**CLASS I**

**LINUX BASICS**

**Multics** (Multiplexed Information and Computing Service) is a timesharing operating system begun in 1965 and used until 2000.

Multics included

* a supervisor program that managed all hardware resources, using symmetric multiprocessing, multiprogramming, and paging
* an innovative segmented memory addressing system supported by hardware
* a tree structured file system
* device support for peripherals and terminals
* hundreds of command programs, including language compilers and tools
* hundreds of user-callable library routines
* operational and support tools
* user and system documentation

**Unics** (Uniplexed Information and Computing Service)

After words Unics is been renamed to **Unix** and it is a Liceansed version in which each system costs around 25 Lakh so that then introduced Linux which is similar to Unix and it is a Open Source (means Installables and Source Code freely avaliabe).

**Linux** is a Unix-like computer operating system assembled under the model of free and open-source software development and distribution. The defining component of Linux is the Linux kernel,an operating system kernel first released on September 17, 1991 by Linus Torvalds(A student in Finland).

For More Info : [http://www.computerhope.com](http://www.computerhope.com/history/unix.html)

**GPL:** General Pubic Liceansing

1.Pay money and don’t release the source code to Public

2.Don’t pay money and release the source code to Public

**Linux has two Flavours :**

1.GUI

2.Non GUI, Our entire discussion will be on Non GUI

|

home

|

pradeep

**pwd** : **pwd** prints the full pathname of the current working directory.

pradeep@ubuntu:~$ pwd

/home/pradeep

**ls** : Lists the contents of a directory.

List information about the *FILE*s (the current directory by default). Sort entries alphabetically if none of **-cftuvSUX** nor –**sort** is specified.

pradeep@ubuntu:~$ ls

Desktop Downloads Music Public Videos

Documents examples.desktop Pictures Templates

**touch :** If **file1.txt** doesn't exist, it is created as a new, empty file.

pradeep@ubuntu:~$ touch file1.txt

pradeep@ubuntu:~$ ls

Desktop Downloads **file1.txt**  Pictures Templates

Documents examples.desktop Music Public Videos

**Note** : If **file1.txt** exists, **touch** updates its access and modification times to the current time.

pradeep@ubuntu:~$ ls -l file1.txt

-rw-rw-r-- 1 pradeep pradeep 0 Nov 27 12:25 file1.txt

**clear :** Clears the screen. This is equivalent to typing **Control-L** when using the bash shell.

pradeep@ubuntu:~$ clear

**mkdir :** If the specified *DIRECTORY* does not already exist, **mkdir** creates it.

pradeep@ubuntu:~$ mkdir LinuxSamples

pradeep@ubuntu:~$ ls

Desktop Downloads file1.txt Music Public Videos

Documents examples.desktop **LinuxSamples** Pictures Templates

Creates the directory **/home/pradeep/a/b/c**. If the parent directory **/home/pradeep/a/b** does not already exist, **mkdir** will create that directory first.

**Note :** More than one *DIRECTORY* may be specified when calling **mkdir**.

For multiple directories at a time mkdir dirname1 dirname2 ....

**cp:** The **cp** command is used to make copies of files and directories.

**cp Syntax: cp [option]...Source..Destination**

pradeep@ubuntu:~/LinuxSamples$ ls

file1.txt tmp1

pradeep@ubuntu:~/LinuxSamples$ cp file1.txt tmp1

pradeep@ubuntu:~/LinuxSamples$ ls

file1.txt tmp1

pradeep@ubuntu:~/LinuxSamples$ cd tmp1

pradeep@ubuntu:~/LinuxSamples/tmp1$ ls

file1.txt

pradeep@ubuntu:~/LinuxSamples/tmp1$

If the destination file **file1.txt** already exists, it will be overwritten without a confirmation prompt. This is the default behavior for all **cp** operations.

If you want to be prompted before overwriting a file, use the -i (interactive) option. For example:

cp –i oldfile new file

If newfile already exists, you will be prompted:

cp: overwrite ‘newfile’?

If you type y (or yes, Y, YES, or any other combination of upper and lowercase of these), then newfile will be overwritten with a copy of origfile. Typing anything else will abort the operation.

