**Introduction to Linux Command**

The following steps will guide you through the most common Linux commands.

**1. Connecting and disconnecting**

**1.1 Connecting**

If you are using **windows**, open Xming and double-click *putty.exe*. Load a saved session in putty and open.

If you are using **Mac**, please open the XQuartz, type in “ssh –X username@10.0.106.245”.

**1.2 Interacting with the Shell**

Type a command (ls) at the prompt and press ENTER.

Example user@is437-1:~$ ls

**1.3 Get out (do it at the end of the exercise)**

The shell is killed by “exit” or CTRL-D

Example user@is437-1:~$ exit

**2. Passwords**

After you login, please change your password using **passwd** command

Example user@is437-1:~$ passwd

Changing password for user.

(current) UNIX password:

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

**3. Directories and files**

To check the directory name in which you are now (print working directory), use the **pwd** command. We call this directory (into which you get when you log in) your “home directory”.

Example user@is437-1:~$ pwd

/home/student/user

To make a new directory, use the **mkdir** command.

Example user@is437-1:~$ mkdir is437

To change your working directory, use the cd command.

Example user@is437-1:~$ cd is437

To create a new file, use the **touch** command.

Example user@is437-1:~$ touch test.txt

To edit this file (or create and the same time), use the **gedit** command. Gedit is a very convenient editor for primary users of linux. There are many other editors for Linux, such as vi, vim, emacs and so on.

Example user@is437-1:~$ gedit test.txt

To display the content of the file, use the **cat** (short for concatenation) command.

Example user@is437-1:~$ cat test.txt

If the contents to display are more than one page, you could use the **more**/**less** command for paging through text a screenful at a time. Less allows both forward and backward movement.

Example user@is437-1:~$ more test.txt

user@is437-1:~$ less test.txt

user@is437-1:~$ cat test.txt | less

Type Q to quit less.

To list the contents of a directory, use the **ls** command.

Example user@is437-1:~$ ls

To see all files and directories, including hidden ones use the **-a** flag with the ls command. Hidden files have a “.” in front of them.

Example user@is437-1:~$ ls -a

To copy contents of one file to another, use the **cp** command.

Example1 user@is437-1:~$ cp test.txt copytest.txt

user@is437-1:~$ cp test.txt test3.txt

user@is437-1:~$ ls

Example2 user@is437-1:~$ mkdir junk

user@is437-1:~$ mkdir junk2

user@is437-1:~$ cp test.txt ./junk/test2.txt

user@is437-1:~$ cd junk

user@is437-1:~$ ls

To go a level up from the current working directory, use **cd** command.

Example user@is437-1:~$ cd ..

To remove a file, use the **rm** command.

Example user@is437-1:~$ rm test3.txt

user@is437-1:~$ ls

To remove a directory, use the **-r** option with rm command. You can also use the **rmdir** command to remove an **empty** directory.

Example1 user@is437-1:~$ rm –r junk2

Example2 user@is437-1:~$ rmdir junk2

user@is437-1:~$ ls

A file can be renamed by moving it. This can be achieved by using the **mv** command.

Example user@is437-1:~$ mv copytest.txt test3.txt

user@is437-1:~$ ls

Both absolute and relative paths are used in Linux. The absolute path starts from the root directory, while the relative one starts from the current directory. The following two cd commands have the same effect.

Example user@is437-1:~$ pwd

/home/student/user/is437

user@is437-1:~$ cd /home/student/user/is437/junk

user@is437-1:~$ cd ./junk

To go back to the home directory, use the **cd** command.

Example user@is437-1:~$ cd

user@is437-1:~$ pwd

/home/student/user

**4. Getting Help**

Use the **man** command to get more information about a command – it is like using help in Windows.

Example user@is437-1:~$ man rmdir

You can also use **–h** or **--help** to get help.

Example user@is437-1:~$ rmdir --help

Another command powerful command is **apropos**, which is used to search commands related to the given keyword.

