

Introduction to UNIX - Marking Scheme

	Registration Number	Surname	Forename	% Contribution (Out of 100)	
Student 1	000839875	Turon	Kacper	25%	
Student 2	000871936	Young	Jack	25%	
Student 3	000878481	Kallon	Kevin	25%	
Student 4	000874782	Basharat	Usman	25%	
Task				Mark	Grade
Task 1 - Access to UNIX using Putty (SSH)				30	
Task 2 - Access to UNIX using Xming				10	
Task 3 - Running Programs in UNIX				10	
Task 4 - UNIX Shell Programming				25	
Task 5 - Install Linux (Optional)				-	-
Task 6 - Reflection				25	
Total				100	

COMP1587 Communication Systems

Week 2 Laboratory

Introduction to UNIX

Usman Basharat - 000874782

Kevin Kallon - 000878481

Jack Young - 000871936

Kacper Turon - 000839875

Contents

Abstract.....	3
Introduction	3
Methods and Materials (or Equipment)	3
Equipment/Materials used:	3
Software used:	3
Commands used:.....	3
Experimental Procedure	4
Results.....	4
Task 1 – Access UNIX using Putty (SSH)	4
Task 2 – Access to UNIX using Xming.....	11
Task 3 – Running Programs in UNIX.....	13
Task 4 – UNIX Shell Programming.....	16
Script 1	16
Script 2	17
Script 3	18
Script 4	19
Script 5	21
Discussion	25
Task 6 – Reflection	25
Conclusion.....	27
References	28

Abstract

UNIX was one of the first operating systems to be written in a high-level programming language, namely C (Beal, 2015). The purpose of this laboratory was to get familiar with UNIX, this meant getting to grips with how it works, how UNIX works with a graphical interface and how to write shell programs etc. The key aspect of the laboratory was motivation. As a group, we felt that from Task 4, it tested our skills and ability to be motivated to find answers that may not necessarily be obvious at first glance. To solve this, as a group, we planned to get together at one of the unsupervised lab sessions to complete and understand Task 4. We all did our research in between the hours and with hard-work, we managed to complete all tasks. Of course, it was hard to do at first, but we knew how it was done once the first one was completed. The significant of this is that we wanted to complete all Tasks before handing in the final copy.

Introduction

This laboratory session was all about getting to know UNIX. In this report, we had six tasks to complete, one of which was an optional task. Each task, we had our own specific task set. For example, Usman did Task 2 and Kevin did Task 1 and so on. We had a guide that gives clear steps to help us learn UNIX as beginners. Once all these tasks were complete, we had to reflect on how it went. Find out where we struggled and how we solved it.

Methods and Materials (or Equipment)

Equipment/Materials used:

- Computer
- USB
- Guide (Greenwich, 2015)
- The Lab Report (Toronto, 2015)

Software used:

- Putty
- CMS Student Solaris Server (Xming)
- gedit
- Microsoft Word

Commands used:

In this laboratory, we understood the concept of using commands to execute instructions in UNIX. For example, some of the commands is the following:

- **CD** - We used is 'cd' which means to change directory. This means is that you can use the 'cd' command to replace the current file into another directory.
- **CP** - We used 'cp' to copy a file that we needed. For example, we had to copy the june3000.txt file.
- **LS** - We used 'ls' to show us the files that are currently in the directory. For example, we kept typing ls to show us what was inside the directory.
- **RM** - We used the remove command to remove a file that we created. For example, we removed poem2.txt by using this command.
- **WC** - We used WC to calculate how many words they were within the poem using this command.

Experimental Procedure

For this laboratory session, we followed a guide provided by the University of Greenwich that told us what to do step by step. All steps were followed; no extra steps were taken when doing the exercise that was based on the guide.

Results

Task 1 – Access UNIX using Putty (SSH)

Script started on 6 October 2015 13:27:32 BST

%cal 3000

3000

Jan	Feb	Mar
S M Tu W Th F S	S M Tu W Th F S	S M Tu W Th F S
1 2 3 4	1	1
5 6 7 8 9 10 11	2 3 4 5 6 7 8	2 3 4 5 6 7 8
12 13 14 15 16 17 18	9 10 11 12 13 14 15	9 10 11 12 13 14 15
19 20 21 22 23 24 25	16 17 18 19 20 21 22	16 17 18 19 20 21 22
26 27 28 29 30 31	23 24 25 26 27 28	23 24 25 26 27 28 29
	30 31	
Apr	May	Jun
S M Tu W Th F S	S M Tu W Th F S	S M Tu W Th F S
1 2 3 4 5	1 2 3 1 2 3 4 5 6 7	
6 7 8 9 10 11 12	4 5 6 7 8 9 10	8 9 10 11 12 13 14
13 14 15 16 17 18 19	11 12 13 14 15 16 17	15 16 17 18 19 20 21
20 21 22 23 24 25 26	18 19 20 21 22 23 24	22 23 24 25 26 27 28
27 28 29 30	25 26 27 28 29 30 31	29 30
Jul	Aug	Sep

