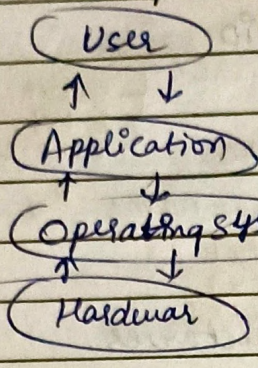
Page No.
Date:

Unit-1

What is an operating System?

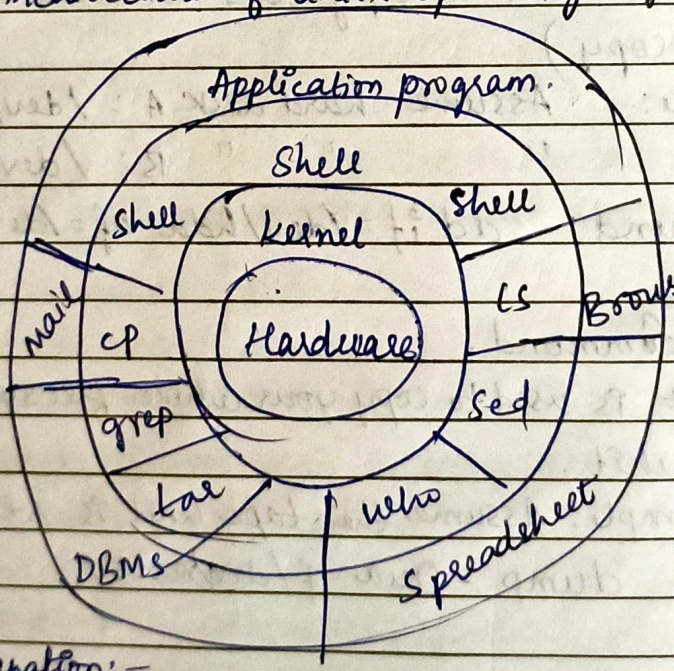


- OS is a manager, it manages all the available resources on a computer, from CPU, to memory, to hard disk.

What does OS do?
- control hardware
- Run Applications
- Manage Data & files.

Architecture of UNIX

Architecture of unix operating system.



Explanation:-

(i) Hardware: CPU, memory, I/O devices.

(ii) Shell:- bridge btw user & kernel; translates the commands entered into instructions for the kernel; Ex: sh, bash, csh

(iii) Application programs (outer layer): Ex: mail, DBMS, Browser, Spreadsheet.
- Built using shell commands & utilities.

Intro to UNIX

File Systems:-

Avoid characters: < or >

Spot seperators (word seperators): _ , - , etc.

Don't use . to start filename, it means hidden files in UNIX.

Wildcards:-

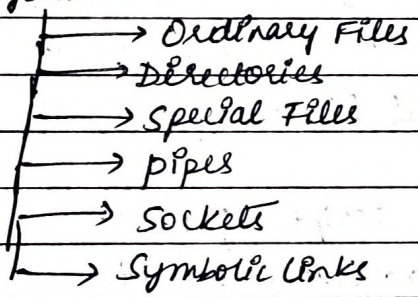
* : every file

f* : every file that starts with f.

*f : all fills that ends in f.

Classification of file types.

Classification of Unix File system



Unix commands:-

(9) • cal : used to see calender of a specific month or the whole year.

By default shows current month with current date highlighted.

cal -y → calender of whole year with today's day highlighted.

cal 2018 → calender of year 2018.

(PP) date command:-

↳: $ date

output: Friday 15 December 2023 01:06:49 PM IST

Variations:- 1) date -d "tomorrow"

output: Sunday 17 July 2023 01:06:49

2) date -d "1 year"

→ // Date and time after 1 year from present.

Page No.
Date:

Some imp commands:

mv → moves files and directories | eg: mv file1.txt file2.txt
cp → Copies files.txt to file2.txt | eg. cp file1.txt file2.txt
rm → removes files or directory | eg. rm old-file.txt.

Side 22

Some imp options:-

-l → Long listing format of files and directories, one per line.
-r → List of all files and directories in reverse order.
-t → ~~started~~ Sorted by modified time, started with the newest file.

→ Hidden files:- There are files beginning with a period (.). They are usually used by ~~linux~~ system to store system configuration info.

Note:- To see all hidden files use "ls -a" command.

*Command Script:-

Eg: script file.txt // If file not provided then
① script geeksforgeeks.txt // default file will be cre
Script started, file is geeksforgeeks.txt
geeks.txt.

