CS202 : Lab Assignment 1
(Groups: G21 and G23)
Getting Familiar with Linux and Shell Script

1. Setting up the Linux environment

We have listed a few ways to setup Linux environments on different operating systems.

1.1. Installing linux

1.1.1. Setting up a virtual machine (Virtual Box/VM Ware)

You can install VMware/VirtualBox and then install a Linux-based OS (e.g. Ubuntu) in it. You can follow the tutorial here.

1.1.2. Dual Booting Linux with Windows

If your device already has Windows installed, you can dual boot it with a different free and open-source operating system, such as the Ubuntu. Here is a tutorial link.

1.2. Windows: Cygwin

Cygwin provides a Linux-based terminal to run Linux commands on Windows. To install Cygwin, go to this website and follow each steps.

1.3. MAC

Since both Linux and Mac are UNIX-based operating systems, you should be able to run the majority of commands without making any modifications.

1.4. Online: JSLinux

It's possible to execute the majority of Linux commands using this excellent online Linux terminal emulator.

2. Linux Commands

2.1. Basic Commands and Uses

Command	Uses	Sample
mkdir	To make a new directory.	`mkdir cs202`
vi / nano	Create/open a file.	`nano test.txt`
cp	To copy the files.	`cp test.txt ./root/cs202`
mv	To move the files	`mv test.txt ./root/cs202`
cd	For changing directory.	`cd cs202`

ls	Show list of contents in a directory.	`ls <directory_name>`
cat	To view the content of a file.	`cat test.txt`
rm	To remove the files.	`rm test.txt`
pwd	To print the current directory.	`pwd`
man	Open manual for a command.	`man <command_name>`

2.2. A Use-Case of working using basic commands

First check the current directory using ***pwd*** command. In this example, we are in the root directory.

```
[root@localhost ~]# pwd
/root
```

Now we will make a directory as name of "iiitg".

```
[root@localhost ~]# mkdir iiitg
```

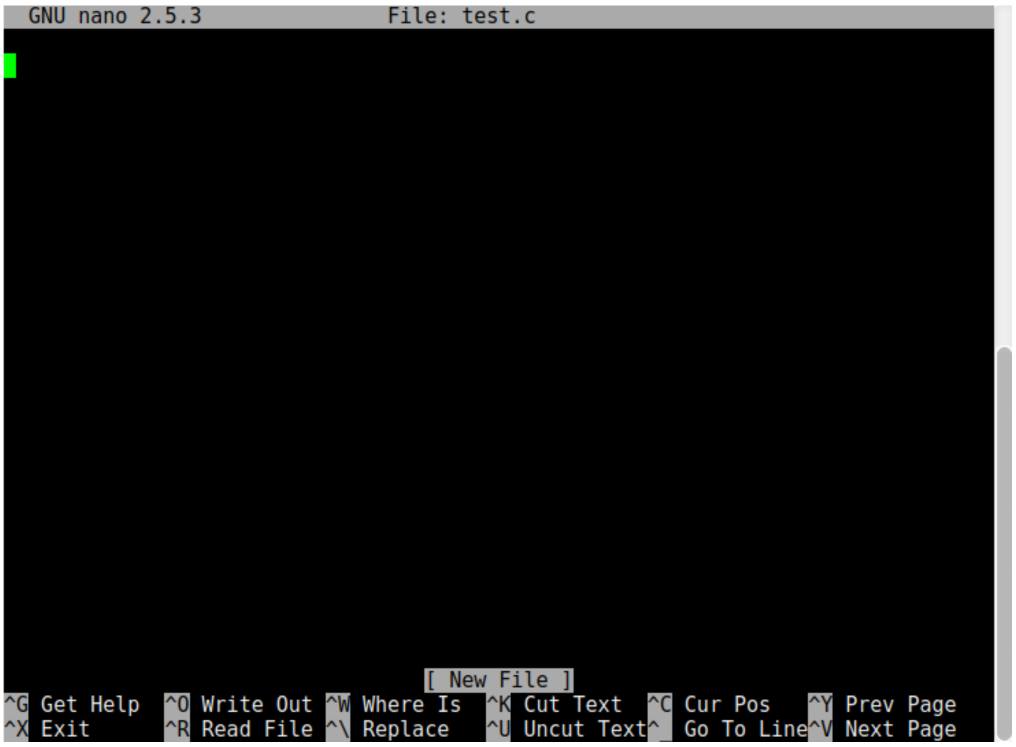
Visit the "iiitg" directory

```
[root@localhost ~]# cd iiitg
[root@localhost iiitg]#
```

Make a c program file using nano editor.

```
[root@localhost iiitg]# nano test.c
```