S	M	Tu	W	Th	F	S	S	M	Tu	W	Th	F	S	S	M	Tu	W	Th	F	S
1	2	3	4	5	1	2	1	2	3	4	5	6								
6	7	8	9	10	11	12	3	4	5	6	7	8	9	7	8	9	10	11	12	13
13	14	15	16	17	18	19	10	11	12	13	14	15	16	14	15	16	17	18	19	20
20	21	22	23	24	25	26	17	18	19	20	21	22	23	21	22	23	24	25	26	27
27	28	29	30	31	24	25	26	27	28	29	30	28	29	30						
31																				

Oct	Nov	Dec																		
S	M	Tu	W	Th	F	S	S	M	Tu	W	Th	F	S	S	M	Tu	W	Th	F	S
1	2	3	4	1	1	2	3	4	5	6										
5	6	7	8	9	10	11	2	3	4	5	6	7	8	7	8	9	10	11	12	13
12	13	14	15	16	17	18	9	10	11	12	13	14	15	14	15	16	17	18	19	20
19	20	21	22	23	24	25	16	17	18	19	20	21	22	21	22	23	24	25	26	27
26	27	28	29	30	31	23	24	25	26	27	28	29	28	29	30	31				
30																				

%call[K 6 3000

June 3000

S	M	Tu	W	Th	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

%cal 9 1752 → Dates

September 1752

S	M	Tu	W	Th	F	S
---	---	----	---	----	---	---

1 2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

```
%cal 6 3000 > june3000[K0.txt
```

```
%ls
```

```
june3000.txt public_html/ task1
```

```
%cp june3000.txt june3k.txt
```

```
%ls
```

```
june3000.txt june3k.txt public_html/ task1
```

```
%ls -l
```

```
total 8
```

```
-rw----- 1 kk1350r student 125 Oct 6 13:28 june3000.txt
```

```
-rw----- 1 kk1350r student 125 Oct 6 13:29 june3k.txt
```

```
drwxr-xr-x 2 kk1350r student 2 Sep 23 10:21 public_html/
```

```
-rw----- 1 kk1350r student 0 Oct 6 13:27 task1
```

```
%cd/homwe[@ //homwe[Ke/sp02/unixfiles
```

```
/home/sp02/unixfiles
```

```
%ls -l
```

```
total 34
```

```
-rwxr--r-- 1 sp02 staff 47 Oct 8 2010 hello.c*
```

```
-rwxr--r-- 1 sp02 staff 58 Oct 1 2010 hello.c~*
```

```
-rwxr-xr-x 1 sp02 staff 752 Apr 22 2009 menu.ksh*
```

```
-rwxr--r-- 1 sp02 staff 502 Oct 8 2010 oddeven.c*
```

```

-rwxr--r-- 1 sp02 staff 555 Oct 1 2010 oddeven.c~*
-rwxr--r-- 1 sp02 staff 1302 Oct 8 2010 owen.txt*
-rwxr--r-- 1 sp02 staff 1305 Apr 15 2009 owen.txt~*
-rwxr--r-- 1 sp02 staff 1187 Apr 8 2009 poem.txt*
-rwxr-xr-x 1 sp02 staff 81 Apr 22 2009 sh1.ksh*
-rwxr-xr-x 1 sp02 staff 60 Apr 22 2009 sh2.ksh*
-rwxr-xr-x 1 sp02 staff 54 Apr 22 2009 sh3.ksh*
-rwxr-xr-x 1 sp02 staff 93 Apr 22 2009 sh4.ksh*
-rwxr-xr-x 1 sp02 staff 63 Apr 22 2009 users.ksh*

```

%cd

/home/kk1350r

%mkdir ccs

%cd ccs

/home/kk1350r/ccs

%cp /home/sp02/unixfiles/*.* .

%ls -l

total 34

```

-rwx----- 1 kk1350r student 47 Oct 6 13:31 hello.c*
-rwx----- 1 kk1350r student 58 Oct 6 13:31 hello.c~*
-rwx----- 1 kk1350r student 752 Oct 6 13:31 menu.ksh*
-rwx----- 1 kk1350r student 502 Oct 6 13:31 oddeven.c*
-rwx----- 1 kk1350r student 555 Oct 6 13:31 oddeven.c~*
-rwx----- 1 kk1350r student 1302 Oct 6 13:31 owen.txt*
-rwx----- 1 kk1350r student 1305 Oct 6 13:31 owen.txt~*
-rwx----- 1 kk1350r student 1187 Oct 6 13:31 poem.txt*
-rwx----- 1 kk1350r student 81 Oct 6 13:31 sh1.ksh*
-rwx----- 1 kk1350r student 60 Oct 6 13:31 sh2.ksh*
-rwx----- 1 kk1350r student 54 Oct 6 13:31 sh3.ksh*

