14.split Command:

- The split command is used to divide a single large file into several ones.
- The file is split into smaller files with equal number of lines in each file, regardless of the file contents.

Syntax: `split -n file outfile.`

- The file is split into lines of 1000 by default. Each file is named with xaa, xab,xac appended to it and so on lexicographically up to zz. If no output is given x is default. By integer n the number of lines in the output file can be controlled.

```
$ split -4 file1
$ ls
xaa
xab
xac
$_
```

file1 has 12 lines is split into three files xaa, xab, xac of 4 lines each or

```
$ split -4 file1 num
```

The file "file1" of 12 lines is split into three files numaa, numab, numac of 4 lines each.

15.cmp Command: Comparing two files

- Frequently you may require knowing whether two files are identical in all respects so that one of them can be deleted.
- The cmp (compare) command is used to achieve this task.

```
$ cmp chap1 chap2
chap1 chap2       differ : char 9, line1
$_
```

- The two files are compared byte by byte and the location of the first mismatch is echoed to the screen. If the two files are identical then cmp displays no message, but simply returns the $ prompt.
- The –l (list) option gives detailed list of the byte number and the differing bytes in octal for each character that differs in both the files.

```
$ cat file1
abcd
xyz
$ cat file2
abed
wxy
$_
```

14/3

```
$ cmp -l file1 file2
3     143   145
6     170   167
7     171   170
8     172   171
$_
```

16. File differences with diff:

- diff is the third command which can be used to display the differences.
- It produces a detailed output.

Example:

```
$ diff file1 file2
oa1
>barunsengupta
2c3, 4
-------
>anilagarwal
>chakroborthy
$_
```

- The instruction oa1 indicates that a single line has to appended after line number 0 of the first file and the resultant line will have line number 1 and 2nd file.

17. Further comparisons with comm Command:

- Comm compares two sorted files and compares each line of the first file with its corresponding line in the second.
- Its displays a three columnar output. The first column contains lines unique to the first file, while the second column shows the lines unique to second file. The third column displays lines common (hence its name) to both files.

```
$ cat file1
shukla p.k.
chanchal singh
s.n.dasgupta
chakroborthy
$_
$ cat file2
barun sengupta
shukla p.k.
```

```
anil agrawal
chowdhary
s.n.dasgupta
$_
$ comm file1 file2
                barun sengupta
                                shukla p.k.
                anil agrawal
chanchal singh
                chowdhary
                                s.n.dasgupta
chakroborthy
$_
```

- first column contains two entries i.e. the lines common only to file1 (chanchal singh and chakroborthy) while second column shows three, you can see that shukla p.k. and s.n.dasgupta, who are present in both files, have been displayed in the third column.
- comm can also produce selective output using the option –1, –2 and –3.
- To drop a particular column simply use its column number with the –sign. Thus to select only those lines which are not common to both files you should exclude the third column from the output:

```
$ comm -3 file1 file2
            barun sengupta
            anil agrawal
chanchal singh
            chowdhary
chokroborthy
$_
```

- You can also combine option and display only those lines which are common:

```
$ comm -1 2 file1 file2
shukla p.k.
s.n.sengupta
$_
```

18. file Command: File types

- There are basically three types of files (ordinary, directory and device).
- Unix provides the file command to determine the type of file.

```
$ file emp.lst
emp.lst :ascii text
$_
$ file bin
/bin : directory
```

- When the command is used to apply to all files in the progs directory, you will see an informative list.

```
$ file progs/*      (* indicates all files)
abcd          :     ascii text
calender      :     english text
compchk.sh    :     commands text
helpdir       :     directory
text1         :     data
note          :     empty
cat           :     separate executable
prog1.c       :     c program text
$_
```

- This command has a built in mechanism of identifying the type of file by context. It identifies text files, shell programs, C and fortran programs, documents, executables, directories, empty files and data.
- Option: -f (input file) When a list of files to be classify this option is used.

  i.e. a text file names in your current working directory contains the following file names.

```
/bin , /bin/ls, /dev, /usr/abc.c
```

- Enter the following command

```
$ file -f names
```

- Here, the file command will read every file name and classify it

```
output:      /bin : directory
            /bin/ls : executable
            /dev : directory
            /usr/abc.c : c program text
```

19. chmod Command: File access permissions

Three types of file accesses can be granted or denied.

Read	A file can be read, displayed on a terminal, copied, complied.
Write	A file can be read, modified and deleted.
Execute	A file can be executed and deleted.

Change mode (chmod) command sets or changes the file access permissions for a file.

Syntax: `chmod mode file(s)`

- The mode is the permission to be assigned. The mode can be specified in two ways:

(i) Symbolic Mode: This mode uses letters to represent different permissions.