**Copy a file into another directory:**

**cp file /Samples/sub**

Creates a copy of the file in the working directory named **file**. The copy will be located in the directory **/Samples/sub**, and will be named **file**.

**cd:** The **cd** command, which stands for "change directory", changes the shell's current working directory.It allows you to change your working directory. You use it to move around within the hierarchy of your file system.

pradeep@ubuntu:~$ cd LinuxSamples

pradeep@ubuntu:~/LinuxSamples$ ls

file1.txt tmp1

pradeep@ubuntu:~/LinuxSamples$

**gedit**: **gedit** is a text editor designed with ease of use in mind. The gedit editor is often used for editing source code and markup languages (e.g. HTML and XML).

pradeep@ubuntu:~/LinuxSamples$ gedit file1.txt

it will open a new window to write, write the content, save and close it. After closing it will redirect you to the Terminal.

**mv:** The mv command is used to move or [rename](http://www.computerhope.com/jargon/r/rename.htm)files. mv renames file *SOURCE* to *DEST*, or moves the *SOURCE* file (or files) to *DIRECTORY*.

**mv Syntax : mv [Option]...Source...Destination**

pradeep@ubuntu:~/LinuxSamples$ ls

file1.txt tmp1 tmp2

pradeep@ubuntu:~/LinuxSamples$ mv file1.txt tmp2

pradeep@ubuntu:~/LinuxSamples$ ls

tmp1 tmp2

pradeep@ubuntu:~/LinuxSamples$

pradeep@ubuntu:~/LinuxSamples$ cd tmp2

pradeep@ubuntu:~/LinuxSamples/tmp2$ ls

file1.txt

**NOTE :**

**. –** Current or Present Directory

**.. –** Parent Directory

**/** **–** Absolute Path (Starts from root)

**. –** Relative Path(Starts from Current directory )

**cd .. (or) cd ~ (or) cd** :**–** Navigate back to Previous Directory

pradeep@ubuntu:~/LinuxSamples/tmp1$ pwd

/home/pradeep/LinuxSamples/tmp1

pradeep@ubuntu:~/LinuxSamples/tmp1$ cd ..

pradeep@ubuntu:~/LinuxSamples$

cd ../.. **–** Navigate back to 2 Directory

**rmdir:** Removes a [directory](http://www.computerhope.com/jargon/d/director.htm).

The rmdir utility removes the directory entry specified by each directory [argument](http://www.computerhope.com/jargon/a/argument.htm), provided the directory is empty.

Arguments are processed in the order given. To remove both a [parent](http://www.computerhope.com/jargon/p/parechil.htm) directory and a [subdirectory](http://www.computerhope.com/jargon/s/subdirec.htm) of that parent, the subdirectory must be specified first so the parent directory is empty when **rmdir** tries to remove it.

pradeep@ubuntu:~/LinuxSamples$ mkdir tmp3

pradeep@ubuntu:~/LinuxSamples$ ls

tmp1 tmp2 tmp3

pradeep@ubuntu:~/LinuxSamples$ rmdir tmp3

pradeep@ubuntu:~/LinuxSamples$ ls

tmp1 tmp2

**rm:** The **rm** command removes (deletes) files.

pradeep@ubuntu:~/LinuxSamples$ ls

file2.txt tmp1 tmp2

pradeep@ubuntu:~/LinuxSamples$ rm file2.txt

pradeep@ubuntu:~/LinuxSamples$ ls

tmp1 tmp2

Create a file with 30 lines (FileText.txt) using gedit ?

pradeep@ubuntu:~/LinuxSamples$ gedit FileText.txt

**ls :**

pradeep@ubuntu:~/LinuxSamples$ ls FileText.txt **(Command with Argument)**

FileText.txt

To check whether file exists or not if exists displays the File Name if not it will display

No such file or directory error.