```



```
-rwx----- 1 kk1350r student 93 Oct 6 13:31 sh4.ksh*  
-rwx----- 1 kk1350r student 63 Oct 6 13:31 users.ksh*  
%poem.txt
```

poem.txt: On: not found
poem.txt: Long: not found
poem.txt: That: not found
poem.txt: And: not found
poem.txt: By: not found
poem.txt: Flowing: not found
poem.txt: Four: not found
poem.txt: Overlook: not found
poem.txt: And: not found
poem.txt: The: not found
poem.txt: By: not found
poem.txt: By: not found
poem.txt: and: not found
poem.txt: Skimming: not found
poem.txt: But: not found
poem.txt: Or: not found
poem.txt: Or: not found
poem.txt: The: not found
poem.txt: Only: not found
poem.txt: In: not found
poem.txt: Hear: not found
poem.txt: From: not found
poem.txt: Down: not found
poem.txt: Lady: not found

```
%cat poem.txt
```



Poem written up.

On either side the river lie
Long fields of barley and of rye,

That clothe the wold and meet the sky;
And thro' the field the road runs by
 To many-towered Camelot;
And up and down the people go,
Gazing where the lilies blow
Round an island there below,
 The island of Shalott.

Willows whiten, aspens quiver,
Little breezes dusk and shiver
Thro' the wave that runs forever
By the island in the river
 Flowing down to Camelot.
Four gray walls and four grey towers,
Overlook a space of flowers,
And the silent isle embowers
 The Lady of Shalott.

By the margin, willow-veil'd,
Slide the heavy barges trail'd
By slow horses; and unhail'd
The shallop flitteth silken-sail'd
 Skimming down to Camelot:
But who hath seen her wave her hand?
Or at the casement seen her stand?
Or is she known in all the land,
 The Lady of Shalott.

Only reapers, reaping early
In among the bearded barley
Hear a song that echoes cheerly

From the river winding clearly,
Down to tower'd Camelot:
And by the moon the reaper weary,
Piling sheaves in uplands airy,
Listening, whispers "'Tis the fairy
Lady of Shalott".

Part 1 of "The Lady of Shalott"

by Alfred, Lord Tennyson.

```
%cat poem.txt > poem2.txt[K[K.txt
```

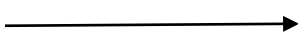
```
%ls -l
```

```
total 35
```

```
-rwx----- 1 kk1350r student 47 Oct 6 13:31 hello.c*  
-rwx----- 1 kk1350r student 58 Oct 6 13:31 hello.c~*  
-rwx----- 1 kk1350r student 752 Oct 6 13:31 menu.ksh*  
-rwx----- 1 kk1350r student 502 Oct 6 13:31 oddeven.c*  
-rwx----- 1 kk1350r student 555 Oct 6 13:31 oddeven.c~*  
-rwx----- 1 kk1350r student 1302 Oct 6 13:31 owen.txt*  
-rwx----- 1 kk1350r student 1305 Oct 6 13:31 owen.txt~*  
-rwx----- 1 kk1350r student 1187 Oct 6 13:31 poem.txt*  
-rw----- 1 kk1350r student 1187 Oct 6 13:32 poem2.txt  
-rwx----- 1 kk1350r student 81 Oct 6 13:31 sh1.ksh*  
-rwx----- 1 kk1350r student 60 Oct 6 13:31 sh2.ksh*  
-rwx----- 1 kk1350r student 54 Oct 6 13:31 sh3.ksh*  
-rwx----- 1 kk1350r student 93 Oct 6 13:31 sh4.ksh*  
-rwx----- 1 kk1350r student 63 Oct 6 13:31 users.ksh*
```

```
%cat poem.txt >> poem2.txt
```

```
%ls -l
```



Accessing file

total 35

```
-rwx----- 1 kk1350r student 47 Oct 6 13:31 hello.c*
-rwx----- 1 kk1350r student 58 Oct 6 13:31 hello.c~*
-rwx----- 1 kk1350r student 752 Oct 6 13:31 menu.ksh*
-rwx----- 1 kk1350r student 502 Oct 6 13:31 oddeven.c*
-rwx----- 1 kk1350r student 555 Oct 6 13:31 oddeven.c~*
-rwx----- 1 kk1350r student 1302 Oct 6 13:31 owen.txt*
-rwx----- 1 kk1350r student 1305 Oct 6 13:31 owen.txt~*
-rwx----- 1 kk1350r student 1187 Oct 6 13:31 poem.txt*
-rw----- 1 kk1350r student 2374 Oct 6 13:32 poem2.txt
-rwx----- 1 kk1350r student 81 Oct 6 13:31 sh1.ksh*
-rwx----- 1 kk1350r student 60 Oct 6 13:31 sh2.ksh*
-rwx----- 1 kk1350r student 54 Oct 6 13:31 sh3.ksh*
-rwx----- 1 kk1350r student 93 Oct 6 13:31 sh4.ksh*
-rwx----- 1 kk1350r student 63 Oct 6 13:31 users.ksh*
```

%more

Usage: more [-cdfllrsuw] [-lines] [+linenumber] [+/pattern] [filename ...].

