



DEPARTMENT OF  
COMPUTER SCIENCE AND ENGINEERING

---

## Title: Linux/Unix Commands for Beginners

---

OPERATING SYSTEM LAB  
CSE 310



GREEN UNIVERSITY OF BANGLADESH

---

# 1 Objective(s)

- To gather knowledge of basic Linux/Unix commands for beginners.

## 2 Linux

Linux has a graphical user interface and it works pretty much like the GUI's on other systems that you are familiar with such as Windows and OSX. This tutorial won't focus on these as I reckon you can probably figure that part out by yourself. This tutorial will focus instead on the command line (also known as a terminal) running Bash. Some basic Linux tutorial for beginners:

- **The Command Line** - What is it, how does it work and how do I get to one.
- **File Manipulation** - How to make, remove, rename, copy and move files and directories.
- **Vi Text Editor** - Discover a powerful Linux based text editor.
- **Wildcards** - Also referred to as globbing, this is a means to refer to several files in one go.
- **Permissions** - Learn to identify and change the permissions of files and directories and what the consequences of these are.

### 2.1 The Command Line

A command line, or terminal, is a text based interface to the system. You are able to enter commands by typing them on the keyboard and feedback will be given to you similarly as text.

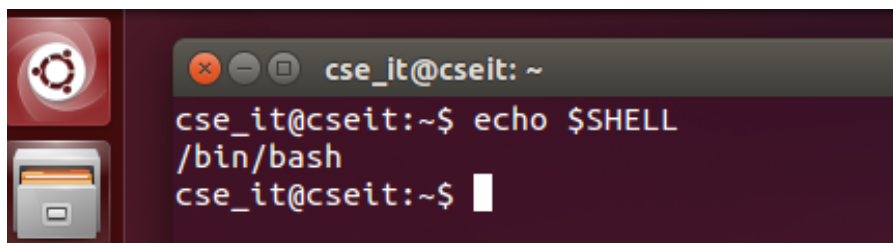
#### 2.1.1 Opening a Terminal

If on Linux then you will probably find it in **Applications -> System** or **Applications -> Utilities**. Alternatively, you may be able to **'right-click'** on the desktop and there may be an option **'Open in terminal'**.

#### 2.1.2 The Shell, Bash

Within a terminal you have what is known as a shell. This is a part of the operating system that defines how the terminal will behave and looks after running (or executing) commands for you. There are various shells available but the most common one is called **bash** which stands for **Bourne** again shell. This tutorial will assume you are using bash as your shell.

If you would like to know which shell you are using you may use a command called echo to display a system variable stating your current shell. **echo** is a command which is used to display messages.



```
1 echo $SHELL
2 /bin/bash
```

### 2.2 File Manipulation

This lesson will introduce the following commands:

---

### **2.2.1 pwd**

print work directory: e.g. if you are at your home directory then it will print something like /home/<username>

### **2.2.2 ls**

list directory: This command will list the items of a directory. If you don't specify a directory then it will list work directory, the place where you currently are.

### **2.2.3 cd**

change directory. It will change your work directly as you specify.

### **2.2.4 cd ..**

change directory one level up.

### **2.2.5 cd ~**

change to home directory.

### **2.2.6 cp**

Copy Command. It will copy a file or directory. It is similar to Copy-Paste in GUI.

### **2.2.7 mv**

Move or rename files. It is like renaming a file or cut-paste in GUI.

### **2.2.8 rm**

remove file or directory.

### **2.2.9 touch**

Create a Single Empty File

### **2.2.10 mkdir**

make directory. It will create a new directory. We will need to specify a name.

### **2.2.11 rmdir**

remove empty directory. This will not remove content of the directory but it will delete a directory if it is empty.