*Command Uname:-

Syntax: $ Uname - option

options:-	meaning
-a	displays all info.
-s	Kernel name
-v	Kernel version

Examples:-

① $ Uname -v
Output: version details of the ~~sy~~ Kernel
② Uname -a
Output: all details about system

* command pwd:-

Syntax: $ pwd (print working directory)
Output: /home/shreyas/documents

root directory (indicated by a slash)
present working directory

Page No. 7
Date:

* who command: → It provides the name of user logged in, terminal line numbers, time of login.

Example: $ who
Output: Shreyas tty1 2025-03-10 09:30
arun pts/0 2025-05-10 10:05

Variations: i) $ who -u // shows idle time + PID + above output

ii) $ who -H // displays column headings

Output: NAME LINE TIME
Shreyas tty7 2025-03-10 09:30

iii) $ who -q // concise list of users + count.

output: shreyas arun
#users = 2

iv) $ who -a // display all information.

* tar command:

→ used to archieve files or directories:

Syntax:- tar -cvf files.tar file1.txt file2.txt

Meaning:
- c: create archieve
- v: show files being archieved
- f: archieve file name provided

output:
file1.txt
file2.txt.

* using of gz. (gzip extention to compress the archieved files - compression means to remove the similar data).

tar -czvf files.tar.gz myfolder

new folder name that stored zipped files.

folder to be zipped

-z option tells the tar to compress or decompress the archieved files.

Output.
system boot 2025-03-10 08:45
LOGIN tty1 2025-03-10 09:00
Shreyas tty1 2025-03-10 09:30 2456
arun pts/0 2025-03-10 10:05 02:10 2789

Page No.
Date:

* File permissions :-

Every file has 3 kinds of permissions associated with it namely :- i) owner's permission ii) Group's permission iii) others permission.

Each of these permission include the following ones:

w → write

r → read

x → execute

Example $ ls -l

groups permission

Output: -rwxr-xr-- 1 amrood ----

owners permission | represent permissions | others permission.

Slide 44

* Command ls -l :-

Example :- $ ls -l

output : -rw-r--r-- 1 Shreyas users 2048 Ma

Meaning :- 1 → number of links

Shreyas → owners name

users → group owner

2048 → size of file in bytes

Mar 10 → last modified ~~time~~ date.

file.txt → file name

* Changing Permissions

group others

Method 1 chmod u-x, g=rx, o+rwx file.txt

owner

Method 2 :- --- = 0

--x = 1

r-- = 4

-w- = 2

* Changing owner & group: username ← filename
cmd $ chown user filelist
cmd $ chgrp group filelist
groupname

Example:-

cmd: $ chown arun file1.txt
output: No output if successful
checking change: ls -l file1.txt
output: -rw-r--r-- 1 arun developers 2048 Mar 10 file1.txt
cmd: $ chgrp Innovators file1.txt
output: (No output if successful)
checking change: In place of "developers" we have "Innovators"

* VI editor.

→ Modes

-rwxrwxrwx shreyas user 2025-03-10 2048 Mar10 file.txt

a10 file.txt.

- Command mode: This mode enables you to perform administrative tasks such as saving files, executing commands, moving the cursor cutting and pasting lines or words, and finding and replacing. In this mode, whatever you type is interpreted as a command.
- Insert mode: This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and finally it is put in the file.

* There are different methods to enter into the insert mode:

i	Inserts text before current cursor location.
I	Inserts text at beginning of current line
a	Inserts text after current cursor location.
A	
o	
O	

Unit 2

Page No.
Date:

* login commands:- (back to login thing)
  $echo $SHELL --- Login shell
  $echo $0 --- prints current shell.
* Switch to different shell by commands:-
  - bash
  - ksh
  - csh
* Each process has unique PID. (back to foreground & background process)

Commands:-
- $ps → lists all the processes that are running. (chapter PT - ps page)
- control+c → exit command.
- For processes running in background pid if known then → $ kill 19
  Terminated (bg page)
- df - shows amount of available disk space being used by file systems.

free - Shows total amount of free & used physical & swap memory in the system, as well as the buffers used by the kernel.

Types of processes:-
- parent & child process: most of the processes have shell as their parent process.
- orphan process: No parent, child (parent terminates first before child)
- Zombie process: process terminated but still appears on the terminal when ts ps command is given.
- Deamon process: system related, often run in background

example commands. outputs. file systems?

