GENERAL UNIX COMMANDS

1] pwd: It stands for Print Working Directory/Present Working Directory

The **pwd** command displays the full pathname of the current directory.

Syntax

```
pwd
```

Example:
$pwd
/home/pc10
$

2] Creating & Viewing Files

The 'cat' command is used to display text files. It can also be used for copying, combining and creating new text files.

To create a new file, use the command

1. cat > filename
2. Add content
3. Press **'ctrl + d'** to return to command prompt.

To view a file, use the command -

```
cat filename
```

The syntax to combine 2 files is -

```
cat file1 file2 > newfilename
```

To add content to existing file, use the command -

```
cat >>filename
```

Options

Option	Description
-e or -E	It prints $ mark to the end of each line.
-n	It numbers each line of file.

3] Deleting Files

The 'rm' command removes files from the system without confirmation.

To remove a file use syntax -

```
rm [option] file(s)/directory name(s)
```

Example:
1] rm sample.f - deletes sample.f
2] rm chap?.txt - deletes all files with chap as the first four characters of their name including with any one more character and having extension.txt.
3] rm -i * - deletes all files in current directory but asks first for each file
4] rm -r /olddir - recursively removes all files in the directory olddir, including the directory itself.

Options

Option	Description
-i(interactive)	This option removes file interactively.
-r(recursive)	It is used to remove non-empty directory, together with all the files and subdirectories contains.
-f(force)	This option is used to remove a file forcibly without displaying write protected message.

4] Moving OR Rename files

To move a file, use the command.

```
mv filename new_file_location
```

Suppose we want to move the file "sample2" to location /home/zsp/Document. Executing the command

mv sample2 /home/zsp/Document

For renaming file:

```
mv filename newfilename
```

mv sample.f sample2.f ***- moves sample.f to sample2.f***
mv dir1 newdir/dir2 ***- moves contents of directory dir1 to newdir/dir2***

mv -i file.1 file.new - ***prompts if file.new will be overwritten***

mv *.txt chapt1 - ***moves all files with .txt suffix to directory chapt1***

Options

Option	Description
-I (interactive)	It warns you before overwrite an existing file.
-f (force)	It suppress interactive overwrite message.

cp - copies files. Will overwrite unless otherwise specified. Must also have write permission in the destination directory.

cp sample.f sample2.f - ***copies sample.f to sample2.f***

cp -R dir1 dir2 - ***copies contents of directory dir1 to dir2***

cp -i file.1 file.new - ***prompts if file.new will be overwritten***

cp *.txt chapt1 - ***copies all files with .txt suffix to directory chapt1***

cp /usr/doc/README ~ - ***copies file to your home directory***

cp ~betty/index . - ***copies the file "index" from user betty's home directory to current directory***

Options

Option	Description
-I (interactive)	Destination file is already exist and user want to show an overwriting prompt for an existing file during copy then –I option is used.
-r (recursive)	The – r option recursively copy an entire directory structure to another directory.
-p (preserve)	Copy operation changes access and modification time. This option preserve access and modification date time of file.

Option	Description
-l (link)	This option creates a link instead of copying a file.

5] Creating Directories

Directories can be created on a Linux operating system using the following command

```
mkdir directoryname
```

This command will create a subdirectory in your present working directory, which is usually your "Home Directory".

For example,

```
mkdir mydirectory
```

The **mkdir** command creates a single directories or multiple directories.

Syntax

The syntax for the **mkdir** command is:

```
mkdir [options] directories
```

Options

Option	Description
-m mode	Sets the access mode for the new directory.
-p	If the parent directories don't exist, this command creates them.

Example

```
mkdir -m 444 tech
mkdir -p d1/d2/d3
```

6] Removing Directories

To remove a directory, use the command -

```
rmdir directoryname
```

Example

```
rmdir zsp
```

will delete the directory zsp

Options

Option	Description
-p	This option remove directories and any intervening parent directories that become empty as a result useful for removing subdirectory trees.
-v	Rmdir do not display any message after removal of empty directory.

7] The 'Man' command

Man stands for manual which is a reference book of a Linux operating system. It is similar to HELP file found in popular software.

To get help on any command that you do not understand, you can type

```
man
```

The terminal would open the manual page for that command

8] The History Command

History command shows all the commands that you have used in the past for the current terminal session. This can help you refer to the old commands you have entered and re-used them in your operations again.

9] The clear command

This command clears all the clutter on the terminal and gives you a clean window to work on, just like when you launch the terminal.

10] 'pr' command

This command helps in formatting the file for printing on the terminal. There are many options available with this command which help in making desired format changes on file. The most used '**pr'** options are listed below.

Option	Function
-x	Divides the data into 'x' columns
-h "header"	Assigns "header" value as the report header
-t	Does not print the header and top/bottom margins
-d	Double spaces the output file
-n	Denotes all line with numbers
-l page length	Defines the lines (page length) in a page. Default is 56
-o margin	Formats the page by the margin number

11] Printing a file

Once you are **done with the formatting,** and it is time for you to get a **hard copy** of the file, you need to use the following command:

```
lp Filename
```

or

```
lpr Filename
```

In case you want to print multiple copies of the file, you can use the number modifier.

