*Understanding Absolute and Relative Paths

→ There are two ways to identify paths

- Absolute Pathname

→ An absolute Pathname begins with the root directory (/) and follows the tree, branch by branch

→ until it reaches the desired directory of file.

→ Absolute paths always start with /.

- Relative Pathname

→ A relative Pathname starts from the Present working directory.

→ Relative Paths never start with /.

→ multiple slashes (//) between directories and files are allowed, but all but one slash between element in Pathname is ignored by the system.

→ ////usr//bin is valid, it is seen as just /usr/bin by the system.

→ most of the time, It is most convenient to use relative paths, which require less typing.

→ usually, you take advantage of the shortcut provided by

- . (present directory)
- .. (parent directory)
- ~ (home directory)

→ For example, suppose you are currently working in your home directory and wish to move to the /usr/bin directory.

There are two ways

- Absolute Pathname method

  $ cd /usr/bin

- Relative pathname method

  $ cd ../../usr/bin

* Exploring the File System

→ Traversing up and down the file system tree can get tedious.

→ The tree command is a good way to get bird's-eye view of the file system tree. Use tree -d to view just directories and to suppress listing file names.

Command	Usage
cd /	Change current directory to root(/) directory
ls	List the content of the present working directory.
ls -a	List all files including hidden files
tree	Display a tree view of the file system

* What are links in Linux?

→ A connection between a file name and the actual data on the disk.

→ we call it a shortcut.

* Diffrence between Soft and Hard link

- Soft link :- Link will be removed if original file removed or deleted.

$ ln -s

- Hard link :- Renaming, Deleting or removing the file will not effect the link

$ ln

* Soft link

$ ln -s original location Linkname

If i don't give the link name then it will take filename

→ If you change anything inside original value it will also change Linkname

→ It is actually like Pointer

* Hard link

ln original file location Link name

* Navigating Through Directory History

→ The cd command remembers where you were last and lets you get back there with cd-.

→ For remembering more than just last visited you can use ~~pushd~~ pushd command and it will change directory also.

→ Using popd will then send you back to those directory.

* Working with Files

* Viewing Files

Command	Usage
cat	used for viewing files that are not very long. It does not provide any scroll-back
tac	used to look at file backwards starting with the last line
less	→ used to view larger files because it is a Paging Program. It pauses at each screen full of text. Provides scroll back capability.

	and lets you search and navigate within the file. NOTE :- Use */ to search for a pattern in forward direction ? for pattern in the backward direction.
tail	Used to Print the last 10 lines of a file by default. You can change the number of lines by doing -n 15 or just -15 if you wanted to look at the last 15 instead of the default.
head	The opposite of tail by default, it prints the first 10 line of a file.

* Touch

→ touch is often used to set or update the access, change and modify times of files.

→ By default it resets a file's timestamp to match the current time.

→ However, you can also create an empty file using touch

```
$ touch <filename>
```

→ touch provide several useful options. For example, the -t option allows you to set the date and timestamp of the file to a specific value as in

$ touch -t 12091600 myfile

→ This sets the myfile file's timestamp to 4.P.m, December 9th (12 09 1600)

* mkdir and rmdir

→ mkdir is used to create a directory

. mkdir Sampdir

→ It creates a sample directory named Sampdir under the current directory.

. mkdir /usr/Sampdir

→ It creates a smple directory called Sampdir under /usr.

→ Removing a directory is done with rmdir.

→ The directory must be empty or command will fail.

→ To remove a directory and all of its contents you have to do rm -rf.

* Moving, Renaming or Removing a File

→ Note that mv does double duty, in that it can:

- Simply rename a file
- move a file to another location, while possibly changing its name at the same time.

→ If you are not certain about removing files that match a pattern you supply, It is always good to run rm interactively (rm -i) to prompt before every removal.

Command	Usage
mv	Rename a file
rm	Remove a file
rm -f	forcefully remove a file
rm -i	Interactively remove a file

useful commands

* Renaming or Removing a Directory

→ rmdir works only on empty directories otherwise you get an error.

→ while typing rm -rf is a fast and easy way to remove a whole filesystem tree recursively

→ It is extremly dangerous and should be used with utmost care, espeially when used by root.

Command	usage
mv	Rename a directory
rmdir	Remove an empty directory
rm -rf	forcefully remove a directory recursively

* Modifying the command Line Prompt

→ The PS1 variable is the character string that is displayed as the Prompt on the command line.

→ Most distributions set PS1 to known default value, which is most suitable in most cases.

→ However, users may want custome information to show on the command line.

→ For Example some system administrators require the user and host name to show up on the command line as in

student@rg $

→ This could Prove useful if you are working in multiple roles and want to be always reminded of who you are and what machine you are on.

→ The Prompt above could be implemented by setting the PS1 variable to: \u@\h\$.

For Example

```
$ echo $PS1
\$
$ PS1 = "\u@\h\$ "
student@rg $ echo $PS1
\u@\h\$
student@rg $
```

→ By convention, most systems are set up so that root user has Pound sign (#) as their Trompt