Page No.
Date:

Disk utilities: These are commands that allow users to know about to storage utilization by the files and directories.
- How much space is used?
- How much is left?
- Which files take how much space?

Command: ① df command.
→ df -h
- used to display the size occupied, utilized, allocated remaining for a filesystem.

Ex: df -h

Output:-

Filesystem	Size	Used	Avail	Use%	Mon
/dev/sda1	50G	20G	30G	40%	
tmpfs	2.0G	0	2.0G	0%	

② du command
→ It shows the disk used space by a file specified.
Example:
① du -h myfolder
Output
4.0K myfolder/file1.txt
8.0K myfolder/subfolder
12K myfolder
(ii) du -ah // shows all files & directories in human // readable form.
// Helps find large files
(iii) du --time // displays space occupied and // displays the time when the file was last modified

Network utilities:-
① ping:-
- basic network testing tool used to check weather a remote host is reachable
- It helps diagnose network issues such as unreachability

hosts, slow responses, or intermittent connectivity problems.
- Commonly used as the first step in network troubleshooting.

(ii) Finger:-
- It shows details about user on a system. Details include : login name, full name, login time, idle time and directory
- used to manage system auditing & user activity.
- used by the host to monitor the users.

(iii) Traceroute :-
- shows exact path the packets take from your machine to a destination host.
- Lists each intermediate router along with the time taken to reach each router
- Helps identify where congestions, delays, or failures occurs in/user/1000 in network paths.
- Helps track network issues and slow network segments

(iv) Netstat

- Host
- Traceroute
- Netstat
- Traceroute
- Hostname
- Route
- Nslookup

Test processing utilities:-

Page No.
Date:

① Sort - Sorts lines of text files

cmd: sort file.txt :- sorts file.txt's lines alphabetically.

cmd: sort -n file.txt : sorts the file numerically.

cmd: sort - r file.txt : Sorts the file in reverse order. Thus b comes before a.

cmd: sort -u file.txt : Sorts the file, removing duplicates lines, thereby ensuring each output line is unique.

Examples:-

1. Alphabetical sorting:-

File is names.txt

Ravi
Anu
Kiran
Bala

cmd: sort names.txt

output:

Anu
Bala
Kiran
Ravi

2. sort -r names.txt
// reverses the sorted file

3. File: numbers.txt

10
1
50
2

cmd: sort -n numbers.txt

output:

1
2
10
50

4. Removing duplicates:-

File fruits.txt:

apple
banana
apple
cherry
banana

cmd: Sort -u ~~data~~ fruits.txt

output: apple
banana
cherry.

(ii) uniq.

This command is more specific to recognizing duplicates. Usually requires a sorted input as the comparison is made on adjacent lines only.

options: -d : print only duplicate lines
-c : prefixes count of occurance which out put display
-u : print only unique lines.

Examples:-

1. File: fruits.txt
apple
apple
banana
banana
cherry
apple

cmd: uniq fruits.txt

output: apple
banana
cherry
apple

3. Using same fruits.txt file
cmd: uniq -d fruits.txt
output: apple
banana // only adjacent // duplicates

2. Using same fruits.txt file
uniq -c fruits.txt
output:
2 apple
2 banana
1 cherry
1 apple

4. using same fruits.txt file.
cmd: uniq -u fruits.txt
output: cherry
apple

(iii) Comm :- Outputs lines common to both files.
- Without any option it displays three columns :- 1. Lines unique to file1 2. Lines unique to file2 3. and lines common to both files.

options -1 : suppress lines unique to file1
-2 : suppress lines unique to file2
-3 : Suppress lines unique to both files.

(iv) Cmp :- used to compare two files byte by byte. Usually used to compare two binary files
- If files are identical then exit status 0 returns nothing

- If not-identical, exit status 1, returns line number and byte location of first difference.

(v) Diff: compares files line by line

@ slide no. 31 options.
- If same prints "file1.txt and file2.txt are identical"
- else prints "file1.txt and file2.txt differ"

i) -s: conveys message when both files are same.
ii) -y: two column output
iii) -i: ignore case while comparing.
iv) -w: ignore white-spaces while comparing.

output:

Apple	Apple
Banana	Banana
Mango	| orange

- Matching lines appear side by side '|' indic

Back up Utilities

(1) Tar. (notes are written about this in detail in unit I)

(2) Cpio: cpio (copy input and output) is a UNIX backup and archiving utility used to copy files between directories or into archive files.