**ls – l:** Long Listing

pradeep@ubuntu:~/LinuxSamples$ ls -l FileText.txt **(Command with Option)**

-rw-rw-r-- 1 pradeep pradeep 3432 Nov 27 13:41 FileText.txt

**cat:** cat stands for "catenate." It reads data from files, and outputs their contents. It is the simplest way to display the contents of a file at the command line.cat is one of the most commonly-used commands in Linux. It can be used to:

* Display text files
* Copy text files into a new document
* Append the contents of a text file to the end of another text file, combining them

**cat Syntax:** cat FileName

pradeep@ubuntu:~/LinuxSamples/tmp2$ cat file1.txt

Hello this is gedit Sample

pradeep@ubuntu:~/LinuxSamples/tmp2$ cat file2.txt

This is a Sample text file 2

pradeep@ubuntu:~/LinuxSamples/tmp2$ cat file1.txt file2.txt

Hello this is gedit Sample

This is a Sample text file 2

cat FileText.txt | more (Press space to view more lines)

cat FileText.txt | pg (Press enter to view more lines)

**head:**  To display first 10 lines in a file

head FileText.txt

head –n 3 FileText.txt (To display first 3 lines)(Command option with Value)

**tail:** To display last 10 lines in a file

tail FileText.txt

**File Size Calculation :** (Sample.txt)

This is file sample (19)

Second line(11)

last line(9)

In the above total size is 42 but we can see in the above the no. of characters were 39 and what about the 3 characters those are for newline(\n) and end of line (\0) will be added internally so the total size is 42.See in the below

This is file sample\n(20)

Second line\n(12)

last line\0(10)

pradeep@ubuntu:~/LinuxSamples$ ls -l Sample.txt

-rw-rw-r-- 1 pradeep pradeep **42** Nov 27 14:09 Sample.txt

**man:** On [Linux](http://www.computerhope.com/jargon/l/linux.htm) and other [Unix](http://www.computerhope.com/jargon/u/unix.htm)-like [operating systems](http://www.computerhope.com/os.htm), **man** is the interface used to view the system's reference manuals. **man** is the system's manual viewer; it can be used to display manual pages, scroll up and down, search for occurrences of specific text, and other useful functions.

Location of man file : /usr/share/man$

man1 - Commands

man2 - System Calls

man4 - API or Library Functions

man5 - Configuration Files

**CLASS - II**

**Command Types:**

Command - **ls**

Command with arguments – **ls file1.txt**

Command with options – **ls -l**

Command with option and values – **head –n 5 file1.txt**

**wc: wc**, or "word count," prints a count of [newlines](http://www.computerhope.com/jargon/n/newline.htm), words, and [bytes](http://www.computerhope.com/jargon/b/byte.htm) for each input [file](http://www.computerhope.com/jargon/f/file.htm) and a total if more than one FILE is specified. With no FILE, or when FILE is a dash ("**-**"), **wc** operates on [standard input](http://www.computerhope.com/jargon/s/stdin.htm). (A word is a non-zero-length sequence of characters delimited by [white space](http://www.computerhope.com/jargon/w/whitspac.htm).)

pradeep@ubuntu:~/LinuxSamples$ wc FileText.txt

42 564 3432 FileText.txt

pradeep@ubuntu:~/LinuxSamples$ wc -c FileText.txt (**-c for no.characters**)

3432 FileText.txt

pradeep@ubuntu:~/LinuxSamples$ wc -w FileText.txt (**-w for no. words**)

564 FileText.txt

pradeep@ubuntu:~/LinuxSamples$ wc -l FileText.txt(**-l for no. lines**)

42 FileText.txt

**Important Directories :**

pradeep@ubuntu:/$ ls

**bin** cdrom **etc** initrd.img **lib** lost+found mnt **proc** run snap **sys** **usr**  vmlinuz

boot **dev** home initrd.img.old lib64 **media** opt root sbin srv **tmp** var vmlinuz.old

**Binaries :**

pradeep@ubuntu:/$ which ls

/bin/ls

pradeep@ubuntu:/$ which python

/usr/bin/python

pradeep@ubuntu:/$ which perl

/usr/bin/perl

**Pipe:** (|) Pipe will take input from another commands output and generate its result

pradeep@ubuntu:/$ ls | wc -l

26

pradeep@ubuntu:/$ ls | wc -l | wc -l

1

**sort:** sort sorts the contents of a text file, line by line.sort is a simple and very useful command which will rearrange the lines in a text file so that they are sorted, numerically and alphabetically. By default, the rules for sorting are:

* lines starting with a number will appear before lines starting with a letter;
* lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet;
* lines starting with a lowercase letter will appear before lines starting with the same letter in uppercase.

pradeep@ubuntu:/$ sort (Press Enter)

v

o

t

a

r

y (ctrl +d for end of file)

O/P:

a

o

r

t

v

y

Create a file names.txt and sort then .