### **2.2.12 chmod**

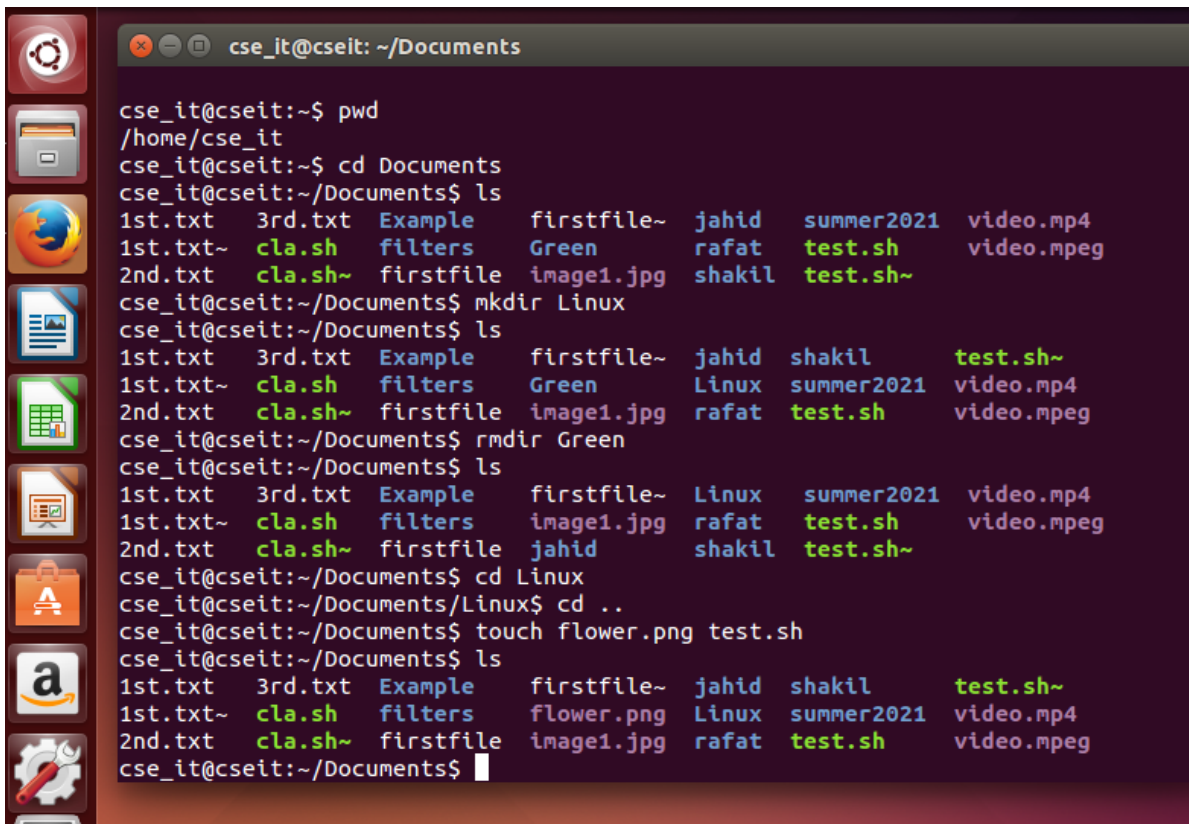
chmod is very useful command for beginners. It is used to change file permission. chmod stands for change mode in Linux.

### **2.2.13 clear**

Clear Screen: Use clear command to clear the terminal screen.

### 2.2.14 exit

exit from terminal.