%rm poem,[K2.txt

rm: remove regular file `poem2.txt'? y

Remove file

%exit

exit

script done on 6 October 2015 13:33:55 BST

Task 2 – Access to UNIX using Xming

On either side the river lie

Long fields of barley and of rye,

That clothe the wold and meet the sky;

And thro' the field the road runs by

To many-towered Camelot;
And up and down the people go,
Gazing where the lilies blow
Round an island there below,
The island of Onion.

Willows whiten, aspens quiver,
Little breezes dusk and shiver
Thro' the wave that runs forever
By the island in the river
Flowing down to Camelot.

Four gray walls and four grey towers,
Overlook a space of flowers,
And the silent isle embowers

The Lady of Onion.

By the margin, willow-veiled,

Slide the heavy barges trailed

By slow horses; and unhailed

The shallop flitteth silken-sailed

Skimming down to Camelot:

But who hath seen her wave her hand?

Or at the casement seen her stand?

Or is she known in all the land,

The Lady of Onion.

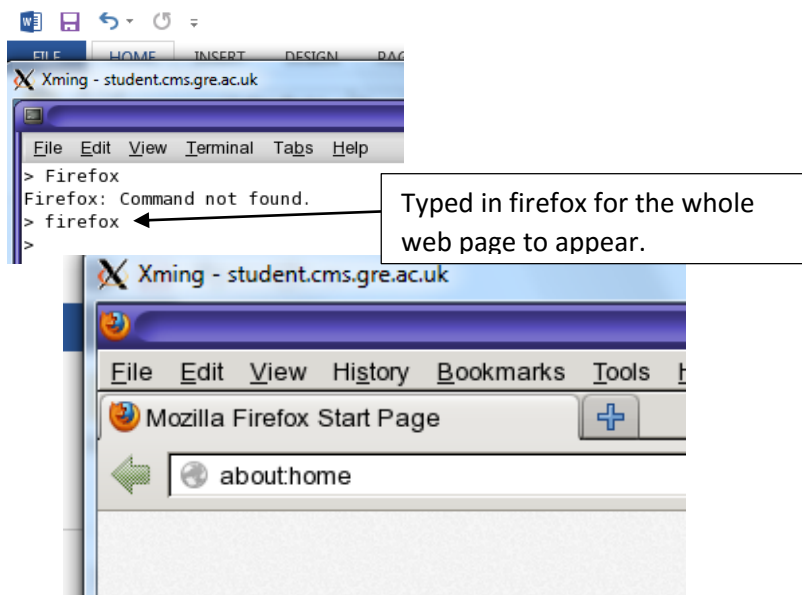
Only reapers, reaping early
In among the bearded barley
Hear a song that echoes cheerly
From the river winding clearly,
Down to towered Camelot:

These words are replaced
with Onion instead of Shalott
and ed instead of 'd

And by the moon the reaper weary,
Piling sheaves in uplands airy,
Listening, whispers "'Tis the fairy
Lady of Onion".

Part 1 of "The Lady of Onion"

by Alfred, Lord Tennyson.



Task 3 – Running Programs in UNIX

Script started on 6 October 2015 13:45:12 BST

```
%gcc hello.c -o g[khello.out
```

```
gcc: hello.c: No such file or directory
```

```
gcc: no input files
```

```
%cd ccs
```

```
/home/jy6244k/ccs
```

```
%ls -l
```

Accessing files to see where it is located.

```
total 42
```

```
-rwx----- 1 jy6244k student 47 Oct 6 13:12 hello.c*
```

```
-rwx----- 1 jy6244k student 58 Oct 6 13:12 hello.c~*
```

```

-rwx----- 1 jy6244k student 752 Oct 6 13:12 menu.ksh*
-rwx----- 1 jy6244k student 502 Oct 6 13:12 oddeven.c*
-rwx----- 1 jy6244k student 555 Oct 6 13:12 oddeven.c~*
-rwx----- 1 jy6244k student 1302 Oct 6 13:12 owen.txt*
-rwx----- 1 jy6244k student 1305 Oct 6 13:12 owen.txt~*
-rwx----- 1 jy6244k student 1179 Oct 6 13:32 poem.txt*
-rwx----- 1 jy6244k student 1177 Oct 6 13:30 poem.txt~*
-rw----- 1 jy6244k student 1177 Oct 6 13:31 poem3.txt
-rwx----- 1 jy6244k student 81 Oct 6 13:12 sh1.ksh*
-rwx----- 1 jy6244k student 60 Oct 6 13:12 sh2.ksh*
-rwx----- 1 jy6244k student 54 Oct 6 13:12 sh3.ksh*
-rwx----- 1 jy6244k student 93 Oct 6 13:12 sh4.ksh*
-rwx----- 1 jy6244k student 63 Oct 6 13:12 users.ksh*