**i/o Redirection:** Taking input from file instead of keyboard and sending result to a file instead of displaying on the terminal.

Keyboard Terminal

Application Program

File File

**Standard Input : <**

Default

sort <names.txt (or) sort names1.txt (or)

**Standard Output : >**

sort names.txt > sorted\_names.txt (Will write the sorted names to the file sorted\_names)

1. If file not Exists
   1. Creates the file
   2. Writes the text
2. If file Exists
   1. Truncates the file
   2. Write the new text

**Append : >>** To append text to the end of the file

sort names.txt >> sorted\_names.txt

sort sorted\_names.txt >> finalsorted.txt

**Content form Terminal to a File :**

pradeep@ubuntu:~/LinuxSamples$ cat >file4.txt

This is my file

Named file4.txt

End of file file4.txt (ctrl +d for end of file)

pradeep@ubuntu:~/LinuxSamples$ cat file4.txt

This is my file

Named file4.txt

End of file file4.txt

pradeep@ubuntu:~/LinuxSamples$ cat >> file4.txt

EOF (ctrl +d for end of file)

pradeep@ubuntu:~/LinuxSamples$ cat file4.txt

This is my file

Named file4.txt

End of file file4.txt

EOF

**VI Editor:**

**Non GUI :** vi(Visual Editor),vim(Improved Visual Editior)

vim is useful while working on large file (>20 lines)to navigate faster

**GUI:** gedit,emax

**VI Modes:**

1. **Edit Mode (Text)**

Writing new text or modifying existing text

1. **Command Mode**

Process the text (Add line,delete line,delete character, delete word,copy lines,joining the lines,Navigation)

1. **Ex-Mode**

Find,replace,save,quit

Command Mode is default mode, to change it to Edit Mode we can use these characters (i,I,o,O,a,A)

**ESC** to change from Edit Mode to Command Mode

pradeep@ubuntu:~/LinuxSamples$ cat file4.txt

This is my file

Named file4.txt

End of file file4.txt

EOF

**Copy and Paste Lines:**

pradeep@ubuntu:~/LinuxSamples$ vi file4.txt

vi editor will open the file in Command Mode,navigate to the line which you want to copy and press “yy” and press “p” to paste it where ever you want.

Press ESC then :wq to save and quit to terminal

**NOTE:** **2yy** will copy two lines where as the **3p** will paste the copied lines three times

pradeep@ubuntu:~/LinuxSamples$ cat file4.txt

This is my file

Named file4.txt

End of file file4.txt

EOF

This is my file

This is my file

**Replace a Character:**

pradeep@ubuntu:~/LinuxSamples$ vi file4.txt

vi editor will open the file in Command Mode,navigate to the character which you want to replace and press “r” and enter your character it will replace

Press ESC then :wq to save and quit to terminal

**Change a Word :**

pradeep@ubuntu:~/LinuxSamples$ vi file4.txt

vi editor will open the file in Command Mode,navigate to the word which you want to change and press “wc” and enter your word it will change

Press ESC then :wq to save and quit to terminal

**Replace Characters :**

pradeep@ubuntu:~/LinuxSamples$ vi file4.txt

vi editor will open the file in Command Mode,navigate to the characters which you want to replace and press “R” and enter your characters it will replace

Press ESC then :wq to save and quit to terminal

**Delete Line :**

pradeep@ubuntu:~/LinuxSamples$ vi file4.txt

vi editor will open the file in Command Mode,navigate to the line which you want to delete and press “dd” and it will delete the line

Press ESC then :wq to save and quit to terminal

**Delete Word :**

pradeep@ubuntu:~/LinuxSamples$ vi file4.txt

vi editor will open the file in Command Mode,navigate to the word which you want to delete and press “dw” and it will delete the word

Press ESC then :wq to save and quit to terminal

**Delete Character :**

pradeep@ubuntu:~/LinuxSamples$ vi file4.txt

vi editor will open the file in Command Mode,navigate to the character which you want to delete and press “x” and it will delete

