date

Local date and time

Options:

-u -> time in GMT

+%d%m%y ->date,month,year

cal

No -> calendar &mon of current year

-y: 1 –> current year

2 –>the specified month & year

-n → ($n/2$ prev)+(this month)+($n/2$ future months)

who

users currently logged in

(id, terminal, time)

options:

-u ->( )+ idle time, process-id

-H -> header that explains each column(3)

-uH ->header with all 5 col

Idle time -> <24 time is mentioned

>24 old

Active .

whoami

to see information about ourselves

banner

prints the specified output in large size

Ex:$ banner Good Morning

Good Morning

passwd

To change password

echo

Acts as a printf and cout

Ex: $ echo message

Message

man / help

FOR shell builtin	exe cmd
help cd	man cp

online documentation of the topic

whatis - brief description of exe cmd

which

-absolute path of executable command

Ex: $ man topic

tty:

Names the terminal we are using

clear:

Clears the screen

exit

To exit terminal

script:

To record the section

Ex:

$ script ☐ → stores in file ☐
↳ not present - stored in typescript

____

____

$exit

→ to append use $ script -a

uname:

To see details about UNIX system.

bc

Unix into calculator

To set scale: scale=n

To specify the base of

i) Input -> ibase

ii) Output -> obase

To end : ctrl+d

FILE NAMES:

- alphabet -case sensitive
- can have (.) (-)(not in beginning) (_) only

.filename → hidden file

Filename. Extensions

WILD CARDS:

? Match any single character

[.....] Match a single character from the set Ex: [xyz] : matches x, y, z

* Match zero or more characters

Ex: [[:upper:]] * → all uppercase letter

[[:digit:]] → digits

[[:alpha:]] → not alphabet

[[:alnum:]] = [A-Za-z0-9]

OPTIONS UNIQUE TO DIRECTORIES:

pwd : location of current directory in file system

mkdir:
[$ mkdir filename]
creates a new directory
OPTIONS:
-m → control permission mode
-p → creates a parent directory & sub directory in the path specified
Ex: $ mkdir -p solarsystem/planets/Saturn
(existing)

rmdir dir_name
to remove the specified directory
[Cannot remove directory if it is not empty]

cd: cd path : to specified dir
to change the directory
$ cd ../ -> to move to parent directory
$ cd / → directly to root dir
cd ~ → to home dir
cd - → to prev working dir

ls:
lists the file & directory names in alphabetical order

OPTIONS
-l → long listing
(permission |links |owner name |group name |size in bytes| last modified |filename)

-ld → displays only working directory

-nd → to list the user and group id instead of user and group name

-r → sorts the file in descending order

According to time:
-lt → latest file first
-lc → sequence of last access

-i → inode no.
-a → hidden files
-R → recursive [all files within folder]

-p → used to identify directory(/) and file (not preceded by anything)

-lc → lists the files by inode date changes

-Rp → to see compete file system

-1 → prints the file names in one column

-li → prints the inode number with long listing

ls .. → content of parent directory

OPTIONS UNIQUE TO FILES:

lpr
Print out
Ex: $ lpr file1 file2

TO DISPLAY FILE CONTENTS:
1) cat :
$ cat filename $cat f1,f2 f3
2) more:
Ex:
$ more -ds -6 +49 filename
Displays 6 lines starting from 49th line . it also displays % lines displayed so far.

TO EDIT FILE:
$ vi filename
$ sed filename

TO CREATE FILES:
1) cat:
$ cat >filename
Crtl +d (to save)
2) $ vi >filename

3) touch
touch newfile
→ change time stamp if file already exists

OPERTIONS COMMON TO BOTH:

-i → Asks if we want to delete an existing file

COPY COMMAND:

$ cp options source destination

-p → permissions of existing destination can be changed to match those of the source file

-R/-r → recursive copy to copy a collection of files

MOVE COMMAND:

$ mv options file1 file2

File1 after this option will be gone

-f → to skip iterative message
Right protected file

RENAME COMMAND:

→ copy command is nothing rename
mv filename newFileName

REMOVE COMMAND (rm)

-f : removes file even if it is write protected

-R/-r : removes all files and empty directory from the path specified file (first file and then directory)

Write protected file found →
Remove asks for confirmation

-v = verbose - summary of what command did

Cat with visual options

- -v : see control characters
- -ve : $ at the end of each line
- -vt : tabs are printed as ^I
- -vet : No printable characters ^
- -n : displays numbers / line no.