```

```
%script task[K[K[K[K[K[K[K[K[Kgcc hello.c -o hellow[K.out
```

```
%./hello.out
```

```
Hello World
```



```
Hello World appears
```

```
%./hello.out[11D[13@gcc hello.c -o [9C[24Dls -l[K[5Dgcc hello.c -o hello.out
```

```
^[[A^[[A^[[A^[[B %gcc hello.c -o
```

```
hello.out[24D[13P./[9C[K[K[K[K[K[K[K[K[K[K./hello.out[K[K[K[K[K[K[K[K[K[K./hello.out
```

```
Hello Jack Young
```



```
Changed
within its C
program to
say Hello
```

```
%oddeven.c
```

```
oddeven.c: /*This: not found
```

```
oddeven.c: syntax error at line 10: `printf' unexpected
```

```
%ls -l
```

```
total 54
```

```
-rwx----- 1 jy6244k student 52 Oct 6 13:48 hello.c*
```

```

-rwx----- 1 jy6244k student  58 Oct  6 13:12 hello.c~*
-rwx----- 1 jy6244k student 5544 Oct  6 13:50 hello.out*
-rwx----- 1 jy6244k student  752 Oct  6 13:12 menu.ksh*
-rwx----- 1 jy6244k student  502 Oct  6 13:12 oddeven.c*
-rwx----- 1 jy6244k student  555 Oct  6 13:12 oddeven.c~*
-rwx----- 1 jy6244k student 1302 Oct  6 13:12 owen.txt*
-rwx----- 1 jy6244k student 1305 Oct  6 13:12 owen.txt~*
-rwx----- 1 jy6244k student 1179 Oct  6 13:32 poem.txt*
-rwx----- 1 jy6244k student 1177 Oct  6 13:30 poem.txt~*
-rw----- 1 jy6244k student 1177 Oct  6 13:31 poem3.txt
-rwx----- 1 jy6244k student  81 Oct  6 13:12 sh1.ksh*
-rwx----- 1 jy6244k student  60 Oct  6 13:12 sh2.ksh*
-rwx----- 1 jy6244k student  54 Oct  6 13:12 sh3.ksh*
-rwx----- 1 jy6244k student  93 Oct  6 13:12 sh4.ksh*
-rwx----- 1 jy6244k student  63 Oct  6 13:12 users.ksh*

```

%oddeven.x[Kc

oddeven.c: /*This: not found

oddeven.c: syntax error at line 10: `printf' unexpected


%gcc oddeven.c -o hello[K[K[K[Koddeven.out

%/[K./oddeven.out

enter your number, > 0 and < 32768 : 18

number is even

%exit

exit  Stop recording

script done on 6 October 2015 13:58:32 BST

Task 4 – UNIX Shell Programming

Script 1

```
#!/bin/ksh

echo 'Please enter a number'
read number
if [ $number == "1" ]
then
echo 'The value was ' $number
else
echo 'The value was not 1'
fi
```

Script started on Tue 06 Oct 2015 21:11:02 BST

[4mstulinux[24m:[1m~[0m> ./script1.ksh

Please enter a number

20

The value was not 1

[4mstulinux[24m:[1m~[0m> exit

exit

Script done on Tue 06 Oct 2015 21:11:29 BST

First of all we have to get from user a value so we use command *read number*. *number* stands for variable in our example after we have done that we have to check if this variable is equal to 1 so we use *if [\$number == "1"] then echo 'This value was ' \$number* which in this if would always be 1 and then *else echo 'This value was not 1'* every value here would not be equal to 1 and then we end up our code with *fi* to show that we finished with our if statement.

Script 2

```
#!/bin/ksh

echo 'Please enter the first number'
read number1
echo 'Please enter the second number'
read number2
echo 'Please enter the third number'
read number3

let sum=$number1+$number2+$number3
let average=sum/3
echo
echo "The average is: $average"
```

Script started on Tue 06 Oct 2015 21:14:19 BST

[4mstulinux[24m:[1m~[0m> ,.[K[K./script2.ksh

Please enter the first number

4


Please enter the second number

6

Please enter the third number

5

The average is: 5



Finding the value of the average is given. Out of 4-6, the average is 5.

[4mstulinux[24m:[1m~[0m> exit

Script done on Tue 06 Oct 2015 21:14:36 BST

In this exercise we repeat steps from first exercise but instead of getting one value from user we take 3 so we do *read number1 read number2 read number3* then we have to calculate the sum of those numbers so we use *let* function which is used for calculations. *let sum=\$number1+\$number2+\$number3* and then we have to calculate the average so we use again *let average=sum/3* and after that we just have to display our value *echo "The average is: \$average"*.