**Note :** x – will delete only 1 character

10 x will delete 10 characters

Press ESC then :wq to save and quit to terminal

**Navigation :**

L H K J

**EX –Mode:**

:w - Save/Write

:q – Quit

:wq – Write and Quit

:q! – Quit without saving

:w New.txt – Saveas

:1 To first line

:/word (Forward Search)

:?word (Backward Search)

n-> Next Occurance N-> Previous Occurance

:%s/old word/new word/ (will replace all the matched words from top to bottom)

:%s/old word/new word/ (will replace all the matched words from top to bottom)

**CLASS III**

**Concepts Covered : File, Meta Characters, File Permissions**

**File:**

There are three types of I/O, which each have their own identifier, called a file descriptor:

* standard input: 0
* standard output: 1
* standard error: 2

**1.Standard input :**

**Syntax**: $sort 0< input\_file.txt

* **$sort Names1.txt or $sort < Names1.txt**

Above command will sort the Names present in the file **Names1.txt** on Terminal

* **$ sort 0< Names1.txt**

Output same as above command **(i)** by Default before < operator 0 is there.

**2.Standard output:**

**Syntax:** $sort input\_file.txt 0< output\_file.txt

* **$sort Name1.txt > Sorted\_Names.txt**

Above command will sort the Names present in the file **Names1.txt** into **Sorted\_Names.txt**

* **$sort Name1.txt 1> Sorted\_Names.txt**

Above command will sort the Names present in the file **Names1.txt** into **Sorted\_Names.txt**….Same as command (**i**) by default 1 is before the “>”

**$ cat file1.txt file2.txt**

Above command will give the output on the Terminal by clubbing the two files **file1.txt** and **file2.txt**

**$ cat file1.txt file2.txt > files12.txt**

Above command will give the output in the **files12.txt** by clubbing the two files **file1.txt** and **file2.txt**

**$ ls –l file1.txt file2.txt > lsfiles.txt**

Above command will give the output in the **lsfiles.txt** by clubbing the permissions of **file1.txt and file2.txt**

**$ cat lsfiles.txt**

Above command will give the output permissions of **file1.txt and file2.txt on Terminal**

**$ ls –l file1.c file2.c > lsfiles.c**

Above command will give the output in the **lsfiles.c** by clubbing the two files permissions **file1.c** and **file2.c**

**$ cat lsfiles.c**

Above command will give the output permissions of **file1.c and file2.c on Terminal**

**3.Standard Error:**

**$ ls –l file1.c file2.c 2> lsfiles.err**

Above command will give output in **lsfiles.err** if any error come while executing the command those errors only will save in **lsfiles.err**

**$ cat lsfiles.err**

Above command will give errors if exits otherwise empty file

**$ ls –l file1.txt file2.txt file1.c file2.c 1> lsfiles.txt 2> lsfiles.err**

Above command will give two output files

**lsfiles.txt –** will save the file permissions data of **file1.txt file2.txt file1.c file2.c**

**lsfiles.err –** will save errors if any file is not exists or any errors

**Meta Characters:**

**Linux Regular Expression:**

Regular expression is also called**regex or regexp**. It is a very powerful tool in Linux. Regular expression is a pattern for a matching string that follows some pattern. Regex can be used in a variety of programs like grep, sed, vi, bash, rename and many more.

**Regular Expression Meta characters:**

A regular expression may have one or several repeating meta characters.

|  |  |
| --- | --- |
| **Metacharacter** | **Description** |
| **.** | Replaces any character. |
| **^** | Matches start of string and represents characters not in the string. |
| **$** | Matches end of string. |
| **\*** | Matches zero or more times the preceding character. |
| **\** | Represents the group of characters. |
| **()** | Groups regular expressions. |
| **?** | Matches exactly one character. |
| **+** | Matches one or more times the preceding character. |
| **{N}** | Preceding character is matched exactly N times. |
| **{N,}** | Preceding character is matched exactly N times or more. |
| **{N,M}** | Preceding character is matched exactly N times, but not more than N times. |
| **-** | Represents the range. |
| **\b** | Matches empty string at the edge of a word. |
| **\B** | Matches empty string if it is not at the edge of a word. |
| **\<** | Matches empty string at the beginning of a word. |
| **\>** | Matches empty string at the end of a word. |

**Regex Versions**

There are three versions of regular expressions syntax:

1. BRE : Basic Regular Expressions
2. ERE : Extended Regular Expressions
3. PRCE: Perl Regular Expressions

Depending on tool or programs, one or more of these versions can be used.

