09/the-chmod-777-trap-how-and-why-to-avoid-it/ [Assessed 05/11/2019].
23. McDonnell, S., 2020. *How to Switch Users in a Linux Shell*. [online] Available at: https://www.techwalla.com/articles/how-to-switch-users-in-a-linux-shell [Assessed 12/11/2019].
24. MDLog:/sysadmin, 2007. *HowTo Disable a User Account in Linux.* [online] Octopress. Available at: http://www.ducea.com/2007/12/05/howto-disable-a-user-account-in-linux/ [Assessed 08/11/2019].
25. McKinnon, J., 2017. *Understanding File Permissions and Using Them to Secure Your Site.* [online] Available at: https://premium.wpmudev.org/blog/understanding-file-permissions/ [Assessed 07/11/2019].
26. Mitchell, B., 2019. *127.0.0.1 IP Address Explained*. [online] Available at: https://www.lifewire.com/network-computer-special-ip-address-818385 [Assessed 03/01/2020].
27. Name com, 2019. *Understanding the difference between POP3 and IMAP*. [online] Available at: https://www.name.com/support/articles/205935497-Understanding-the-difference-between-POP-and-IMAP [Assessed 04/12/2019].
28. NixCraft, 2006. *Howto: Linux configure the Mouse at a text based terminal for copy and paste operation*. [online] Available at https://www.cyberciti.biz/tips/howto-linux-configure-the-mouse-at-a-text-based-terminal-for-copy-and-paste-operation.html [Assessed 17/1/2019].

29. Nooning, T., 2003. Stay on top of your Linux system by managing processes. [online] Available at https://www.techrepublic.com/article/stay-on-top-of-your-linux-system-by-managing-processes/ [Assessed 13/11/2019].
30. RapidTables, 2019. *ls command in Linux/Unix*. [online] Available at: https://www.rapidtables.com/code/linux/ls.html [Assessed 09/11/2019].
31. RedHat, 2020. *2.2.8. Securing Sendmail.* [online] Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-server_security-securing_sendmail [Assessed 17/12/2019].
32. Saive, R., 2012. *15 Basic 'ls' Command Examples in Linux*. [online] Tecmint: Linux Howtos, Tutorials & Guides. Available at: https://www.tecmint.com/15-basic-ls-command-examples-in-linux/ [Accessed 08/11/2019].
33. Saive, R., 2014. *The Complete Guide to "useradd" Command in Linux – 15 Practical Examples*. {online] Tecmint: : Linux Howtos, Tutorials & Guides. Available at: https://www.tecmint.com/add-users-in-linux/ {Assessed 07/11/2019].
34. SimplyWebHosting, 2014. What is mail spool?. [online] Available at: http://kb.simplywebhosting.com/idx/9/089/article/ [Assessed 20/12/2019].
35. SSH.COM, 2019. *Telnet - and SSH as a Secure Alternative*. [online] Available at: https://www.ssh.com/ssh/telnet [Assessed 25/11/2019].
36. StackExchange, 2019. *Kworker, what is it and why is it hogging so much CPU*?. [online] Available at https://askubuntu.com/questions/33640/kworker-what-is-it-and-why-is-it-hogging-so-much-cpu [Assessed 16/11/2019].
37. StackExchange, 2019. *What is the role of hald-addon-stor?.* [online] Available at https://www.experts-exchange.com/questions/26626460/What-is-the-role-of-hald-addon-stor.html?redirect=/questions/26626460/What-is-the-role-of-hald-addon-stor.html&offering=3035#_=_ [Assessed 17/11/2019].
38. Surendra, A., 2016. *Linux directory structure explained:/etc folder*. [online] The Linux Foundation. Available at https://www.linux.com/tutorials/linux-directory-structure-explainedetc-folder/ [Accessed 06/11/2019).
39. SureSwift Content, 2014. *Why Should You Put Linux In Your Computer*. [online] Available at: https://www.unixmen.com/put-linux-computer/ [Accessed 03/01/2020].
Ubuntu documentation, 2020. *HTTPD - Apache2 Web Server*. [online] Available at: https://help.ubuntu.com/lts/serverguide/httpd.html [Assessed 29/12/2019].
40. SK, 2019. Link In Linux With Examples. [online] Available at: https://www.ostechnix.com/explaining-soft-link-and-hard-link-in-linux-with-examples/ [Assessed 07/01/2020].
41. Vincent, D., 2016. What's the difference between FTPS, SFTP or FTP over SSH. [online] Available at: https://blog.devolutions.net/2016/05/ftps-sftp-or-ftp-over-ssh]Assessed 14/11/2019].
42. Xneelo, 2020. What is PHPinfo and how can I run it?. [online] Available at: https://xneelo.co.za/help-centre/website/what-is-phpinfo-and-how-can-i-run-it/ [Assessed 02/01/2020]