Script 3

```
#!/bin/ksh

echo 'Please enter the number of inputs'
read numberOfInputs
clear
InputSum=0;
for (( loop=1; loop<=numberOfInputs; loop++))
do
echo "Please enter the $loop number"
read Input
let InputSum=InputSum+Input

done

let InputAverage=$InputSum/numberOfInputs
echo "Number of inputs: $numberOfInputs. The average: $InputAverage"
```

Script started on Tue 06 Oct 2015 21:14:43 BST

[4mstulinux[24m:[1m~[0m> ,[K./script3.ksh

Please enter the number of inputs

5

[H[2JPlease enter the 1 number

10

Please enter the 2 number

15

Please enter the 3 number

4

Please enter the 4 number

5 6

Please enter the 5 number

34

Number of inputs: 5. The average: 13

[4mstulinux[24m:[1m~[0m> exit

Script done on Tue 06 Oct 2015 21:15:17 BST

Script 4

19 | Page

[23d^G(B[m Get Help (B[0;7m^O(B[m WriteOut (B[0;7m^R(B[m Read File (B[0;7m^Y(B[m Prev Page
(B[0;7m^K(B[m Cut Text (B[0;7m^C(B[m Cur Pos

[24d(B[0;7m^X(B[m Exit[14G(B[0;7m^J(B[m Justify (B[0;7m^W(B[m Where Is (B[0;7m^V(B[m Next
Page (B[0;7m^U(B[m UnCut Text(B[0;7m^T(B[m To Spell

[3dpoe[m[1;71H(B[0;7mModified[3;6H(B[m new poem 2

[22d(B[0;7mSave modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? [23;1H
Y(B[m Yes[K

[24d(B[0;7m N(B[m No [14G (B[0;7m^C(B[m Cancel[K[22;62H

(B[0;7mFile Name to Write: poem.txt

[23d^G(B[m Get Help[23;21H(B[0;7mM-D(B[m DOS Format[41G(B[0;7mM-A(B[m
Append[23;61H(B[0;7mM-B(B[m Backup File

[24d(B[0;7m^C(B[m Cancel[17G (B[0;7mM-M(B[m Mac Format[41G(B[0;7mM-P(B[m Prefix[22;29H

[23d[39;49m(B[mJ[1;71H(B[0;7m [22;32H(B[m1K (B[0;7m[Wrote 1 line
](B[m[K[24;80H[24;1H[?1049I

[?1I>[?1049h[?1h=

poem new poem 2

[7mpoem.txt (END)[27m[K

[K[?1I>[?1049I[4mstulinux[24m:[1m~[0m> exit

exit

Script done on Tue 06 Oct 2015 21:16:47 BST

We have to get the name of the file that user wants to edit. *read filename if [! -f filename];
then echo "file not found"* We check if the file exists with our if statement ! True if filename exists as
a non-directory -f would show us but we check if file doesn't exist using ! -f then we use *less*
\$filename to display the file in command line then *nano \$filename* to let the user edit the file and
after that we display it again with *nano \$filename* and close the if statement with *fi*.

Script 5

```
#!/bin/ksh

echo "Write a Poem menu"
echo
echo "1: Write a Poem"
echo "2: Word count the Poem"
echo "3: Display the Poem"
echo "4: Sort the output"
echo "5: Exit"
echo
echo "What would you like to do?"
read menuChoice
clear
case $menuChoice in
"1") nano ;;
"2")
echo "What is the file name?"
read filename
if [ ! -f $filename ]; then
    echo "File not found!"
else
    wc -w $filename
fi;;
"3") echo "What is the file name?"
read filename
if [ ! -f $filename ]; then
    echo "File not found!"
else
    clear
    echo "Which format would you like to use LESS or MORE"
    read format
    if [ $format=="MORE" ];
    then
        more $filename
    else
        less $filename
    fi
fi;;
"4") echo "What is the file name?"
read filename
if [ ! -f $filename ]; then
    echo "File not found!"
else
    filename_sorted=${filename%%.*}_sorted.txt
    sort $filename > $filename_sorted
    echo "Display using CAT or MORE?"
    read format
    if [ $format=="CAT" ];
```

```

then
cat $filename_sorted
else
more $filename_sorted
fi

fi;;
"5") exit ;;
esac

```

Script started on Tue 06 Oct 2015 21:17:00 BST

[4mstulinux[24m:[1m~[0m> ./scr[K[K[Kscript5.ksh

Write a Poem menu

- 1: Write a Poem
- 2: Word count the Poem
- 3: Display the Poem
- 4: Sort the output
- 5: Exit

What would you like to do?

1

[H[2J[?1049h[1;24r(B[m[4l[?7h[?12l[?25h[?1h=[?1h=[?1h=[39;49m[39;49m(B[m[H[2J(B[0;7m GNU
nano 2.2.6 New Buffer [23;1H^G(B[m Get Help (B[0;7m^O(B[m
WriteOut (B[0;7m^R(B[m Read File (B[0;7m^Y(B[m Prev Page (B[0;7m^K(B[m Cut Text
(B[0;7m^C(B[m Cur Pos

[24d(B[0;7m^X(B[m Exit[14G(B[0;7m^J(B[m Justify (B[0;7m^W(B[m Where Is (B[0;7m^V(B[m Next
Page (B[0;7m^U(B[m UnCut Text(B[0;7m^T(B[m To Spell

[3d[1;71H(B[0;7mModified

[3d(B[mThis is a new poem I would like to add.