Examples:

Create files like below:

file1.txt

file2.txt

file3.txt

file12.txt

file23.txt

file31.txt

file13.txt

file32.txt

file123.txt

file231.txt

file321.txt

**$ ls \*.txt**

**Output** : **11 files**

file1.txt

file2.txt

file3.txt

file12.txt

file23.txt

file31.txt

file13.txt

file32.txt

file123.txt

file231.txt

file321.txt

**$ ls file\*.txt**

**Output: 11 files**

file1.txt

file2.txt

file3.txt

file12.txt

file23.txt

file31.txt

file13.txt

file32.txt

file123.txt

file231.txt

file321.txt

**$ls file1\*.txt**

**Output: 4 files**

file1.txt

file12.txt

file13.txt

file123.txt

**$ ls file12\*.txt**

**Output: 2**

file12.txt

file123.txt

**$ ls file123\*.txt**

**Output:**

file123.txt

**$ ls file?.txt**

**Output:**

file1.txt

file2.txt

file3.txt

**$ ls file??.txt**

**Output:**

file12.txt

file23.txt

file31.txt

file13.txt

file32.txt

**$ ls file???.txt**

**Output:**

file123.txt

file231.txt

file321.txt

**File Permissions:**

**Linux File Ownership:**

Every Linux system have three types of owner:

1. **User:** A user is the one who created the file. By default, whosoever, creates the file becomes the owner of the file. A user can create, delete, or modify the file.
2. **Group:** A group can contain multiple users. All the users belonging to a group have same access permission for a file.
3. **Other:** Any one who has access to the file other than **user** and **group** comes in the category of**other**. Other has neither created the file nor is a group member.

**File Permissions :**

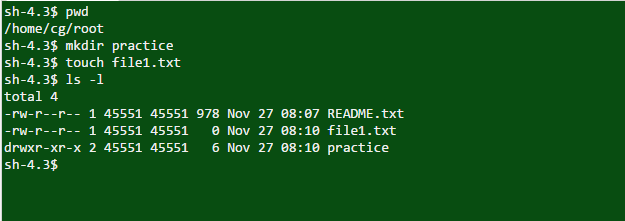
All the three owners (user owner, group, others) in the Linux system have three types of permissions defined. Nine characters denotes the three types of permissions.

1. **Read (r) :** The read permission allows you to open and read the content of a file. But you can't do any editing or modification in the file.
2. **Write (w) :** The write permission allows you to edit, remove or rename a file. For instance, if a file is present in a directory, and write permission is set on the file but not on the directory, then you can edit the content of the file but can't remove, or rename it.
3. **Execute (x):** In Unix type system, you can't run or execute a program unless execute permission is set. But in Windows, there is no such permission available.

**Permissions are listed below:**

|  |  |  |
| --- | --- | --- |
| **permission** | **on a file** | **on a directory** |
| r (read) | read file content (cat) | read directory content (ls) |
| w (write) | change file content (vi) | create file in directory (touch) |
| x (execute) | execute the file | enter the directory (cd) |
| a(all) | read file content (cat) ,  change file content (vi) ,  execute the file | read directory content (ls),create file in directory (touch),enter the directory (cd) |

**Permission Set:**



Look at the above snapshot, there are ten characters (-rw-r--r--) before the user owner. We'll describe these ten characters here.

**File permissions for (-rw-r--r--)**

|  |  |  |
| --- | --- | --- |
| **position** | **characters** | **ownership** |
| 1 | - | denotes file type |
| 2-4 | rw- | permission for user |
| 5-7 | r-- | permission for group |
| 8-10 | r-- | permission for other |

* Look at the above snapshot, different directories and files have different permissions.
* First letter (-) or d represents the files and directories respectively.
* Now, from remaining nine letters, first triplet represents the permission for user owner. Second triplet represents the permission for group owner. Third triplet represents the permission for other .

**Setting Permissions With chmod:**

You can change the permissions with chmod command accordingly to your need. Below are some examples to change the permissions for different groups.