12] Listing files (ls)

It shows the files /directories in your current directory.

Syntax

The syntax for the **ls** command is:

```
ls [options] [names]
```

ls abc* # list all files starting with abc---
ls *abc* # list all files containing --abc--
ls *abc # list all files ending with --abc

The Unix command to return the files that ends with single digit and has .txt extension

ls -l *[0-9].txt

'ls -al' gives detailed information of the files. The command provides information in a columnar format. The columns contain the following information:

1st Column	**File type and access permissions**
2nd Column	# of Hard Links to the File
3rd Column	Owner/the creator of the file
4th Column	Group of the owner
5th Column	File size in Bytes
6th Column	Date and Time
7th Column	Directory or File name

```
ls -a
```

Listing Hidden Files

Hidden items in UNIX/Linux begin with .(period) symbol at the start, of the file or directory.

Any Directory/file starting with a '.' will not be seen unless you request for it. To view hidden files, use the command.

The **ls** command lists all files in the directory that match the *name*. If name is left blank, it will list all of the files in the directory.

Options

Option	Description
-a	Displays all files.
-b	Displays nonprinting characters in octal.

Option	Description
-c	Displays files by file timestamp.
-C	Displays files in a columnar format (default)
-d	Displays only directories.
-f	Interprets each *name* as a directory, not a file.
-F	Flags filenames.
-g	Displays the long format listing, but exclude the owner name.
-i	Displays the inode for each file.
-l	Displays the long format listing.
-L	Displays the file or directory referenced by a symbolic link.
-m	Displays the names as a comma-separated list.
-n	Displays the long format listing, with GID and UID numbers.
-o	Displays the long format listing, but excludes group name.
-p	Displays directories with **/**
-q	Displays all nonprinting characters as **?**
-r	Displays files in reverse order.
-R	Displays subdirectories as well.
-t	Displays newest files first. (based on timestamp)
-u	Displays files by the file access time.

Option	Description
-x	Displays files as rows across the screen.
-1	Displays each entry on a line.

13] cd

The change directory (**cd**) command is used to move one directory to another directory.

Syntax

```
cd [path name/directory name]
```

Examples of CD commands

cd Changes to user home directory
cd / It changes directory to system root
cd . this represents the current directory
cd .. this represents the parent directory or move one level up
cd ../.. Move two directory level up
cd ~ Switch to home directory of the user, similar to cd without argument

cd ./filename Add the child directories

Standard Abbreviation for directories are as follows

.(dot)=> indicates current directory

..(two-dots)=> used to move to its parent directory

/(slash)=> for root directory

~(tild) =>for home directory

14] chmod command

The **chmod** command changes the access mode of one file or multiple files. It works in two modes – Symbolic or Relative and Absolute Mode

Syntax

The syntax for the **chmod** command is:

```
chmod [option] mode files
```

1] **Using chmod in Symbolic Mode:**
The easiest way to modify file or directory permission is to use the symbolic mode. With this, we can add, delete or specify the permission by using following set of operators.

Chmod operator	description
+	Add given permission to file/directory
-	Remove given permission to file/directory
=	Set the permission

2] Using chmod with absolute permission:
The second way to modify permission with chmod command is to use a number to specify each set of permission on the file.
Each permission is assigned a value as per following table and the total of each set of permission provides a number for that set.
Example:
$chmod 777 f1
ls –l f1
rwx-rwx-rwx 1 bca bca 1012 Nov 19 10:00 f1

Number	Octal permission Representation	Reference
0	No permission	**---**
1	Execute permission	**--x**
2	Write permission	**-w-**
3	Execute and write permission:1(execute) + 2(write)=3	**-wx**
4	Read permission	**r--**
5	Read and Execute permission: Read(4) + Execute(1)=5	**rw-**
6	Read and write permission:Read(4) + Write(2)=6	**rw-**
7	All permission 4(read) +2 (Write)+1 (execute) =7	**rwx**

Example

```
chmod 751 tech

chmod u=rwx, g=rx, o=x tech

chmod =r tech
```

Options

Option	Description
-R	Descend directory arguments recursively while setting modes.
-f	Suppress error messages if command fails.

15] passwd

The **passwd** command changes the password for the *user*.

Syntax

```
passwd [options] [user]
```

Option	Description
-s	Displays password information.

16] more

The **more** command displays the file called *name* in the screen. The RETURN key displays the next line of the file. The spacebar displays the next screen of the file.

Syntax

The syntax for the **more** command is:

```
more [options] [files]
```

Options

Option	Description
-c	Page through the file by clearing the window. (not scrolling).
-d	Displays "*Press space to continue, 'q' to quit*"
-f	Count logical lines rather than screen lines (wrapping text)
-l	Ignores form feed (^L) characters.
-r	Display all control characters.
-s	Displays multiple blank lines as one blank line.

Option	Description
-u	Does not display underline characters and backspace (^H).
-w	Waits for a user to press a key before exiting.
-n	Displays n lines per window.
+*num*	Displays the file starting at line number *num*.
+/*pattern*	Displays the file starting at two lines before the *pattern*.

Example

```
more -d tech
```