[22d(B[0;7mSave modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? [23;1H
Y(B[m Yes[K

[24d(B[0;7m N(B[m No [14G (B[0;7m^C(B[m Cancel[K[22;62H

(B[0;7mFile Name to Write:

[23d^G(B[m Get Help[23;21H(B[0;7mM-D(B[m DOS Format[41G(B[0;7mM-A(B[m
Append[23;61H(B[0;7mM-B(B[m Backup File

[24d(B[0;7m^C(B[m Cancel[17G (B[0;7mM-M(B[m Mac Format[41G(B[0;7mM-P(B[m
Prefix[22;21H(B[0;7mt(B[m(B[0;7mh(B[m(B[0;7mi(B[m(B[0;7ms(B[m(B[0;7mi(B[m(B[0;7ms(B[m(B[0;
7ma(B[m(B[0;7mn(B[m(B[0;7me(B[m(B[0;7mw(B[m(B[0;7mp(B[m(B[0;7mo(B[m(B[0;7me(B[m(B[0;7
mm(B[m(B[0;7m.(B[m(B[0;7mt(B[m(B[0;7mx(B[m(B[0;7mt(B[m

[23d[39;49m(B[m[J[1;28H(B[0;7mFile: thisisanewpoem.txt[1;71H [22;32H(B[m[1K (B[0;7m[
Wrote 1 line](B[m[K[24;80H[24;1H[?1049I

[?1l>[4mstulinux[24m:[1m~[0m> ./script5.ksh

Write a Poem menu

- 1: Write a Poem
- 2: Word count the Poem
- 3: Display the Poem
- 4: Sort the output
- 5: Exit

Menu, so if 1 is typed in, the
poem shows up.

What would you like to do?

2

[H[2JWhat is the file name?

thisisanewpoem.txt

10 thisisanewpoem.txt

[4mstulinux[24m:[1m~[0m> ./script5.ksh

Write a Poem menu

- 1: Write a Poem
- 2: Word count the Poem
- 3: Display the Poem
- 4: Sort the output
- 5: Exit

What would you like to do?

3

[H[2JWhat is the file name?

^[A thjis isisanewpoem.txt

[H[2JWhich format would you like to use LESS or MORE

LESS

This is a new poem I would like to add.

[4mstulinux[24m:[1m~[0m> ./script5.ksh

Write a Poem menu

1: Write a Poem

2: Word count the Poem

3: Display the Poem

4: Sort the output

5: Exit

What would you like to do?

4

[H[2JWhat is the file name?

thisisanewpoem.txt

Display using CAT or MORE?

CAT

This is a new poem I would like to add.

[4mstulinux[24m:[1m~[0m> e[Kexit

Script done on Tue 06 Oct 2015 21:18:32 BST

We create a simple menu with echo statements then we *read menuChoice* to get the number corresponding to the menu action. After that we use *case \$menuChoice in "1") nano ;;* to do the action corresponding to the variable in *menuChoice*. First one is writing a poem so we just open for the user txt editor using the function *nano*. The second one is word count for the poem so first we use our code from previous script to check if the file exists and then we use *wc* command which stands for word count *wc filename*. The third option from menu is displaying the poem so yet

again we check if the file exists after that we ask the user in what format would he like to see the file either *Less* or *More* and we use a simple if statement to check what the user chose and depending on that we either use command *more filename* or *less filename*. Our fourth menu action is to sort the file save it and then display it so again we check if the file exists then we create file *filename_sorted=\$(filename%.*)_sorted.txt* which creates file with the same name and at the end adds *_sorted* after that we sort the file using the command *sort filename > \$filename_sorted* and last but not least we display the new sorted file but first we ask if he wants to see it via *Cat* or *More* the same we did previously with *Less* and *more*.

Discussion

Task 6 – Reflection

By undergoing a laboratory session on UNIX, we are able to understand the basic elements that make up the process of generating Korn Shell scripts and many other commands learnt during the process. This has really helped us as a group in our degrees as most of us had little programming language especially when it came to UNIX as no one has heard of UNIX before so this laboratory helped us gain knowledge that we can apply in future programmes. As a group, we have had experience with programming in java during this academic year on the Computer Programming course we did a bit of objective java. It took us a while to translate functions from java to shell so we can achieve results expected from us in the exercise. We have learnt that syntax is quite different from other programming languages for example we do not use { } brackets to indicate the beginning and the end of functions or if statements, switches and loops but instead we use words like then to start declaring our methods and fi after we are done. We have also learnt that Shell is sensitive.