Common modes (options):-
- -o → create an archive
- -i → extract files from an archive
- -p → copy files from one directory to another (pass-through)
- -v → verbose (shows files being processed).

Ex1:

```
ls *.txt | cpio -ov > backup.cpio
```

output:

```
file1.txt
notes.txt
report.txt
3 blocks
```

} appear because of v

Page No.
Date:

(2) Restoring files:-

cpio -iv < backup.cpio

file1.txt
notes.txt
results.txt
3 block

Assume src/
- a.txt
- b.txt
- test.c

cmd: cd src

find . -name "*.txt" | cpio -pvd ../backup

output: ./a.txt
./b.txt
2 blocks

(3) Copying files between directories.

find . -name "*.c" | cpio -pvd /back

now backup directory contains up.

backup/
- a.txt
- b.txt

(3) Dd command.

- It is used to copy hard drives exactly (deep/low-level copy)

Example:- Assume hard disk A : /dev/hda
" " B : /dev/hdb

Command: dd if=/dev/hda of=/dev/hdb

ates diff
ereme is
these.

(4) dump command.

- It is used to copy your whole file system to a tape unit.

Example: Assume the tape unit is /dev/nst0

cmd: dump -0ua -f /dev/nst0/

diff

1c1
< apple

> Apple

3c3
< banarana

> orange

Unix

How to enable the web search in notebook lm.

Unit-3

$ls -l file1 nofile | more

Page No.
Date:

- Input redirection: To force a command to read from a file instead of the keyboard, use the < operator.
  like: command 0< file1 or simply command < file1

- output redirection: Similarly you can redirect the output of a ~~file~~ command to a file using this:-
  command 1 > file1 or command > file1
  Output operators:
  - '>' (overwrite) → If file exists it overwrites it, if doesn't exists it creates new one and write in it.
  - '>>' (appends) → Adds output to the end of the existing file without deleting previous content.
  - '>|' (force overwrite) → Overrides the file even if safeguards (like noclobber) are no.

  * noclobber → A setting that prevents accidental overwriting of files. If on, > will throw an error if the file exists.

- ~~Error~~ Redirection:-
  There are mainly two types of output stream in Linux:
  i) Standard output ii) Standard error.
  command > file1 works only for standard output but Standart error output stream is still console so. use:-
  Example for error direction:- ls nonexistentfolder 2>error.txt

  Example for error + output redirection:-
  ls folder nonexistingfolder > output.txt 2>&1
  (folder: existing; nonexistingfolder: not existing; 2>&1: tell that "same as 1 which means same as output")

* Pipes Intro:-
  - pipe operator:- connects the stdout of one command directly to stdin of other command.
  - pipes are unidirectional → the data flow is always from

left to right.

Key example: sort record.txt | uniq

Page No.
Date:

Demo:- Suppose name.txt contains:

arun
Kumar
ravi
arun
ravi
arun
Sachin

sort names.txt :-

arun
arun
arun
Kumar
ravi
ravi
Sachin

Note:- uniq only removes adjacent duplicates not all duplicates.

Now:- sort names.txt | uniq results in

~~names.txt:-~~ output:-

arun
Kumar
ravi
Sachin

* Tee ~~com~~ Command:-

→ It acts as a T-junction for data. Originally it takes in the input from stdin then displays it on the screen & simultaneously ~~copper~~ write it to a file.

Use -a option with tee to append it to file instead of over-writing.

Example of appending: wc -c file1.txt | tee -a file2.txt

(appends output to file2.txt).

* Command Execution types:

- Sequenced (seperated by ;) → ex: cmd1 ; cmd2 ; cmd3

- → no matter weather ~~cmd~~ commands fail or succeed they simply execute one after the other.

→ All commands run independantly, and any command that fails will print its own error message on the terminal.

* Redirection

→ By default, the standard Input stream is keyboard.
by default, the standard output stream is screen.
by default, the standard error stream is also screen.

- We can change ~~this~~ these streams by a method called redirection. (1 is called file descriptor / filename)
  command 1 > file.txt. (1 for redirecting output)
- Even errors can be redirected to files.
- When the given file exists, 2 cases arise:-
  case 1: If file is ~~emp~~ empty → then no problem it writes into the file
  case 2: If noclobber is on then it prevents appending / overwriting the file else the file gets overwritten.
  Three symbols:- '>', '>>', '>|' (look at previous notes)

* Error Redirection:-

- We can redirect an error using this syntax:-
  command 2> filename
  (2 this options tells error should be redirected)

* Pipes

- pipes act as a temporary buffer where the output of a command is stored and later provided as an input for the other command.
- pipes act as a mode of redirection where the output of one command is redirected to ~~other~~ input of another command.
  Ex: $ who | lpr
  // This prints a list of logged in users.