**'+' - add permissions and '-' - remove the permissions**

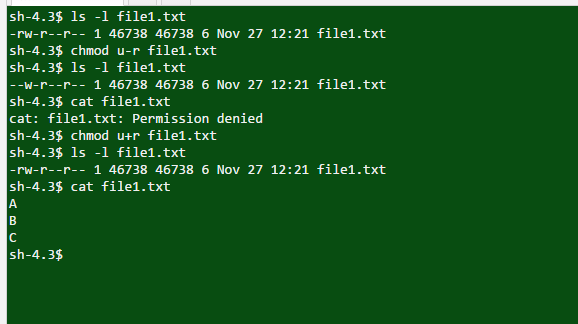
**To add permissions to a user/group/other:**

**Syntax:**

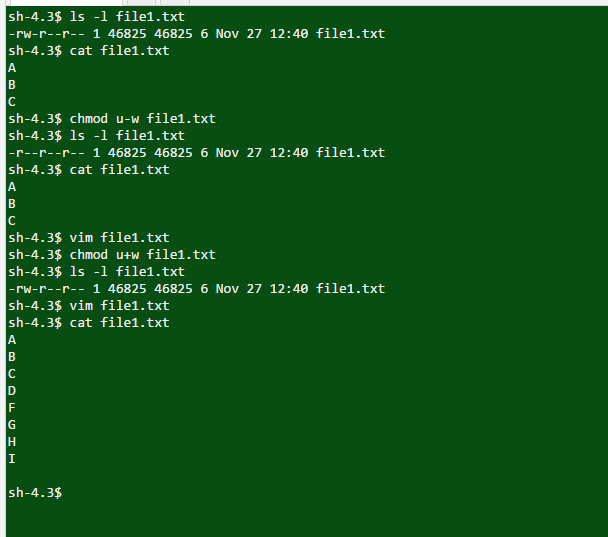
chmod **<username/groupName/otherName>**+**<permissionName>** **<fileName>**

**Ex**: chmod u+r file1.txt

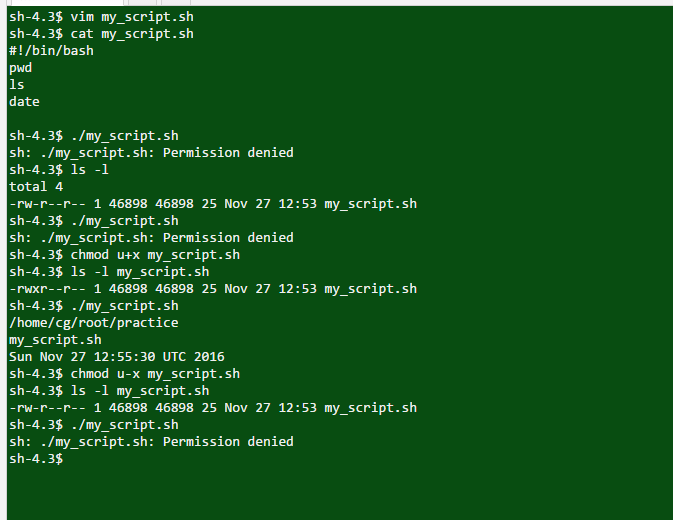
**File Read Permissions:**



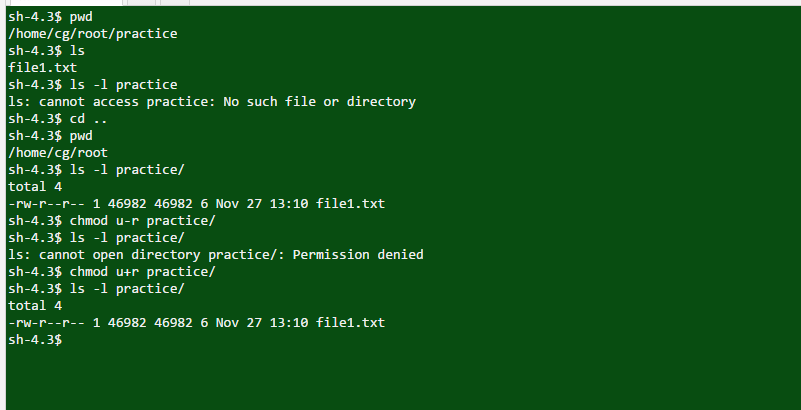
**File Write Permissions:**



**File Execute Permissions:**



**Directory Read Permissions:**



**…....And Other Operations of Directory is as like the file permissions**

**CLASS – IV**

**Find:**

This command lists out all the files in the current directory as well as the subdirectories in the current directory.

