COMP4035 – Systems and Networks - Unix Lab 1

The purpose of this lab is to familiarise you with the basic UNIX commands for working with files and filesystems.

Login and Environment Setup:

You can do this lab either on your own Unix system (e.g. if you have Linux installed or are using Unix on a Mac) or you can connect to the school fileserver **mersey** by downloading and installing the VMWare software. Instructions for doing this can be found in the notes for Section 4 (Unix and Command Line Interfaces).

UNIX Filesystem Exercises:

Try working through the following exercises.

more

- Read/scan the man page for **more** with the command:

  man more

- As you are reading, notice the "more" prompt at the bottom of the page.
- Try pressing the return key - what happens?
- Try pressing the space bar once - what happens?
- Type the letter **b** - what happens?
- Use the search forward feature to find the word "environment" by entering the command:

  /environment

- **more** will continue until the end of the file is reached or until you type **q** for quit. Try typing **q** to quit.

ls

- Read/scan the man page for **ls** with the command:

  man ls

- Use **ls** without any arguments to display this directory's contents. How many files do you see?
- Now use **ls** with the **-a** option. How many files do you see this time? Notice that the "new" files all begin with a "dot", which indicates they are "hidden" files.

  ls -a

- The following command is useful for distinguishing between directories, ordinary files, and executable files. Notice how its output differs from **ls** without arguments.

  ls -F

- Use the command **ls -lg** to obtain a "long" listing of your files. Sample output from this command and an explanation of the information it provides appears below.

```
-rwxr-xr-x 1 jsmith staff 43 Mar 23 18:14 prog1
-rw-r--r-- 1 jsmith staff 10030 Mar 22 20:41 sample.f
drwxr-sr-x 2 jsmith staff 512 Mar 23 18:07 subdir1
drwxr-sr-x 2 jsmith staff 512 Mar 23 18:06 subdir2
drwxr-sr-x 2 jsmith staff 512 Mar 23 18:06 subdir3
```

1 2 3 4 5 6 7

1 = access modes/permissions
2 = number of links
3 = owner
4 = group
5 = size (in bytes)
6 = date/time of last modification
7 = name of file

- Recursive listings can be very useful. Try both of the commands below. What does the output tell you?

  ls -R
  ls -Rl

- Try three options together:

  ls -lFa

mkdir

- Read/scan the man page for **mkdir** with the command:

```
man mkdir
```

- Make a new directory called SAN:

```
mkdir SAN
```

- Change into this new directory using the **cd** command below. Create a new directory within this using the **mkdir** command called unixlab1:

```
cd SAN
mkdir unixlab1
```

  List your directory after the command completes to prove that it worked. Change to this new unixlab1 directory and create a subdirectory called newdir:

```
cd ~/SAN/unixlab1
mkdir newdir
ls
```

- Now create some additional subdirectories within your newdir. List newdir after the command completes to prove that it worked:

```
mkdir newdir/sub1 newdir/sub2 newdir/sub3
ls newdir
```

- Try to create a directory in a location where you don't have permission. What happens?

```
mkdir /etc/mydir
```

cd

- Read/scan the man page for **cd** with the command:

```
man cd
```

- Change to your SAN directory:

```
cd ~/SAN
```

- Change to a subdirectory within your SAN directory and list its contents:

```
cd ~/SAN/unixlab1/newdir
ls
```

- Go up one level to the current directory's parent directory and list the contents:

```
cd ..
ls
```

- Change to the root (top-most) directory and list the contents:

```
cd /
ls
```

- Change to another person's directory and list the contents:

```
cd ~username
ls
```

- Change to another one of your subdirectories and list the contents:

```
cd ~/SAN/unixlab1/newdir/sub1
ls
```

rmdir

- Read/scan the man page for **rmdir** with the command:

```
man rmdir
```

- First make sure you are in the unixlab1 directory. Then try to remove the newdir directory. What happens?

```
cd ~/SAN/unixlab1
rmdir newdir
```

- Recursively list the contents of newdir. Notice that its subdirectories are all empty. Remove all of the empty subdirectories within newdir and then list newdir again to confirm that they were removed:

```
ls -R newdir
rmdir newdir/*
ls -R newdir
```

- Finally, remove the empty newdir directory:

```
rmdir newdir
```

- Make sure you are in your SAN directory.
- Using a text editor create 3 or 4 text files with miscellaneous information in them that you can work on in this lab. Call them test1, test2, test3 and test4. (To create an

empty file called **test** from the command line you can use the command: **touch test** which you can then edit if you prefer to do it this way.)