Example user@is437-1:~$ apropos directories

**5. Creating a tarball**

**TAR** (Tape Archive) command bundles files and sub-directories together and creates an archive (known as tar file or tarball) .

To create a tarball of all the files and sub-directories in the directory /home/student/user/is437, use c flag:

Example user@is437-1:~$ tar -cvf mytar.tar \*

To extract the contents of a tar file, use x flag:

Example user@is437-1:~$ tar -xvf mytar.tar

To compress the tar file as it is being created use z flag with c flag :

Example user@is437-1:~$ tar -cvzf mytar.tar.gz \*

To extract the contents of a compressed tar file use x flag:

Example user@is437-1:~$ tar -xvf mytar.tar.gz

user@is437-1:~$ tar -xvf mytar.tar.gz /home/student/user/junk

**6. File permissions**

Users typically perform the following operations on files:

1. Read files (using more, cat, etc.)

2. Write files (using >, vim, emacs, etc.)

3. Execute commands in a file (executables, etc.)

Accordingly, each file has three permissions – read, write and execute (rwx).

Person creating the file is the owner or user and can modify permissions as desired. That means owner can modify permissions on files to grant or revoke access to other users.

To check the file permissions use the **-l** flag with the ls command.

Example user@is437-1:~$ ls –l

total 12

drwxrwxr-x 2 user student 4096 Jul 17 19:48 junk

-rw-rw-r-- 1 user student 16 Jul 17 18:41 newtest.txt

-rw-rw-r-- 1 user student 4 Jul 17 18:43 test3.txt

Take the directory “./junk” as an example. This basically says, interpreting this from RIGHT to LEFT that the directory “junk” was created at 19:48 on July 17 and is 4096 bytes large. It belongs to the group “student”. It belongs to “user” in particular and this inode is linked by 2 filenames **.** Then, it comes to the file permission symbols:

“d” means this is a directory.

“rwx” means the **owner** “user” has read, write and execute permissions.

The second “rwx” means the **group** “user” has permissions of read, write and execute.

“r-x” means **other users** have read and execute permissions.

chmod command is used to change permissions on a file.

To add specific permission use **chmod +**

To add write permission to all users use:

chmod a+w filename

To add read permission to only the users in your group use:

chmod g+r filename

To make a file executable and runnable by any user

chmod a+x myfile

To remove specific permission use **chmod –**

Add and remove permissions can be combined in a single step

chmod u+x,g+r,o-rwx filename

Instead of using alphabets **u, g, o for user, group, and others** we can use numbers to specify file permissions:

rwx = 111 = 7

rw- = 110 = 6

r-x = 101 = 5

r-- = 100 = 4

-wx = 011 = 3

-w- = 010 = 2

--x = 001 = 1

--- = 000 = 0

Note that:

chmod go+rx filename = chmod 755 filename

**7. Other commands**

Previously executed commands in a shell can be viewed by using the **history** command.

Example user@is437-1:~$ history

“**>**” symbol can be used to redirect the output to a file or a utility.

Example user@is437-1:~$ ls > ./is437/myContent.txt

**grep** is used to search text or the given files for line containing a match to the given strings or words. Search a keyword in a file, using grep <word> <filename> or cat <filename> | grep <word>. You can force grep to ignore word case i.e. match user, User, USER and all other combination with the -i option:

Example user@is437-1:~$ grep user /etc/passwd

user@is437-1:~$ man rmdir | grep direcotries

user@is437-1:~$ cat /etc/passwd | grep user

user@is437-1:~$ grep -i “user” /etc/passwd

**8. Useful links**

More common Linux command: <http://www.pixelbeat.org/cmdline.html>

<http://ss64.com/bash/>

A Tutorial to Emacs: <http://www2.lib.uchicago.edu/keith/tcl-course/emacs-tutorial.html>

A Tutorial to Vim: http://www.openvim.com/tutorial.html