

• rm -r <<directory>>

remove empty directory.

NOTE:- . . . dot denote current directory/location whereas
- dash denote last directory.

File extension

• .txt

• .mp3

• .jpg

• .avi

File type

text file

audio file

pictures

videos.

→ Matching file name by path name extension:-

→ Pattern matching:-

In this what we do is, ~~that~~, we search for file with some pattern name.

We use different pattern with 'ls' command to search for files

with same pattern of name.

Pattern	matches
* ?	replace any number of characters replace single character.
[abc...]	File name starting with any one character in the enclosed class.
[!abc...]	File name starting with not any one character in the enclosed class.
[[:alpha:]]	File name starting with alphabetic character.
[[:lower:]]	File name starting with alphabet lowercase character.
[[:upper:]]	File name starting with uppercase character.
[[:alnum:]]	File name starting with any alphabetic character or digit.

[[punct:]] file name starting with only printable character not a space or alphanumeric.

Assume we are in a directory with many files in it.

- If we want to search a file starts with 'a' we use Command

`ls a*`

It will list all file with name 'a'

Initially -

- If we want to search a file name starts with 'a' but have specific (say 4) no. of character in it, we use Command

`ls , a????`

It will list all files, name starts

with a and have 3 more character after that.

- If we want to list all files name starts with 'a' or 'c' we use command

`ls [a c]*`

→ Brace expansion :-

Brace expansion is used when we want to recall or create files/directories with similar type of name, only differentiable by prefix (like 1-9, a-z etc.).

ex. -

music.1.mp4 music2.mp4

We can create as many number of file or move them by typing their name, but brace expansion makes it easier. You can call all ~~same type~~ similar name files in one call.

Creating the files from Song1.mp3 to Song10.mp3 by brace expansion.

- touch Song1..10.mp3

Creating files with 2 brace expansions.

~~touch Song1..3~~

- touch Song1..332a3.mp3

it will create file

Song1a.mp3 Song2a.mp3 Song3a.mp3

- touch Song1,232a,b3.mp3

it will create file

Song1a.mp3 Song1b.mp3 Song2a.mp3
Song2b.mp3

- touch text11a..c3, 2, 33.txt

it will create file

text1a.txt text1b.txt text1c.txt
text2.txt text3.txt

Date _____
Page _____

We can also use the brace expansion method with commands like `cp`, `mkdir`, `mv`, `rm` etc. instead of multiple similar type file name.

Ex -

```
mkdir ../Real1..33
```

It will create 3 directories

Real1/ Real2/ Real3/

→ Variable expansion:-

Variable is named container that can store a value in memory. It makes easy to access and modify the stored data whenever we want.

Syntax:-

VARIABLENAME = value

To access the data inside variable, we use `echo` command.

```
echo $VARIABLENAME
```

Its result will be value.

here, \$ is used to denote that after that next word is variable name which value we want.

To avoid mistakes for bigger variable name or over shell expansion we use `{ }` around variable.

`echo ${VARIABLENAME}`

Example of variable expansion.

`USERNAME = Anuj`

`echo Name of the User is ${USERNAME}`

Result:-

Name of the User is Anuj

→ Command Substitution :-

Command Substitution allows the output of a command to replace the command itself on the command line.

Command substitution occurs when a command is enclosed in parentheses, and preceded by a dollar sign (\$). \$ can nest multiple command expansion inside each other.

ex. -

```
echo The time is $(date +%I:%M)  
minutes past $(date +%I:%Y:%P)
```

output

The time is 26 minutes past 11AM.

→ protecting argument from expansion -

If we want to print \$HOME without expansion in command line, we use escape character backslash (\) before that.

It will protect the character immediately following it from expansion.

ex. -

```
echo my name is $user
```

output - My name is Anuj

echo My name is \ \$user
output- My name is \$user

We also use single quotes (') and double quotes (") to protect longer character strings.

Both ' and " have different effect,
(') Stop all shell expansion while
(") allows command and variable
~~expand~~ substitution.