* $ find . -name filename.txt

./subdirectory/filename.txt

./subdirectory/filename.txt

./filename.txt

* Find command with relative path:

find ./subdirectory -name filename.txt

./subdirectory/filename.txt

* Find command with absolute path:

find ~/home/directory/subdirectory -name filename.txt

/home/directory/subdirectory filename.txt

**grep:**

Grep command is used to search the particular word/text in the file.

* **grep** [*options*] *PATTERN* [*FILE*...]

**File System:** Hierarchical representation of Files and subdirectories for particular Directory or location or path

**LINKS**

Two types of links are there in the links i.e,

1.hard link

2.soft or symbolic link

**hard link:**

* Hard link have same inodes
* Hard link have actual file contents
* It will work only with in the file system
* Cannot create hard link to a directory
* If original file removed will not effect the link file

$ ln [original/targetfile] [linkfile]

**Soft or symbolic link:**

* Soft link have different inode numbers
* Soft link contains the path to original directory not to file contents
* It will work across the file system
* Can create a link to directory
* If original file removed will effect the link file(Unable to access the link file)

$ ln -s [Original/targetfile] [linkfile]

**mount:**

This command will listed the mounted file systems.

* $ mount ----> Will list the mounted file systems
* $ mount -V ----> Will print the version

**Foreground and Background process**

As a multitasking operating system, Linux allows you run multiple processes in the background while you continue to work in the foreground.

**Background process:**

Shell does not have to wait for a background process to end before it can run more processes. Within the limit of the amount of memory available, you can enter many background commands one after another

To run a command as a background process, type the command and add a space and an ampersand to the end of the command. For example:

* $ command1 &

**Foreground process:**

Some foreground processes show the user an interface, through which the user can interact with the program.

The user must wait for one foreground process to complete before running another one.

Command for foreground process:

* $ command1

**Foreground to Background:**

Assume that one process is running in forground and want to keep that in background.

For ex:

* ~$ sleep 20

^Z

[1]+ Stopped sleep 20

$ bg

[1]+ sleep 20 &

After that user can run another process in terminal

**Background to Foreground:**

Assume that one process in background and want to bring that to foreground.

For ex:

* $ sleep 30 &

[1] 2614

$ fg

sleep 30

After that user con't run another process in foreground until process completed or terminated from the process

**Process Status:**

If user running mutiple number of process in background and user wants to see the status of that process then the command is

* $ ps ----> Will gives the process status
* $ ps -f ----->Will gives long listing of process
* ps -ef | grep [proces name] ---------->Will gives the particular process info

**Process Termination:**

If user wants to terminate from the process then user can terminate through

1. Normal termination
2. Termination through signals.

**Normal Termination:**

Run one process and after that terminate from that by using ctrl+c

* $ sleep 1000 &

[2] 3221

$ fg

sleep 1000

^C

$

Process got terminated

**Termination through signals:**

Command for showing all the signals on terminal or knowing all the signals

* $ kill -l

If should know pid (process id) for termiante the process using signals

* kill [-signal] [pid] or
* kill [-signal number] [pid]

**/proc:**

/proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc)

* /$ cd /proc

/proc$ ls

That will list the runtime system information

**/etc:**

/etc contains all system related configuration files in here or in its sub-directories. A "configuration file" is defined as a local file used to control the operation of a program; it must be static and cannot be an executable binary. For this reason, it's a good idea to backup this directory regularly.

* cd /etc

/etc$ ls

That will list the configuration files information

**at:** at will able to schedule the task and it will create as one job and run from the bash

User can able to create mutiple jobs

* $ at [time] [date]

[ctrl+d]

job created

**atq:** atq command will display the job user were created

* $ atq

will display all the jobs created by user

**atrm:** atrm is used to delete the job

* $ atrm [job number]

job will be deleted