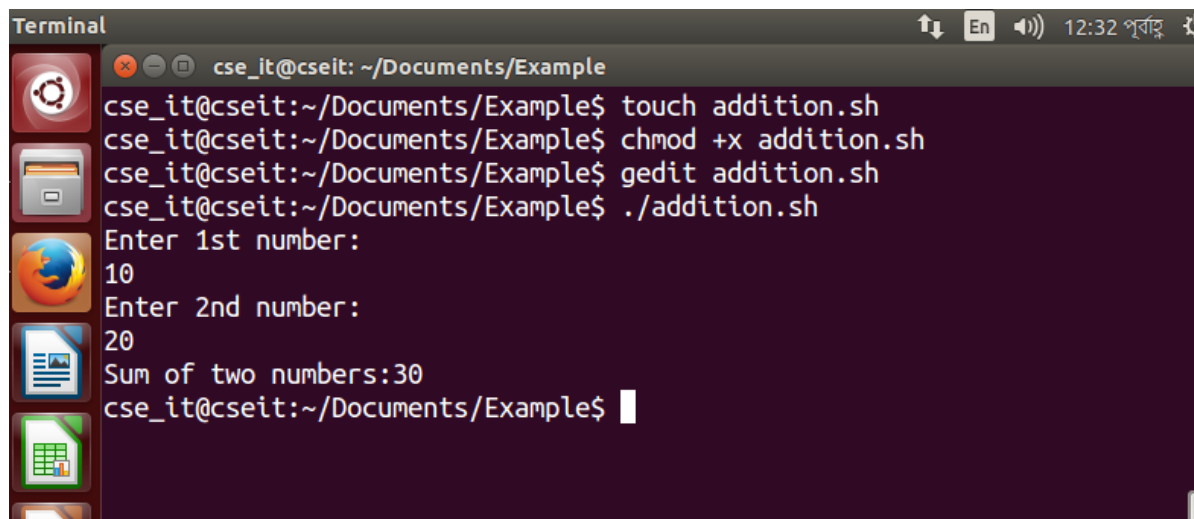


```
cse_it@cseit: ~/Documents
cse_it@cseit:~$ pwd
/home/cse_it
cse_it@cseit:~$ cd Documents
cse_it@cseit:~/Documents$ ls
1st.txt  3rd.txt  Example  firstfile~  jahid  summer2021  video.mp4
1st.txt~ cla.sh  filters  Green       rafat  test.sh     video.mpeg
2nd.txt  cla.sh~  firstfile image1.jpg  shakil test.sh~
cse_it@cseit:~/Documents$ mkdir Linux
cse_it@cseit:~/Documents$ ls
1st.txt  3rd.txt  Example  firstfile~  jahid  shakil  test.sh~
1st.txt~ cla.sh  filters  Green       Linux  summer2021  video.mp4
2nd.txt  cla.sh~  firstfile image1.jpg  rafat  test.sh  video.mpeg
cse_it@cseit:~/Documents$ rmdir Green
cse_it@cseit:~/Documents$ ls
1st.txt  3rd.txt  Example  firstfile~  Linux  summer2021  video.mp4
1st.txt~ cla.sh  filters  image1.jpg  rafat  test.sh     video.mpeg
2nd.txt  cla.sh~  firstfile jahid      shakil test.sh~
cse_it@cseit:~/Documents$ cd Linux
cse_it@cseit:~/Documents/Linux$ cd ..
cse_it@cseit:~/Documents$ touch flower.png test.sh
cse_it@cseit:~/Documents$ ls
1st.txt  3rd.txt  Example  firstfile~  jahid  shakil  test.sh~
1st.txt~ cla.sh  filters  flower.png  Linux  summer2021  video.mp4
2nd.txt  cla.sh~  firstfile image1.jpg  rafat  test.sh  video.mpeg
cse_it@cseit:~/Documents$
```

### 2.2.15 Run bash program in Linux

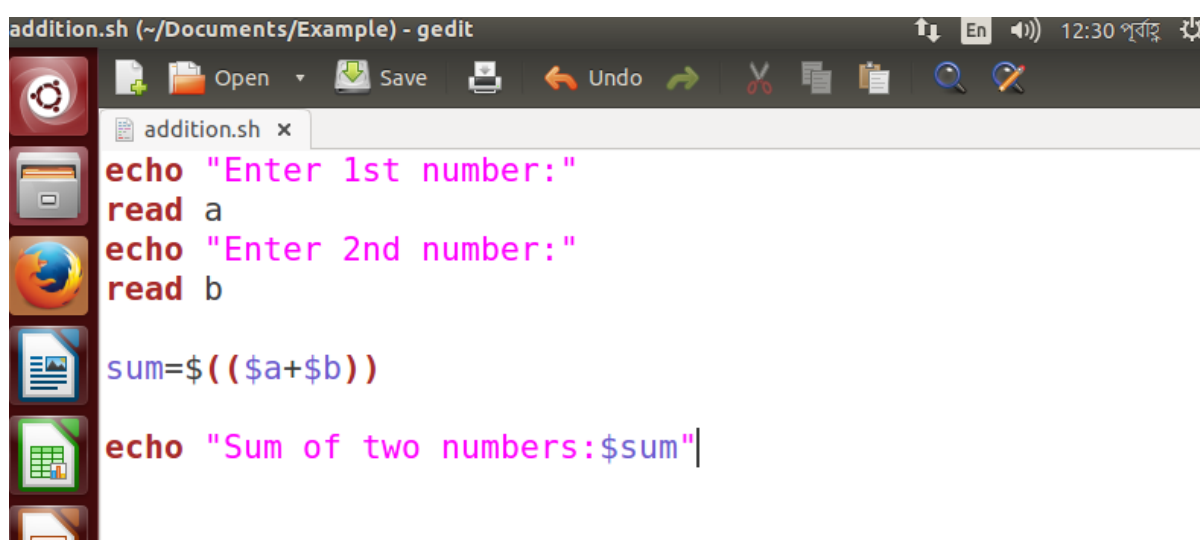
Making a bash script is a lot simpler than you might think.

