Lab One Report

JAHJA Darwin, 16094501d

1. Unix commands

- ls -- List the files in the current directory
- cd -- Change directories
- cat -- Concatenate and print (or display) the content of files
- rm -- Remove files
- cp -- Copy one or more files to another location
- mv -- Move a file or directory to a different location or Rename a file or directory
- mkdir -- Create new folder(s)
- cc -- The "C compiler" which can compile the C code.
- who -- Print all usernames currently logged in
- dir -- Briefly list directory contents

2. *cat* and *wc*

The subdirectory "q2" consists of 2 text file:

- *origin.txt* -- the created small text file.
- *repeated.txt* -- Another file consisting of five repetitions of *origin.txt*.

To create "*repeated.txt*", I have used a for loop to repeat `cat origin.txt` and appended the printing result using `>>` into "*repeated.txt*" for 5 times. Heres the command to run in bash:

```
~/lab_one/q2
$ for i in {1..5};do cat origin.txt >> repeated.txt; done
```

Then, I use `wc` count the number of characters and words in *origin.txt* and *repeated.txt*:

```
~/lab_one/q2
$ wc origin.txt repeated.txt
  1   9  46 origin.txt
  5  45 230 repeated.txt
  6  54 276 total
```

For the result, the meaning of each column are *Line numbers*, *Word counts*, *Character counts*, and *file name* respectively.

3. Execute commands and explain purposes of >, >>, echo

`echo` -- Display a line of text/string on standard output or a file. For example,

```
~/lab_one/q3
$ echo "I'm happy"
I'm happy
```

`>` -- Redirect standard output of a file into a new file else existing file. For example,

```
~/lab_one/q3
$ echo "I'm happy" > f1.txt

$ cat f1.txt
I'm happy
```

Beware that existing contents of '*f1.txt*' will be overwritten by contents of the standard output.

`>>` -- Appends standard output in existing file. For example,

```
~/lab_one/q3
$ echo "You are also happy" > f2.txt

$ cat f2.txt >> f1.txt

$ cat f1.txt
I'm happy
You are also happy
```

In this case, contents of '*f2.txt*' will be appended at the end of '*f1.txt*'.

4. C hello program: hi

Source code: `/q4/hi.c`

Compile and execute,

```
~/lab_one/q4
$ gcc hi.c -o hi

$ ./hi
Hi there, have a nice day~
```

5. C program: my_f

Source code: `/q5/my_f.c`

There are two errors in here:

1. A `void` keyword needs to be added in front of the main function.

```
void main(int argc, char *argv[]) {...}
```

2. The execution command of `my_f` should not include parentheses between the arguments. It should be:

```
./my_f "who am i"
# Or
./my_f who am i
```

Compile and execute. Here are the results:

```
~/lab_one/q5
$ gcc my_f.c -o my_f

$ ./my_f "who am i"
argv[0]: ./my_f
argv[1]: who am I

$ ./my_f who am i
argv[0]: ./my_f
argv[1]: who
argv[2]: am
argv[3]: I
```

6. C program: hw

Source code:

1. `/q6/hello_f.c` -- Contains the function to print 'hello, world'.
2. `/q6/hello_m.c` -- Contains the main function to call the function.

To compile and execute,

```
~/lab_one/q6
$ gcc hello_f.c hello_m.c -o hw

$ ./hw
hello, world
```

7. C program: count_text

Source code: `/q7/count_text.c`

To compile and execute,

```
~/lab_one/q7
$ gcc count_text.c -o count_text

$ ./count_text
it is not my work to count the number of words
it should be the business of the program
should it? *


The number of Char is: 99
The number of Word is: 21
The number of Line is: 3

$ ./count_text
Testing testing
The quick brown fox jumps over the lazy dog
Very quick...
The end *


The number of Char is: 82
The number of Word is: 15
The number of Line is: 4
```