cat

- Read/scan the man page for **cat** with the command:

```
man cat
```

- Change directory to your SAN directory. Use this command to display the contents of a file: What happens?

```
cat test1
```

- Now try this command notice the difference. How many lines are in the file?

```
cat -n test1
```

- The **cat** command is more often used for purposes other than just displaying a file, for example concatenating the contents of two files:

cat test1 ***- first, show file1***
cat test2 ***- then, show file2***
cat test1 test2 > newtest ***- now do the actual concatenate***
cat newtest ***- finally, show the result***

cp

- Read/scan the man page for **cp** with the command:

```
man cp
```

- Copy an existing file in your current directory to another file in the current directory and then list your directory to prove that it was done:

```
cp test1 test1b
ls
```

- Use the copy command with the "inquire" option. What happens?

```
cp -i test1 test1b
```

- Make two new directories, subdir1 and subdir4. Use the recursive option to copy an entire subdirectory to a new subdirectory and then list both directories to prove that it worked:

```
cp -R subdir1 subdir4
ls subdir1 subdir4
```

- The copy command accepts "wildcard" characters. Try the command below. What did it do? List the subdir1 subdirectory to find out.

```
cp test* subdir1
ls subdir1
```

mv

- Read/scan the man page for **mv** with the command:

```
man mv
```

- The **mv** command can be used for renaming files. Try this command and then list your files to prove that the command worked:

```
mv test3 newtest3
ls
```

- **mv** can be used to rename directories also. Try this command and then list your files to prove that the command worked:

```
mv subdir4 dir4
ls
```

- The **mv** command is also used for moving files. Use the command below to move newtest3 into a new location, then list your files to prove that the command worked:

```
mv newtest3 unixlab1
ls
ls unixlab1
```

- The **mv** command can also be used for moving directories. Use the command below to move dir4 and subdir1 to unixlab1, then list your files in your SAN directory and in unixlab1 to prove that the command worked:

```
mv dir4 unixlab1
mv subdir1 unixlab1
ls
ls unixlab1
```

- Like many other UNIX commands, **mv** recognizes wildcard characters. Try the command below and then list your files to prove that the command worked:

```
mv *test* unixlab1
ls
ls unixlab1
```

rm

- Read/scan the man page for **rm** with the command:

```
man rm
```

- Change directory to unixlab1. Use the **rm** command to delete a file. List your directory after the command completes.

```
cd ~/SAN/unixlab1
rm test4
ls
```

- **cd** to the **subdir1** subdirectory. List the directory to view its contents. Then use the "*" wildcard to remove all of the files. NOTE: using **rm** in this manner can be dangerous! If you are in the wrong directory you'll remove files you didn't mean to remove. You may want to use the **-i** option to protect yourself from accidents.

```
cd subdir1
ls
rm -i *
ls
```

- Get out of the **subdir1** subdirectory by using the command **cd ..** Now try to use **rm** to remove a directory. What happens?

```
cd ..
rm subdir1
```

- This time, include the **-r** option when you try to remove a directory. What happens?

```
rm -r subdir1
ls
```

file

- Read/scan the man page for **file** with the command:

```
man file
```

- Change directory to unixlab1. Use the **file** command to determine a file's type:

```
cd ~/SAN/unixlab1
file test1
```

- Now try it with a directory:

```
file dir4
```

- Finally, try it with a wildcard character:

```
file *
```

find

- Read/scan the man page for **find** with the command:

```
man find
```

- Use the find command to find the file newtest3.

```
find . -name newtest3 -print
```

- Now use the find command to find all files with "test" as part of their name. Don't forget to put the wildcard specification in quotes - it won't work otherwise:

```
find . -name '*test*' -print
```

- Try to find only directories with "file" as part of their name. Are there any?

```
find . -name 'file*' -type d -print
```

pwd

- Read/scan the man page for **pwd** with the command:

```
man pwd
```

- Issue the **pwd** command to display the name of your current working directory:

```
pwd
```

- Change to several other directories and issue the **pwd** command between each change. Notice the different outputs:

```
cd dir4
pwd
cd /usr/bin
pwd
cd ~/Mail
pwd
cd ~/SAN/unixlab1
pwd
du
```

Some questions for you to investigate:

Q - How much disk space do my files take up? Type **du** to find out.

Q - What does **du** by itself do?

Q - What does **du -s** do?

Q - What does **du -s *** do?

Q - Which is most useful toward helping you determine which files take up the most disk space?