- Create a file called addition.sh, using the touch command.
- Open the file using gedit command and edit the file with the program of your choice.
- In order to run a file directly, we'll need to change the permissions to allow the script to be executable for the user. chmod is a command that changes permissions on a file, and +x will add execute rights to the script.
- you can execute program simply using ./addition.sh [Note that: ./, which means a file in the current directory.]



A terminal window titled "Terminal" with the prompt "cse\_it@cseit: ~/Documents/Example". The user enters the following commands: `touch addition.sh`, `chmod +x addition.sh`, `gedit addition.sh`, and `./addition.sh`. The script prompts for "Enter 1st number:" (10) and "Enter 2nd number:" (20), then outputs "Sum of two numbers:30".

```
cse_it@cseit:~/Documents/Example$ touch addition.sh
cse_it@cseit:~/Documents/Example$ chmod +x addition.sh
cse_it@cseit:~/Documents/Example$ gedit addition.sh
cse_it@cseit:~/Documents/Example$ ./addition.sh
Enter 1st number:
10
Enter 2nd number:
20
Sum of two numbers:30
cse_it@cseit:~/Documents/Example$
```



A screenshot of the gedit text editor window titled "addition.sh (~/Documents/Example) - gedit". The window shows the following script content:

```
echo "Enter 1st number:"
read a
echo "Enter 2nd number:"
read b

sum=$((a+b))

echo "Sum of two numbers:$sum"
```

## 2.3 Vi Text Editor

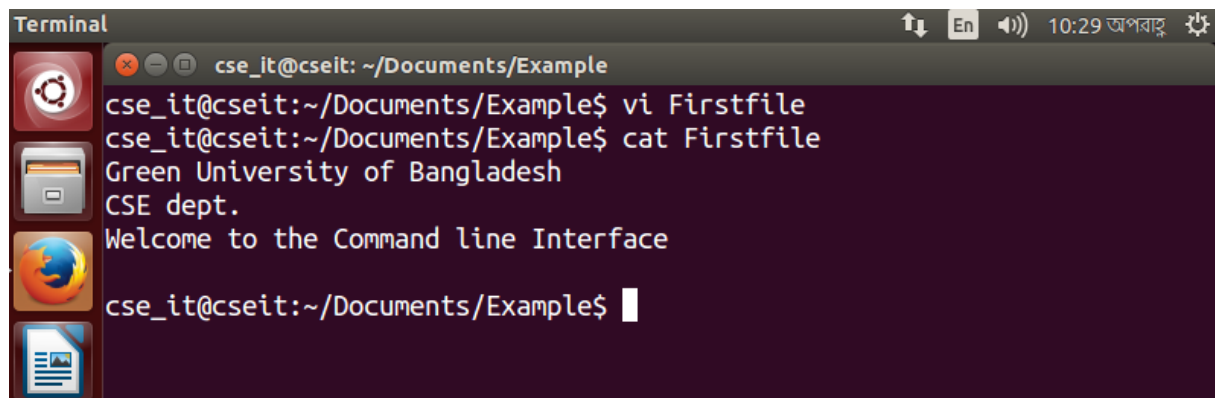
Vi is a command line text editor. As you would be quite aware now, the command line is quite a different environment to your GUI. It's a single window with text input and output only. Vi has been designed to work within these limitations and many would argue, is actually quite powerful as a result. Vi is intended as a plain text editor (similar to Notepad on Windows, or Textedit on Mac) as opposed to a word processing suite such as Word or Pages. It does, however have a lot more power compared to Notepad or Textedit.

As a result you have to ditch the mouse. Everything in Vi is done via the keyboard.