· Inode Number:
Every file in sys has an inode

Inode : personal ID
has all info except file content & na

↳ inode no. - file type
- file size - number of links
- owner info
- permission

LINKING: - cannot be used with directori
- same file size

copy
HARD LINKS: different name of same link

$ ln file1 filename

--> same link and same inode number for both the files

```
29428 _________ file1
29428 _________ file2
```

shortcut
SOFT LINKS: Symbolic link = short cut
= smaller size
= useless when origi
file del

$ ln -s file1 file2

- Different inode number but will look like this when ls command is used

```
29428 _________ file1
29430 _________ file2→file1
```

- can be done for directories also

CHANGING FILE PERMISSION

who operator permission

$ chmod options mode file/directory

Options: -R → recursive – changes permission of all files and directories

1) SYMBOLIC

$ chmod category operation permission f1

Category: u, g, o, a (user / group / other / all)

Operation: +, - (change 1/2) = (new permission replaced)

Permission : r, w, x

2) OCTAL

Completely represent all of the permission (read)(write)(execute)

CHANGING OWNERSHIP and GROUP OWNERSHIP

$ chown newowner[: group] file

FILTERS:

PIPE $ command1 | command2 (o/p → I/p)

when you pipe two commands the output of 1st command acts as the input for the next command

head

$ head option filenames

=> specified no of lines from the beginning of one or more files

=> default 10 lines ⇒ no file - from input

option: -N number of lines from top

tail

$ tail options filenames

=> specified no of lines from the end of file

=> f=default last 10 lines

Range of lines [8-13]

$ head -13 file1 | tail +8

OPTIONS:

-N → copies last n lines

-c → count by character

$ tail -c30 file1

-b → count by disk block

-r → output in reverse order from bottom to top

-l → count by line

+N = -n+k → skips (N-1) lines and copies the rest to end of file

CUT AND PASTE:

1. $ cut option filename

OPTIONS:

-c → character

$ cut -c1-10,20-24 file1

-f → field

$ cut -f4,5-7 fie1

-d → delimiter

$ cut -f1.3-5 -d "/" file1

-s → supress output is no delimiter in line

2. $ paste options input-filenames

$paste file1 file2 (horizontally)

$paste -d $\underbrace{\text{"\textbackslash t\$"}}_{\text{delimiter}}$ file1 file2 file3

COUNT CHARACTERS, WORDS ,LINES

wc

$wc options input_files

Lines words character filename

-l → no of lines and filename

-c → no of character

-w→ no of words

-L → no of chars in longest line

groups - determine user's group

↳ your group

↳ with usrid - user's group

<u>change group</u>

with changing owner

chgrp new-group filename

<u>user masks</u> - <u>permission set</u>

<u>Mask</u>	0	1	2	3	4	5	6	7
dir	7	6	5	4	3	2	1	0
file	6	6	4	4	2	2	0	0

(read)(

umask
current permission

umask 022
↑
changes permission to
022

FILES WITH DUPLICATE LINES

$ uniq options filename

OPTIONS:

-u → to print only unique lines
-d → to print only duplicate lines
-c → counts the duplicate and unique lines
-f → skips the mentioned no of fields and compares after that

```
$ uniq – d – f 4 file1
```

-s → skips leading characters and then compares

```
$ uniq – d – s 5 file1
```

SORT :

$ sort options field_specifiers input_files

Field specifiers: +n1 –n2
n1- no of fields to be skipped
n2 – column to be sorted

$\boxed{1} , \boxed{2} , \boxed{3} , \boxed{4}$ delimitre

OPTIONS:

-c → checks if the file is sorted returns 1st out of seq. line
-t → alternate delimiter

```
$ sort – t '&' +1 -2 file1
```

-n → numeric sort
-r → reverse
-m → merges two sorted file
-u → unique sort (removes repeated lines)
-b → ignores leading blanks
-d → dictionary sorting (spe.cha – upp - lower)
-f → fold lower case(ignores difference b/w lower and upper case)

tee command:

copies standard input to standard output and at the same time copies it to on e or more files

$ tee options filename

$$\begin{tikzcd} \text{input} \arrow[r] & \boxed{\text{tee}} \arrow[r] \arrow[d] & \text{monitor std.output} \\ & \text{files} & \end{tikzcd}$$

OPTION:
-a appends to the existing file rather than deleteing the present contents.

tr: TRANSLATING CHARACTER

$ tr options exp1 exp2 standard_inputs

```
$ tr "aeiou" "AEIOU" this is my notes
                 => th I s Is my nOtEs
$ tr "aeiou" "AEIOU" < file1
$ tr "aei" "AEI" < file1 | head -3
```

- Each char in user specified set of character is replaced by corresponding char in 2nd string

exp1>exp2 : unmatched characters will be changed to last character in exp2
exp2>exp1 : extra characters in exp2 are ignored

OPTIONS:

-d – deletes the characters
Ex: $ -d "aeiou" < file1
Deletes all the mentioned characters fror the file

-s – compressing multiple consecutive character
Ex: $ tr -s 'a' < file   saaaaadaaa
                                   sada

-c – complement
Ex: $ tr -c "aeiou" "*" < file1
Accept aeiou replace all by *

-d – delimiter (used with -cd)

```
tr -s "ie" "dd"
   ie → dd & consecutive d are compressed with single d
```

comm - finds lines identical in two files & displays $\boxed{\text{uniq line of file1}}$ $\boxed{\text{uniq line of file2}}$ $\boxed{\text{line in both file}}$