→ Examples of variations of error redirection:-
(1) cmd:- ls wrongfile 2> error.txt // Error method is saved in error.txt and nothing is shown on the screen
//n

(ii) Redirect errors and output seperately:-

cmd: ls file1 wrongfile > output.txt 2> error.txt

Note by default '>' means '1>'

// Normal output → output.txt

// error messages → error.txt

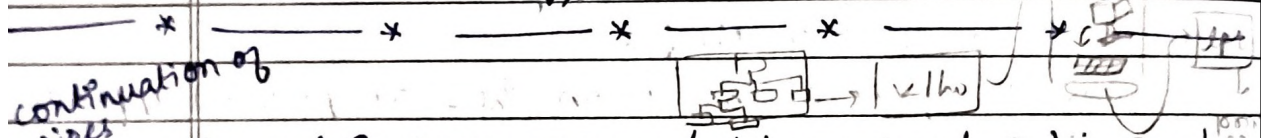
(iii) Redirect both output & errors to same file

cmd: ls file1 wrongfile > all.txt 2>&1

(iv) Appends error to a file:-

cmd: ls wrongfile 2 >> error.txt

continuation of pipes



General Syntax: command-1 | command-2 | ... | command-N.

Examples:- sort record.txt | uniq

// sorts the file and then displays only the unique

// elements of that file.

* tee command.

- The tee command is used to send the output of a command to the screen as well as one or more file.
- It simultaneously write to both, copying the result into files and also displaying the output.
- There is a -a option.

Ex usage: wc -c file1.txt | tee file2.txt

(1 command)

output: 15 file1.txt

$ cat file2.txt (used to view the file contents)

15 file1.txt

slide 20

continuation of Unit-3

Page No.
Date:

Command Execution:-

• sequenced commands: Here we seperate the commands by a semicolon. The commands are given in sigle line. Irrespective of the success of failure of preceding commands they get executed one by one.

Ex: mkdir test; cd test; touch file.txt.

output:- (No output if all succeed).

Note:- touch → if file not exits, creates new file, if exists update the timestamp

• Grouped commands:

Ex: { ls; pwd; date; } > output.txt.

Contents of output.txt:-

file1.txt
file2.txt
/home/shreyas
Mon Mar 10 11:30:45 IST 2025

- Multiple commands are grounded together and executed simultaneously as one single unit.
- often associated with redirection.

• ~~Chained~~ conditional commands:

Ex:- Case 1: Using AND.

Ex: cd project && echo "Opened"

Output:- If ~~ed~~ cd command is successful only then Opened is printed else it is skipped.

Case 2: Using OR

Ex: cd test || echo "Directory Not found"

output:- only if cd command fails echo is executed else it is skipped.

* Key points:- - Each next command will use the exit status of the chained command to decide its execution.

- Each command will return 0 for success, non zero for failure

* Meta characters :- special characters which mean something to the editor.

Page No.
Date:

To print them we need to stop editor from interpreting them as special characters.

Meta characters — Quotes — single quote, double quote, backslash

To print them, 2 ways :-

1) either include them in single/double quote :-

```
$ echo " ' "
// prints '
$ echo " " "
// prints "
```

2) include backslash before them :-

```
$ echo \'
  // prints '
$ echo \"
  // prints "
$ echo \\
  // prints \
```

* Command substitution :- provides the capability to convert the result of a command to a string.
- This string can be assigned to any other string variable or printed.

Ex: echo "today is $(date)"

output is converted to string.

* Job control.

What is a job?

→ In UNIX a job is defined as a command or set of command entered on 1 command line.

The jobs are divided into two : • foreground • background.

→ The foreground jobs are the ones which are connected to the keyboard and monitor.

→ There can be many background jobs at the same time

* Some Important keys for foreground job:-
- A job starts as soon as you hit the return key.
- use ~~ctrl~~ ctrl + z to suspend the job.
- use fg command to resume it.
- use ctrl + c to kill the foreground job
-

* Some Important keys for background jobs:-
- To suspend a background job use stop command
- To restrat it we use the bg command.
- To terminate the background job we use kill command
- All three commands require the job preface number with a % symbol

Example:-

$ longjob.scr &

output: [1] 179541
↑ PID
preface number

$ stop %1

output: [1]+ 179541 stopped longjob.scr&

$ bg %1

output 1 longjob.scr&

$ kill %1

output: [1] + Terminated longjob.scr&.

