CL205 - Operating Systems Lab

Lab#04

1. NANO Editor

There are many text editors available for Linux. At the moment you will have access to the nano editor. Nano is an advanced text editor provided by GNU. Simply typing *nano* on the shell will give you the editor. You may also start your nano by explicitly mentioning the file you want to work on. This file may be already existing or you may be creating a new one.

```
nano <myFile>
```

Near the end of your screen you will see a list of shortcuts. The ones which you should get yourself familiar with are as such:

CTRL+X for exit

CTRL+O for saving

CTRL+W for searching

CTRL+K for cutting

CTRL+U for pasting

CTRL+C for displaying cursor position

Other commands are listed at the bottom of the text-editor window.

2. Links

Links are the equivalent of Shortcuts in Windows. The syntax of creating a link to a file or directory is as such:

```
ln existingfile linkname
```

To practice, try out the following commands:

```
# ls
# touch blankfile
# mkdir blankdir
# ls
# ln blankdir dirpointer
# ln -s blankdir dirpointer
# ln blankfile filepointer
# cat < blankfile
# cat < filepointer
# echo "Hello dear" > filepointer
# cat < blankfile
# cat < filepointer
# ls
# ls –lh
```

Look carefully at the contents of both filepointer and dirpointer. Changing one will automatically change the other. Also look at how the directory pointers appear using ls -lh.

3. More | Less

Sometimes a user may give a command which generates so much output that it scrolls very fast. As a result, the top information simply flows out of the screen whereas only the low end information is visible. For example, use the following command

```
ls /bin -lh
```

We can view the lost information using the more or less commands. Try both of them first.

```
ls /bin -lh | more
```

To quit, press q.

```
ls /bin -lh | less
```

The syntax of both above commands are such that we are specifying two commands in one go. The first command starts with ls, the second command starts with more or less. Both these commands are joined by the pipe symbol **|**.

With more, you are able to browse down the display 1-page at a time using spacebar. With less, you are able to browse up and down the display 1-line at a time using up and down arrow keys. An alternative is using output redirection that you have covered in section-2.4.4. Usage would be as such:

```
ls /bin -lh > temp.txt
nano temp.txt
```

4. Searching

By default, searching for files or directories is performed using the find command.

The syntax of find command is as such:

```
find <where> -name <what>
```

So, if I want to find all pdf files in / directory, I will give:

```
find / -name *.pdf
```

5. File Permissions

All files and directories in Linux have an associated set of owner permissions that are used by the operating system to determine access. These permissions are grouped into three sets of three bits. The sets represent {owner, group, everyone else} whereas the bits represent {read, write, execute}. So overall, we have 9 permissions, as shown in the following table:

	Read	**Write**	**Execute**
Owner	1	1	1
Group	1	1	0
Others	0	0	1
	2^2	2^1	2^0

Representation of file permissions

If we look at the first row for Owner, we see three 1's. Which specify that the Owner has read, write, and execute permissions on a file or directory. Since all of the bits are in allow mode, we can add them up together to get $2^2 + 2^1 + 2^0 = 4 + 2 + 1 = 7$. Similarly, the second row for Group, i.e., users within the same group as that of the file owner, have read and write, but no execute permissions. This will be translated only as $2^2 + 2^1 = 4 + 2 = 6$. And lastly, every other user can only execute files and have no permission to read or write to them. This will be translated as 20 = 1. So the file permissions would be 761. To give the permission of 761 to a file, we would use the command:

```
chmod 761 <filename>
```

You can view the file permissions using the command:

```
ls –lh
```

6. File Ownership

A file's owner can be changed using the command:

```
chown <username> <filename>
```

Where, <username> would be the username of any user on your system, and <filename> is the target to which this setting is going to apply to.

7. GNU Compiler Collection

Programs written in C on linux are compiled using the gcc compiler. Programs written in C++ are compiled using the g++ compiler. Both these compilers are provided by GNU under the label GCC.

Any C program written for linux should have the extension of .c (e.g., hello.c), whereas a C++ program should have an extension of .cpp (e.g., hello.cpp). When compiling, the general syntax of command is as follows:

```
gcc first.c [-o first]
g++ first.cpp [-o first]
```

A breakdown is as such:

- gcc|g++ is the compiler itself
- first.c|first.cpp will be the filename with extension of your source code.
- [ ] Anything inside this square brackets is an optional arguments. Do not write the square brackets themselves in your command.
- -o is the output filename. It is the argument for making an executable file with the name you specify. Without this, the source code will be compiled into an executable file named a.out first is the name of executable file which you pass to -o argument.

So considering that we have a source code with the name first.c, and compile it with the above command, we can execute it using:

```
./first
```

Where ./ specifies the current directory, and first is the name of executable file which you passed as an argument to the compiler. Try it out using the following code:

```
#include <stdio.h>
main() {
        printf("hello, world\n");
}
```

Exercise

- Write your very firrst C program for linux. Name it as myFirst.c. Your program should prompt the user for his name, and then should display the name on the shell. **Tip:** Use the printf() and the scanf() calls for this purpose & remember to check their respective man pages as well (man printf & man scanf)
- To run a linux command from your program, we can use the system() function. Use the following inside your myFirst.c program and see it's output.

  system("ls");
- Using the system() function, do the following:
  - ➢ Create a directory with the name test in your current directory.
  - ➢ Then display the contents of that directory
  - ➢ Finally erase the test directory