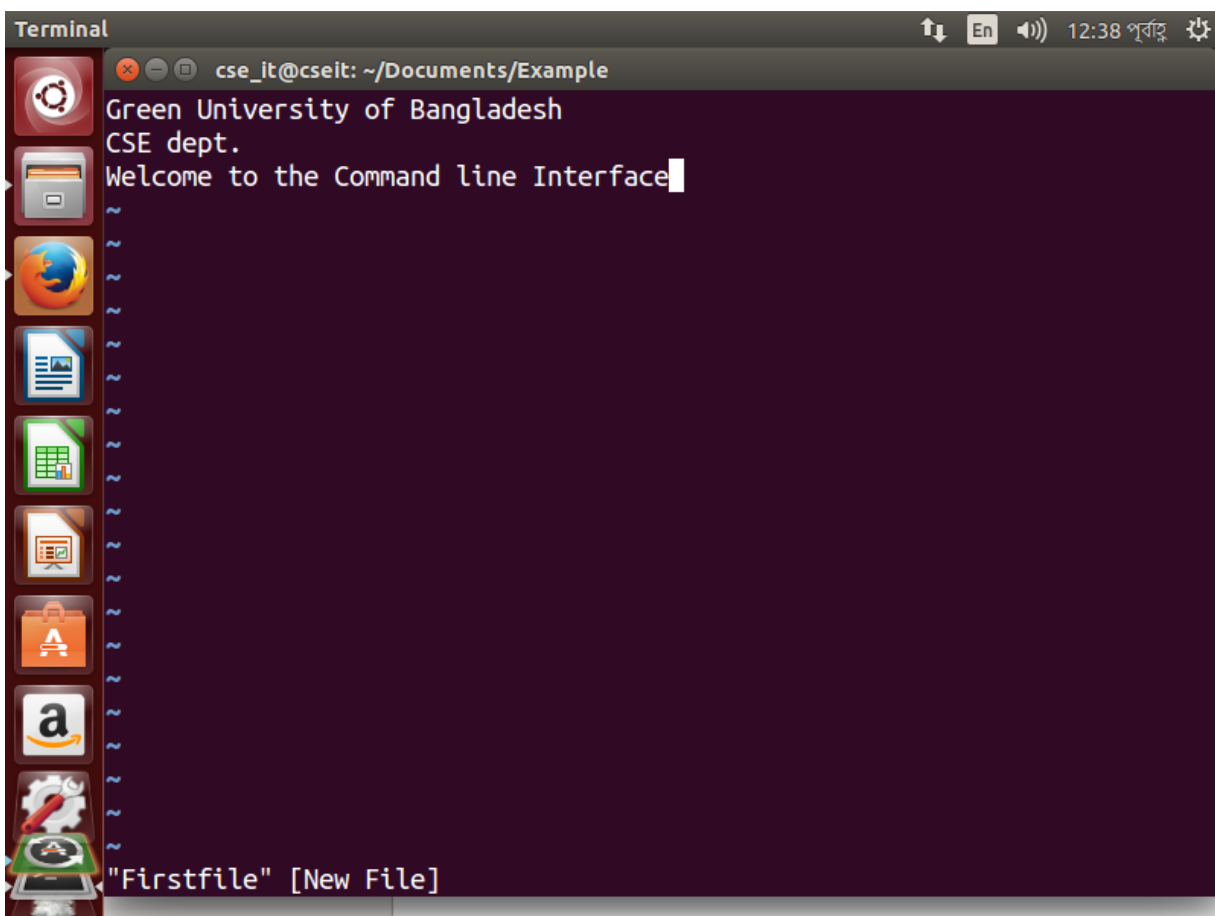
There are two modes in Vi. **Insert (or Input) mode** and **Edit mode**. In input mode you may input or enter content into the file. In edit mode you can move around the file, perform actions such as deleting, copying, search and replace, saving etc. A common mistake is to start entering commands without first going back into edit mode or to start typing input without first going into **insert mode**. If you do either of these it is generally easy to recover so don't worry too much.

You always start off in edit mode so the first thing we are going to do is switch to insert mode by pressing **i**.

Now type in a few lines of text and press **Esc** which will take you back to edit mode.



The terminal window shows a user named cse\_it@cseit in the directory ~/Documents/Example. The user runs the command `vi Firstfile` to create a new file. Then, they run `cat Firstfile` to display the contents of the file. The output shows the text: "Green University of Bangladesh", "CSE dept.", and "Welcome to the Command line Interface". The prompt is `cse_it@cseit:~/Documents/Example$`.