Symbol	Meaning
R	Read
W	Write
x	Execute or search

(ii) Symbolic entities: There are different groups of users of a system. u, g, o, a specify the categories of users.

Symbol	Meaning
u	Owner
g	Members of the same group
o	All other users
a	All users

Syntax:

```
chmod [who] [+ - =] [permission] file(s)
```

The who could be either g (for group), o (for others), u (for user or owner)

The + sign adds a permission, – sign removes a permission,

And = assigns the specified permission and removes all other permissions.

Example:

`$ chmod o-rw poem1`	Removes read and write permission for others for file poem1.
`$chmod a+r textfile`	Adds read permission for all users for file text file.
`$chmod g-w textfile`	Removes write permission for the group accessing text file.
`$chmod a=rwx myfile`	Assigns read, write and execute permissions for all the users of the file myfile.
`$chmod ug+rw, o-r myfile`	adds read, write permission for owner & group to which the owner belongs, removes read permission for users other than owner & group members for "myfile".

Absolute mode (The Octal Notation):

- With the absolute mode, permission for users, even for those that are not being changed have to be specified. A series of octal digits are used to represent each of the permission.

Value	Meaning
4	Read
2	Write
1	Execute

- The octal values are added together to give the actual permissions. To calculate the permissions add the octal values for each group of users.

Value	Permissions
1	Execute permission
2	Write permission
3	Write and Execute permission
4	Read permission
5	read and Execute permission
6	Read and Write permission
7	Read, Write and Execute permission

- To set access permissions specify octal values instead of symbolic values.

Syntax: `chmod [ugo] file(s)`

Example:

```
$ chmod 750 sample
```

- This example changes access permission for the file sample in the current working directory. The '7' gives read, write and execute permission for owner. The '5' gives read and execute permission for group and '0' gives no permission for other users.

20. grep Command: Searching for a pattern:

- Searches files for a pattern.

  Syntax: `grep<option><pattern><filename(s)>`

- This command sometimes referred to as the global regular expression printer.
- Options used by the grep family:

option	Significance
-c	Display count of the number of occurrences
-l	Display list of filenames only
-n	Display line numbers along with the lines
-v	All lines but those matching are displayed
-i	Ignores case for matching
-s	Ignore case
-e	Matching multiple pattern Grep –e "Agrawal" –e "aggrawal" –e "agarwal" student

Example:

```
$ cat student
Geeta       12123    female    thane
Mahesh      24321    male      mumbai
Vijay       43231    male      mumbai
Ajay        54252    male      thane
Sanjay      65421    male      thane
Seeta       87652    female    mumbai
$_
```

- To search for sanjay

```
$ grepsanjay student
Sanjay      65421
$_
```

```
-c (count) option counts the occurrences
$ grep -c 'thane' student
3
$_
```

- The -n(number) option can be used to display the line numbers which contains the patterns along with the lines

```
$ grep  -n 'female' student
1 :Geeta          12123      female      thane
6 :Seeta          87652      female      mumbai
$_
```

- The -l (list) option displays only the names of files where a pattern has been found.

```
$ grep -l 'thane' *.lst (or *.*)

student
list.lst
$_

$ grep '[cC]ho[wu]dh*ary' emp.lst
4290  | jayantchoudhary | executive | 6000
6521  | lalitchowdury   | director  | 8200
$_
```

21.nl Command:

- The nl command is used to add line numbers to a file.
- The line number records the relative position of a line in a file.

Syntax: `nl [-i incr] [file]`

Example:

```
$cat file1
jasmine
rose
lily
$nl file1
1 jasmine
2 rose
3 lily
$
```

- The –i option is used to specify the value, other than 1, by which line numbers must be incremented.

22.pr Command:

- The pr command prints a file in an unformatted form.
- The pr command is used to format a file suitably before printing.

Syntax: `pr [options] [files]`

- Each page formatted by pr begins with a five line header and ends with a five liner footer.
- The header consists of two blank lines, a line containing the page number, date and time and the filename followed by two blank lines.
- The pr sends the output to the standard output. By routing the output of pr to *lp* using a pipe, the file formatted by pr can be sent for printing.

Example:

```
$pr file | lp
  request-id is dmp-240(1 file)
$
```

23.wc Command (Line, word and Character Counting):

- The wc command is used to count the number of lines, words and characters that a file contains.

Syntax: `wc [-clw] [file .....]`

Option	Description
-c	Displays number of characters.
-l	Displays number of lines.
-w	Displays number of words.

Example:

```
$wc poem
12 40 200 poem
```

lines words characters

The output indicates that the file poem has 12 lines, 40 words and 200 characters.