* Moving jobs between foreground and background:-
- To move a job from foreground to background or vice versa the job must be suspended.
- To move to background:- once suspended use bg command, no number required as it is in foreground
- To bring back to foreground: $ fg %1
preface number.

Page No.
Date

* Multiple background jobs:-

→ jobs command:

- List out all the jobs happening with preface numbers, concurrency and state

Ex: $ jobs -l

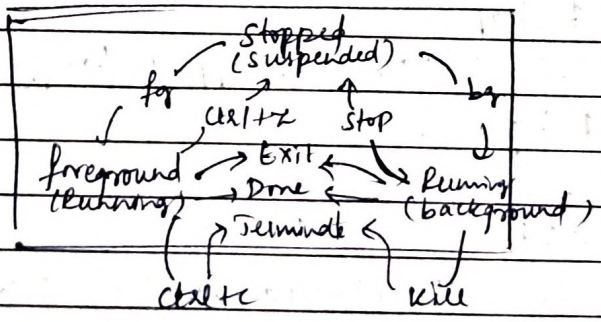
```
[1]   7895  Running   g pass &
[2]   7906  Running   gname-calculator &
[3]-  7910  Running   gedit fetch-stock-prices.pg &
[4]+  7946  Stopped   ping cyberciti.biz.
```

jobID (prefix number) | PID | state | jobs

'+' → Indicates current default job, if you enter suppose fg ~~bg~~ without preface number it will apply to this job.

'-' → Indicates the job that had '+' sign before the currently '+' sign job ran.

Figure for reference of full jobs concept



* Aliases:

These are names assigned to a command. We can use these names instead of the command.

Ex:- $ alias dir='ls -l' ↵

Note:- There is no space before and after '=' sign

* Unalias command:- ~~It is to~~

Ex:- $ unalias dir ↵ ls -l

// It unaliases the previously aliased ~~dir~~ command

Note:- Use 'unalise -a' to unalise all commands, including system defined aliases

→ Using alias:-

$ alias

output:-

alias l1='ls -l'

alias la='ls -a'

alias gs='git status'

The alias is different for C shell. This Table summarizes the points of aliases in a:-

Feature	Korn & Bash	C
Define	$ alias x=command	% alias x command
List	$ alias	% alias
Remove	$ unalias x y z	% unalias x y z
Remove all	$ unalias -a	% unalias *
Argument	Only at end	Anywhere.

* Variable:-

Variables

↓ user-defined ↓ pre-defined

no space eg: x="Hello" ✓

Action	Korn & Bash	C
Assignment	variable = value	set variable=value
Reference	$ Variable	$ variable

* Some important predefined variables:-

1) noglob :- Used to toggle matching wildcards (by default feature ON & OFF)

ex: set +o noglob

$ ls * txt

output:- (* will appear as * is matched)

Page No.
Date:

set -o noglob
$ ls *.txt
Output:-
ls: cannot access '*.txt': No such file or directory.

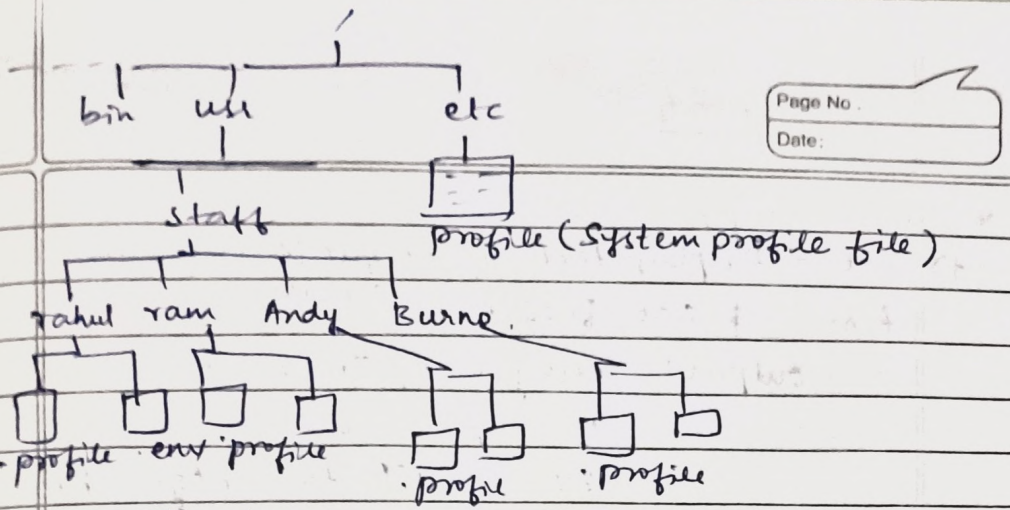
(* is treated literally)

* Customization:- Two types:- i) Temporary
ii) Permanent.