COMPARING FILES

compare - cmp → Byte no. of first difference

cmp option f1 f2

-l : all difference found in file byte by byte
-s : no output
exit status = 0 ; files identical
= 1 ; atleast 1 B differ

difference diff

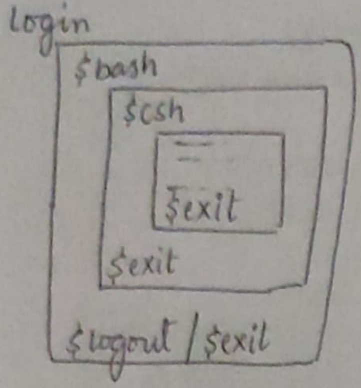
line by line difference b/w 2 files

-b : ignore blanks
-w : ignore whitespace
-i : ignore case

LOGIN shell verification

$echo $SHELL
  -path to login shell

$echo $0
  - what cur shell is



STANDARD STREAMS

standard	input - 0
	output - 1
	error - 2

Redirecting Input

command 0< file1
command < files

Redirecting Output

com 1> f1 ; → not present - creates & writes ; noclobber - on - error msg

com 1>| f1 ; empties & then write

com 1>> f1 ; append to output file

Redirecting Error 2>

Redirecting to different files

com 1> file1 2> file2

Redirecting to One file

com 1> file 2>& 1

Command Execution

1. Grouped Commands
2. Sequenced Commands
3. Chained commands - pipes
4. Conditional Commands

$\boxed{1}$ && $\boxed{com}$ — 1 is successful

com executes only if

$\boxed{1}$ || $\boxed{com}$ — 1 fails

Command Substitution

provides capability to convert the result of command to string

$(date) (Korn - bash shell)   'date' (c shell)

$echo hello! Date & time are: $(date)

→hello! Date & time are: Apr 21 2021 10:05:00

ALIASES

Creating customized commands by assigning a name to command.

alias dir=ls
alias dir='ls-l'
alias dir="ls-l | more"

alias fl ="ls-l'
fl f*

listing alias : alias

Removing alias

unalias alias_name
↓
[-a] option removes all alias

Ex: alias invent = "cd Desktop; mkdir dir1; touch file1"

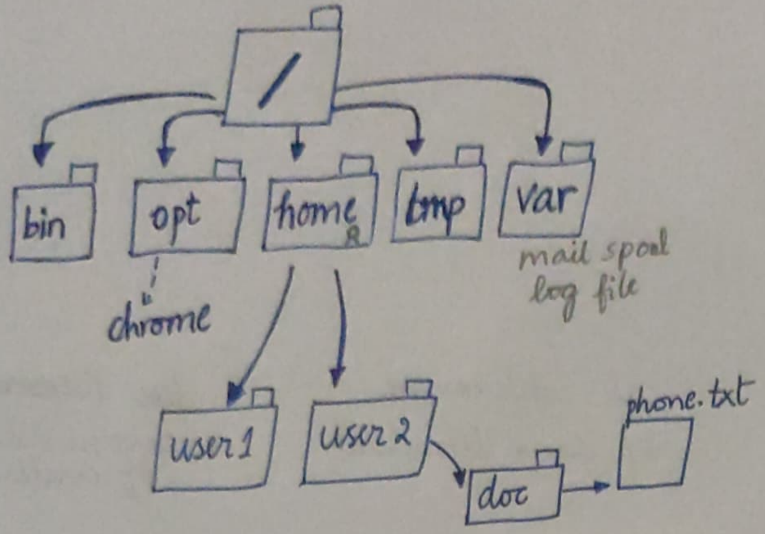
alias openFile= 'cd Desktop;
cd Todo;
gedit todo

alias del = "rm -i"
↳ del file_name

↳ will be stored in .bashrc

Linux Folder System

. current directory
.. parent directory



Access file : **Absolute path**
begins with root directory
home / usor2 / doc / phone.txt

Relative Path
starts with current working directory
usor2/ doc / phone.txt
./ doc / phone.txt

Linux command basics
- No concept of file extensions ∴ more flexibility dealing with files

img. png = img. baba

No know what type of file

```
file your_file_name
```

SPECIAL CHARACTERS
$ > < & | ;
" ' \

space in file names

```
mkdir my-cat
mkdir "my cat"
mkdir my\cat
mkdir my\_\_cat
```

special characters in filename

```
mkdir \$dollors\>\'you
mkdir cats\&dogs
```

gedit = **graphical text editor**

```
gedit  file_name
```

Command line editor

nano filename

To just view text files

less file_name
- separate file to see contents

cat filename
- on same terminal

tac filename
- reverse order of content

cat filename1 file_name2
- content of both files

TYPES OF COMMANDS

1) Executable commands (cp) (date)

2) Shell builtins (type)

3) Shell scripts (bzdiff) (bzexe)

4) Alias (ls)

→ type command_name

file filepath-from_last_cmd

Execute multiple commands

→ command1; command2; command3

→ cmd1 && cmd 2