The terminal window shows the user in the vi editor editing the file Firstfile. The text "Green University of Bangladesh", "CSE dept.", and "Welcome to the Command line Interface" is visible. The bottom of the screen shows the status bar with the filename "Firstfile" and the mode "[New File]". The prompt is `cse_it@cseit:~/Documents/Example`.

### 2.3.1 Saving and Exiting

If you are unsure if you are in edit mode or not you can look at the bottom left corner. As long as it doesn't say **INSERT** you are fine. Alternatively you can just press **Esc** to be sure. If you are already in edit mode, pressing Esc does nothing so you won't do any harm.

- **ZZ (Note: capitals)** - Save and exit
- **:q!** - discard all changes, since the last save, and exit
- **:w** - save file but don't exit
- **:wq** - again, save and exit

## 2.4 Wildcards

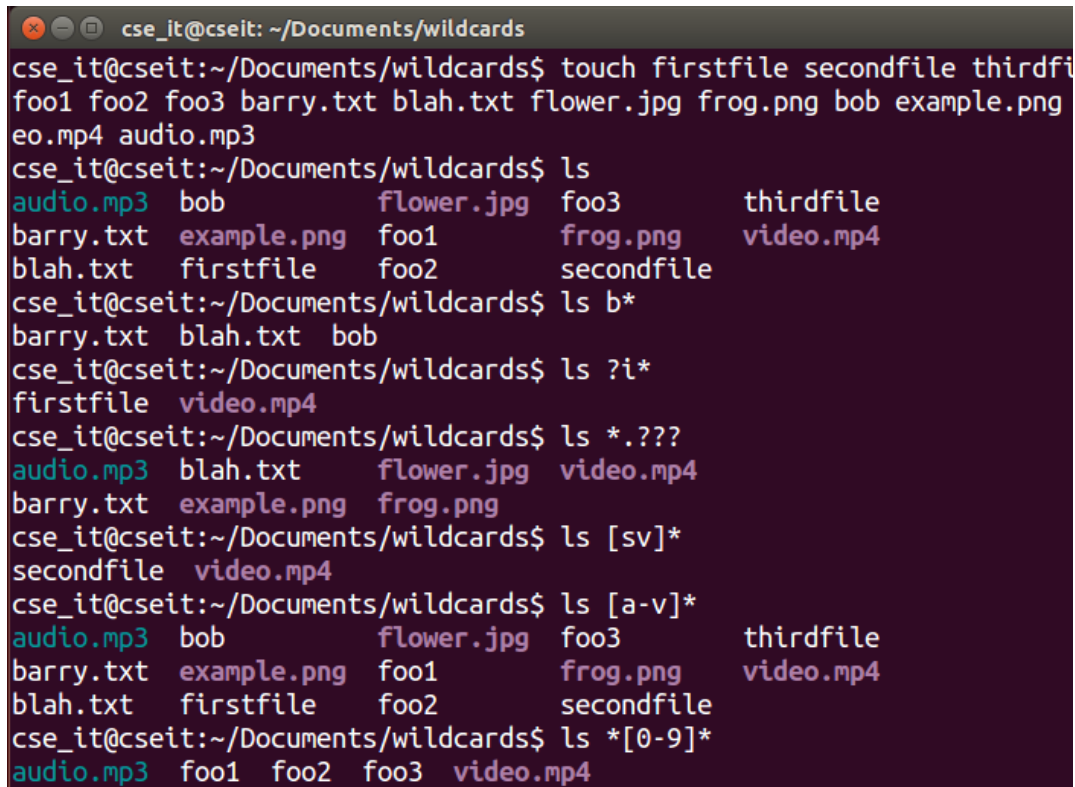
Wildcards are a set of building blocks that allow you to create a pattern defining a set of files or directories. As you would remember, whenever we refer to a file or directory on the command line we are actually referring

---

to a path. Whenever we refer to a path we may also use wildcards in that path to turn it into a set of files or directories.