* -System ~~Profile~~ Environment File:-
- contains all the system variable that are global and apply across all the users
- The global variable include PATH, HOME, SHELL etc
- This file is read as soon as the system boots
- The changes in the file apply across all users and applications
- used to maintain consistent system-wide behaviour

* Personal Profile File:-
- Contains user-specific variable settings
- Read when the user logs into their user ID.
- It provides the user an opportunity to customize the variable settings.
- Examples:- ~/.profile (KornShell); ~/.bash_profile file

* System Profile File:-
- commonly located at /etc/profile.
- Same points as system Environment File



* Fig Korn's Environmental File.

* For Bash shell:-
- Personal profile file, one of the three files is used
  ~/.profile, ~/.bash_profile, ~/.bash_login
- for the system profile file we use:- /etc/profile file.
- There is also a logout file ~/.bash_logout.

* For C shell:-
~/.login is equalent to the ~~system~~ user profile file.
~/.cshrc is equivalent to the environmental file
~/.logout is equivalent to ~/.bash_logout.

tr command

Ex:- $ echo "hello" | tr 'a-z' 'A-Z'
output: HELLO
$ echo "hello123" | tr -d '0-9' (using -d option)
output: hello // deletes all digits.
$ echo "aaabbbccc" | tr -s 'a-c' (using -s option)
output:- abc // squeezes the characters from range 'a-c'

Redirection using files:-
→ tr 'a-z' 'A-Z' < data.txt // does the same
→ tr 'a-z' 'A-Z' < data.txt > output.txt // with contents of file & displays it
// puts the output into output.txt.

UNIT-4

Page No.
Date:

* Filters and pipes:-

1) cat - Displays text of the file line by line
Ex:- $ cat file1.txt ← path
output:- MergeSort, QuickSort
Binary Search
Hare and Tortoise Algorithm.

2) head - It displays the first 10 lines of the file by default. If mentioned explicitly with option prints that much no. of lines.
Ex:- $ head file1.txt
output:- first 10 lines.
Options $ head -6 file1.txt
output: first 6 lines.

3) tail - It works same as that of head but in a reverse order. It displays lines from bottom to up.
Suppose file1.txt has:- Shreyas
Saanvi
Biradar
Ex:- $ tail -2 file1.txt
output:- Saanvi
Biradar
//default it displays 10 lines.

4) Sort - By default sorts the contents of the file alphabetically. There are options to change it.
Ex:- $ sort file1.txt
output:- displays line by line that are sorted

5) uniq - Same as studied in unit-2.
Ex:- $ sort file1.txt | uniq
Output:- Only unique lines in the file1.txt.

Page No.
Date:

6) wc - prints no. of lines, no. of words, no. of characters in a file.
Ex:- $ wc file1.txt
output:- 4 24 128 file1.txt
(4 ↑ lines, 24 ↑ words, 128 ↑ characters, file1.txt ↑ path.)

7) grep - It is used to find a word in a file. The output has that word highlighted.
Ex:- $ grep Scooby scoobyDoo.txt
output: has Scooby word highlighed with red.
$ cat scoobyDoo.txt | grep Scooby
↑ works as the same.

8) tac - reverse of cat. prints last line through to 1st line

9) sed - It is used to replace a particular word in a file.
Ex:- $ sed 's/scooby/scrapy/g' scoobyDoo.txt
output:- Replace every occurance of 'scooby' with 'scrapy'

* Concatenating files (cat)
→ cat command can be used for 3 purpose:-
$ cat f1, f2
output:- // concatenates f1 and f2
$ cat f1
output: // Displays content of f1 line by line
$ cat > filename
// creates as file with filename.

Options - 6 options divided into 4 categories
1) visual characters: a) cat -v --- // prints control characters
b) cat -vet filename (-ve option)
// '$' sign printed at the end
// (-vt) option tabs appear as ^|

Page No.
Date:

ii) Buffered output
a) cat -u ---- written to the file immediately
iii) Missing files
a) cat -s ---- To avoid error message on output screen when file is missing
iv) Numbered lines
a) cat -n ---- numbered line output.