For Task 1, we had to enter a command prompt-type system called UNIX, which when we entered it prompted that we should enter a username and password, from there we made a script which could be made by entering “script task 1”, then pressing enter, this made UNIX record everything that happens within the window. From here, we examined the effect of “cal 3000”; this made a script of the calendar in the year 3000. We then tried out “cal 6 3000”; this was the month June in the year 3000, we also tried out “cal 9 1752”, which was the month of September in the year 1752. The remaining set of commands were also simple to write and gain results from in one of the remaining exercises. We had to add and remove a file, we changed the directories file and had to name a directory and change its name. All of which was a success.

What I would improve of this task is by writing in the guide what is to be expected by entering all of these codes, this would help because the it shows that what we are doing is correct or not.

For Task 2, we had to access UNIX by using Xming. One of the first things we did is to launch Xming by typing in the CMS Student Solaris Server. Once logged in, we had to change the words from a poem which was a part of the task. We used the replace tool to find and change the word Shallot to Onion; this was saved as a new poem file. Then, to complete the task, we had to access files on Xming by typing in /home/userid (in this case, my user id is ub2232e, so I would put that in instead). This shows all of my files on the U drive. Once we completed this, we went onto opening up Firefox by using the Terminal in the Solaris server. We typed in the application’s name and Firefox appeared. I felt that this was very quick and easy to do.

One thing that I would change about Task 2 is to access one of the files, change it and see if the change was complete over UNIX and Windows. This would prove over two operating systems that it has accessed some files, edited it and saved it.

For Task 3, it was quite a small quick task to help us understand how to compile C programs in UNIX. This is the command we used as shown in the guide:

gcc filename.c -o filename.out

Where filename is the name of your chosen file.

By using this command, we are able to run C programs. To test this, we ran the hello file which was located in the ccs directory that was created in the previous task. By using a text editor, we managed to change "Hello world" to "Hello YourName" (in this case, YourName will be replaced as Jack Young, as it is my name). Saving the file and re-running the gcc command to run the hello file will appear as the new text "Hello Jack Young".

This was quite an easy task to do which got us to understand the fundamentals of C programs. I would not change anything for next year, as it was very insightful and easy to understand, not too much information at once.

In Task 4, we gained knowledge such as permissions of files, which you can specify how the owner, group or everyone can access the particular file. If I wanted only the owner of the file to access, edit and execute, and no else has access, then I will need to add the numbers together:

Read = 4
Write = 2
Execute = 1

 $4+2+1=7$

Therefore, seven will be my first number and as I do not want a group or everyone to access the file, I would need to put zero. The result would be:

chmod 700 filename
First number represents the owner.
Second number represents group.
Third number represents everyone.

Another example would be that if I wanted the owner to have all access but the group and everyone else has access to only read the file, then the number would be **744**.

The first two scripts and the fourth script for Task 4 were quite straight forward and everything we have had to do was already explained in the UNIX laboratory document. We did this

by typing up the codes for each of them onto text editor to create it and we ran each of them and got the results for it. The third and fifth script required a bit of know-how and we had to check syntax in Shell for loops. We have had some troubles with code everything seemed fine with syntax but it turned out that Shell is really whitespace and case-sensitive so *if[operation]* would be wrong and it has to be like *if [operation]*.

Overall, I think the group worked well as a team and we kept a positive attitude all the way through. Everyone contributed to the work and were able to overcome difficult tasks. I think that Task 4 really helped us to grasp the basics of Shell programming and understand how UNIX at its core works and it also helped us to understand concepts like loops, variables & user input output. The first two parts and the fourth part of Task 4 were reasonable but the other parts of Task 4 took a bit longer to understand. To help us understand Task 4, it was the communication and motivation with other group members to help solve the problem. Fortunately, we were able to complete Task 4 in an unsupervised lab session, the time we had helped up understand more of task 4 which helped us greatly; as if we do make certain mistakes again, we can easily recall back to the files that we created in this laboratory session and revise from them.

To improve the laboratory session for next year, I would make Task 4 a little easier to understand, as it was a big leap from Task 1-3 to Task 4.

Conclusion

In conclusion, we felt that we have learnt the way UNIX works. Each task shows us a different way of how it works. For example, Task 2 shows us how two operating systems could access each other's file etc.

References

B

Beal, V. (2015) What is UNIX Operating System? A Webopedia Definition, *Webopedia.com*, [online] Available at: <http://www.webopedia.com/TERM/U/UNIX.html>

(Accessed 9 October 2015).

G

Greenwich, University of. (2015) *Introduction To UNIX*, 1st ed, Greenwich, University of Greenwich, [online] Available at: <http://staffweb.cms.gre.ac.uk/~lg47/lectures/COMP1587/COMP1587Lab2.pdf>

(Accessed 6 October 2015).

T

Toronto, University of. (2015) The Lab Report, *Writing at the University of Toronto*, [online] Available at: <http://www.writing.utoronto.ca/advice/specific-types-of-writing/lab-report>

(Accessed 8 October 2015).