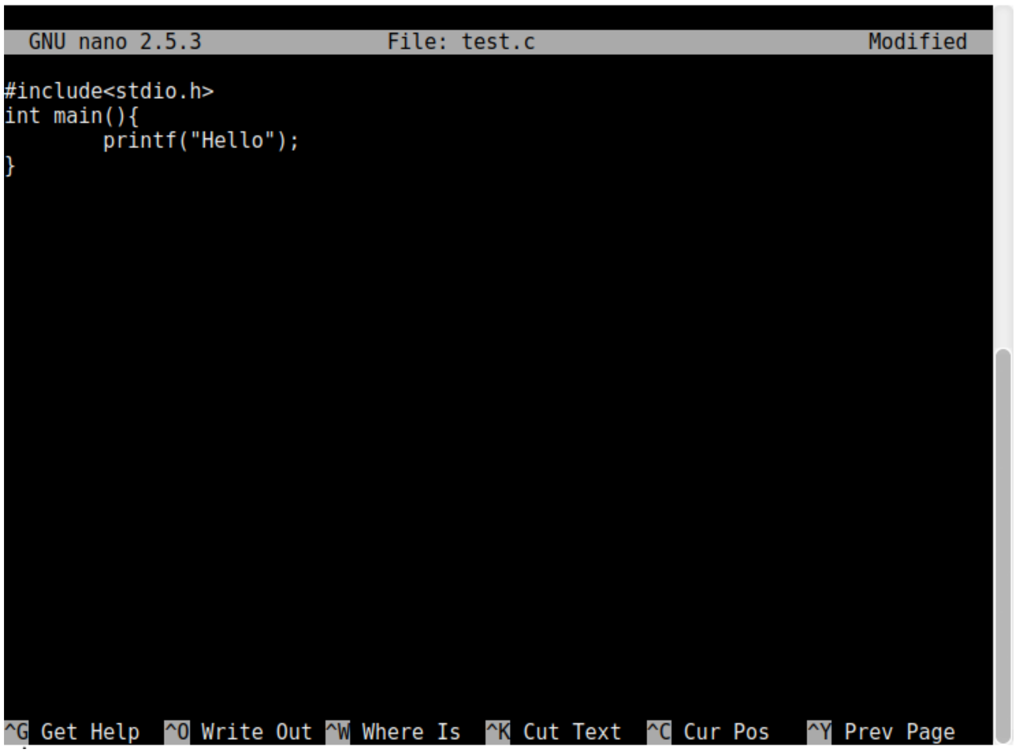
It will open like the following interface, here we write a *main()* which prints a message "Hello".



```
  GNU nano 2.5.3                File: test.c

                              [ New File ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^C Cur Pos    ^Y Prev Page
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text^  Go To Line^V Next Page
```

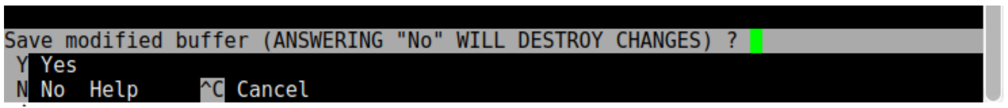
To save the file, press ***Ctrl+x*** from the keyboard and then press ***y***.



```
  GNU nano 2.5.3                File: test.c                              Modified

#include<stdio.h>
int main(){
        printf("Hello");
}

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^C Cur Pos    ^Y Prev Page
```



```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
 Y Yes
 N No  Help      ^C Cancel
```

To view the file what we have created now, use the ***ls*** command

```
[root@localhost iiitg]# ls
test.c
```

Compile the c file using ***gcc*** compiler.

```
[root@localhost iiitg]# gcc test.c -o test
```

Execute the executable file

```
[root@localhost iiitg]# ./test
Hello
```

Moving back to mother directory.

```
[root@localhost iiitg]# cd ..
```

Delete the file what we have created.

```
[root@localhost ~]# rm -r iiitg
```

3. Shell Script

3.1. Some sample shell script

Let's start by creating a simple shell script that prints "Hello, World!" to the console. We'll use Bash, one of the most popular Unix shells.

- Open a text editor and create a new file named *hello.sh*.
- Add the following code to the *hello.sh* file:

```
#!/bin/bash

# This is a comment
echo "Hello, World!"
```

- Save the file.
- Making the Script Executable by the command *chmod*.

```
[root@localhost ~]# chmod +x hello.sh
```

- Running the Shell Script

```
[root@localhost ~]# ./hello.sh

Hello, World!
```

- Shell script example for accepting user input

```
#!/bin/bash

# This is a comment
echo "What's your name?"
read name
echo "Hello, $name!"
```

- Shell script using conditional statements

```
#!/bin/bash

# This is a comment
echo "Enter a number:"
read num

if [ $num -gt 0 ]; then
echo "The number is positive."
elif [ $num -lt 0 ]; then
echo "The number is negative."
else
echo "The number is zero."
fi
```

- Shell script using loops

```
#!/bin/bash

# This is a comment
count=5

while [ $count -gt 0 ]; do
echo $count
((count--))
done

echo "Blastoff!"
```

3.2. To do

1. Write a shell script that takes a filename as input and checks if the file exists or not. If it exists, display its content; otherwise, create a new file with that name.
2. Create a shell script that takes a directory name as input and lists all the files and directories inside that directory.
3. Write a shell script that takes two filenames as input and concatenates the content of the first file to the second file.
4. Create a shell script that takes a filename as input and counts the number of lines, words, and characters in that file.
5. Write a shell script that reads a list of filenames from a file and copies all those files to a specified backup directory.
6. Create a shell script that takes a C source code file as input, compiles it, and generates an executable with the same name as the source file (without the extension).
7. Write a shell script that takes a directory name as input and compiles all the C files in that directory and its subdirectories. The resulting executables should be stored in a separate "bin" directory.
8. Write a shell script that takes a filename as input and checks if the file is executable. If it is executable, run the file; otherwise, display an error message.