Here is the basic set of wildcards:

- \* - represents zero or more characters
- ? - represents a single character
- [] - represents a range of characters



```
cse_it@cseit: ~/Documents/wildcards
cse_it@cseit:~/Documents/wildcards$ touch firstfile secondfile thirdfile
foo1 foo2 foo3 barry.txt blah.txt flower.jpg frog.png bob example.png
eo.mp4 audio.mp3
cse_it@cseit:~/Documents/wildcards$ ls
audio.mp3  bob          flower.jpg  foo3          thirdfile
barry.txt  example.png  foo1        frog.png      video.mp4
blah.txt   firstfile    foo2        secondfile
cse_it@cseit:~/Documents/wildcards$ ls b*
barry.txt  blah.txt  bob
cse_it@cseit:~/Documents/wildcards$ ls ?i*
firstfile  video.mp4
cse_it@cseit:~/Documents/wildcards$ ls *.???
audio.mp3  blah.txt  flower.jpg  video.mp4
barry.txt  example.png  frog.png
cse_it@cseit:~/Documents/wildcards$ ls [sv]*
secondfile  video.mp4
cse_it@cseit:~/Documents/wildcards$ ls [a-v]*
audio.mp3  bob          flower.jpg  foo3          thirdfile
barry.txt  example.png  foo1        frog.png      video.mp4
blah.txt   firstfile    foo2        secondfile
cse_it@cseit:~/Documents/wildcards$ ls *[0-9]*
audio.mp3  foo1  foo2  foo3  video.mp4
```

## 2.5 Permissions!

Linux permissions dictate 3 things you may do with a file, read, write and execute. They are referred to in Linux by a single letter each.

- **r read** - you may view the contents of the file.
- **w write** - you may change the contents of the file.
- **x execute** - you may execute or run the file if it is a program or script.

For every file we define 3 sets of people for whom we may specify permissions.

- **owner** - a single person who owns the file. (typically the person who created the file but ownership may be granted to some one else by certain users)
- **group** - every file belongs to a single group.
- **others** - everyone else who is not in the group or the owner.

Three permissions and three groups of people. That's about all there is to permissions really. Now let's see how we can view and change them.

### 2.5.1 View Permissions

To view permissions for a file we use the long listing option for the command ls.

```
1 ls -l [path]
```

---

```
1 ls -l /home/ryan/linuxtutorialwork/frog.png
2 -rwxr----x 1 harry users 2.7K Jan 4 07:32 /home/ryan/linuxtutorialwork/frog.png
```

### 2.5.2 Change Permissions

To change permissions on a file or directory we use a command called `chmod`. It stands for change file mode bits which is a bit of a mouthfull but think of the mode bits as the permission indicators.

```
1 chmod [permissions] [path]
```

`chmod` has permission arguments that are made up of 3 components

- Who are we changing the permission for? [u]go[a] - user (or owner), group, others, all
- Are we granting or revoking the permission - indicated with either a plus ( + ) or minus ( - )
- Which permission are we setting? - read ( r ), write ( w ) or execute ( x )

Grant the execute permission to the group. Then remove the write permission for the owner.

```
1 ls -l frog.png
2 -rwxr----x 1 harry users 2.7K Jan 4 07:32 frog.png
3
4 chmod g+x frog.png
5 ls -l frog.png
6 -rwxr-x--x 1 harry users 2.7K Jan 4 07:32 frog.png
7
8 chmod u-w frog.png
9 ls -l frog.png
10 -r-xr-x--x 1 harry users 2.7K Jan 4 07:32 frog.png
```

Don't want to assign permissions individually? We can assign multiple permissions at once.

```
1 ls -l frog.png
2 -rwxr----x 1 harry users 2.7K Jan 4 07:32 frog.png
3
4 chmod g+wx frog.png
5 ls -l frog.png
6 -rwxrwx--x 1 harry users 2.7K Jan 4 07:32 frog.png
7
8 chmod go-x frog.png
9 ls -l frog.png
10 -rwxrw---- 1 harry users 2.7K Jan 4 07:32 frog.png
```

### 2.5.3 Lab Task (Please implement yourself and show the output to the instructor)

1. Practice the Linux commands:

- File manipulation
- Wildcards
- Permissions

## 3 Discussion & Conclusion

Based on the focused objective(s) to understand about the basic linux commands, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).



---

## 4 Lab Exercise (Submit as a report)

- Implement the Linux commands:
  1. File manipulation
  2. Wildcards
  3. Permissions

## 5 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.