When wc is used with multiple filenames, it produce a line for each file as well as total count.

```
$ wc chap1 chap2 chap3
305    4058        23179      chap1
550    4732        28132      chap2
377    4500        25221      chap3
1232   13290       76532      total
$_
```

24.cut Utility:

- The cut utility copies the specified columns to the standard output file. It is used to cut parts of a file.
- It takes filenames as command line arguments or input from the standard input. It does not delete the selected parts of a file.

Syntax: `cut <option> <character or field list> <file(s)>`

option	Function
-f	Displays the fields specified
-c	Cutting of columns i.e. character by character
-d	Specifies the column delimiter

Example:

```
$ cat shortlist
2233 | shulka a.k.   | g.m.       | sales       | 6000
9876 | Sharma        | director   | production  | 7000
5678 | charkrobarty  | d.g.m.,    | marketing   | 6000
2365 | sengupta      | director   | personnal   | 7800
5423 | gupta n.k.    | chairman   | admin       | 5400
$_
```

- Cut can be used to extract specific columns from the file ex. Name (second field) and designation (third field). The name starts from column number 6 and goes

up to column no. 22, while designation data occupies columns 24 through 32. use -c (column) option for cutting columns:

```
$ cut -c 6-22, 24-32 shortlist
2233    shulka a.k.
9876    Sharma
5678    charkrobarty
2365    sengupta
5423    gupta n.k.
$_
```

column numbers must immediately follow the option.

```
$ cut -f 1, 5 shortlist
2233  | 6000
9876  | 7000
5678  | 6000
2365  | 7800
5423  | 5400
```

25.paste Command:

- In contrast to cut, the paste command takes tables and combines them side by side to form a single wide table. Paste command is used to paste several files together.

Syntax: `paste file1 file2.....`

- By default, paste assumes a tab as the delimiter.

Example:

```
$cat ROLL     $cat MARKS
    101           67
    102           80
    103           45
    104           50
$             $

$paste ROLL MARKS
    101           67
    102           80
    103           45
    104           50
$
```

- Paste with –d option can be used to specify a number of delimiters to separate the fields.

Example:

```
$paste -d: ROLL MARKS
        101 : 67
        102 : 80
        103 : 35
        104 : 50
$
$paste -s ROLL MARKS
101     102     103     104
67      80      35      50
$_
```

26. sort Command: Ordering a File

- Many times you want to arrange contents of a file in specific order.
- Whether it is comparing files in alphabetic order or setting up database in numeric order.

Sort option	Description
-k n	Sort on n th field, Ex. sort –t "|" –k 2 xyz (sort on 2nd field)
-t char	Uses delimiter char to identify fields
-u	Removes repeated lines
-n	Sort numerically
-r	Sort in reverse order
-c	Checks if file is sorted
-o filename	Places output in file filename
-m list	Merges sorted files in list
-k m.n	Starts on n th column of m th field
-k m,n	Starts sort on m th field and end sort on n th field
-f	Case insensitive sort

Example:

```
$ cat numfile
4
2
27
```

```
10
$_
$ sort -n numfile
2
4
10
27
$_
```

Duplicate lines are removed with the –u (unique) option.

27. tr Command: Translate characters:

- The tr command copies standard input to standard output with substitution or deletion of characters.
- Input characters found in strings are mapped into the corresponding characters found in string 2.

Syntax: `tr <option> <expressions1> <expression2> <standard input>`

```
tr [-cs] string1 string2
tr [-s] [-c] string1
tr -d [-c] string1
tr -ds [-c] string1 string2
```

–c complements the string of characters in string1 with respect to the universe of characters whose ASCII codes are 001 through 377 octal.

–d deletes all input characters in string1

–s squeezes all occurrences of repeated characters that are in string1 to single characters.

Examples:

(i) `$ tr "[a-z]" "[A-Z]" < xyz` or `( 'a-z' 'A-Z')`

`$ tr "[:lower:]" "[:upper:]" < xyz`

Used to fold all lowercase characters to uppercase in the file xyz

(ii) `$ tr "[a-d]" "[0-3]" < xyz`

First four lowercase alphabets are substituted with numbers from 0-3

(iii) `$ tr -s " " < xyz`

Squeezes all occurrences of blank spaces to one space in the file xyz

(iv) `$ tr -d "e" < xyz`

Deletes all occurrences of e from file xyz

(v) `$ tr -cd "e" < xyz`

All characters other than e would be deleted from the file xyz

(vi) `$ tr -c "[a-z]" "[\n]" < file1`

Outputs a list of all words in file1, one per line in file2