* Displaying beginning and end of the file.

1) head command - Refers previous note
$ head -n file1 file2
output :- gives filename before the displaying of lines.
Ex :- $ head -n 5 states.txt
output: Andra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhatisgarh
Ex :- $ head -c 6 states.txt
output: Andra // prints the first 6 bytes of the // file provided.
Ex :- $ head -v States.txt
output :- // data always preceded by its file name.

2) tail command - Displays by default last 10 lines of the file.
Option :- i) $ tail -n 5 file1.txt
output :- Last 5 lines
ii) $ tail -c 20 file1.txt
output :- Last 20 characters
iii) $ tail -v file1.txt file2.txt
output :- displays headers while displaying multiple files.

Page No.
Date:

Cut and Paste commands.

→ using -b option with cut command

cut command -> It is used to cut some lines, columns, bytes from a file and directly display it on the screen

Ex:- $ cut -b 1,2,3 state.txt (without Range)

output:- And
Aru
Ass

$ cut -b 1-3,5-7 state.txt (with range)

output:- Andra
A ruach
Assm

$ cut -b 1- state.txt

output:- Andhra Pradesh // cuts from
Arunachal Pradesh // 1st byte to
Assam // end of each
// line.

$ cut -b -3 state.txt

output:- And // cuts from 1st
Aru // bit to 3rd bit
Ass

→ using -c option:-

Ex:- $ cut -c 2,5,7 state.txt

output:- nr // cuts 2nd, 5th,
rah // 7th columnan
sm // d displays

$ cut -c 1-7 state.txt

output:- Andhra
Arunach
Assam

→ using -d option (delimiter)

$ cut -d " " -f 1 State.txt

→ using -f option

Assume students.txt has: 101: Raul: 85 (delimiter = ':')
102: Arun: 92
103: Kiran: 78

cmd: $ cut -d ':' -f 1 students.txt

output: 101
102
103

Similarly use -f 1, 3 states.txt to print first and 3rd column.

$ cut --version // displays the version of // cut currently being used // on your system

Paste command:

paste command -> The 'paste command is used to merge two columns of different files and display them on output.

→ By default it is parallel merging

Ex:- $ paste number state capital
(file1 file2 file3)

output:-
1	AndraPradesh	Hydrabad
2	Arunachal Pradesh	Itanagar
3	Assam	Dispur

// By default the columns are seperated by tab.

→ using -d option (delimitor).

$ paste -d "|" number state capital

output:- 1|Arunachal Pradesh|Itanagar
2|Andhra Pradesh|Hydrabad
3|Assam|Dispur.

→ $ paste -d "|," number state capital

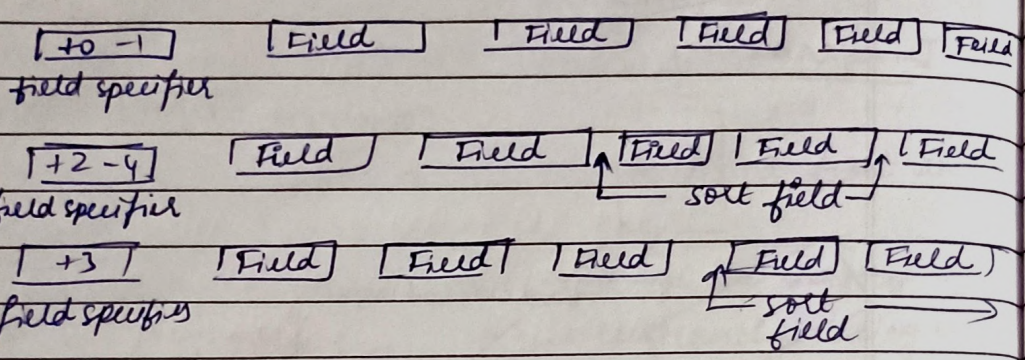
output:- 1|Arunachal Pradesh|Itanagar
2|Andhra Pradesh,Hydrabad
3|Assam,Dispur

// It uses them (multiple delimitors) in a circular // fashion

Page No.
Date:

Sorting

* Sorting using fields:-



Example:- Assume a file called marks.txt

Output:

101 Ravi 85
102 Arun 92
103 Kiran 78
104 Sita 88

cmd: $ sort +0 -1 marks.txt

output: 101 " "
102 " "
103 " "
104 " "

cmd: $ sort +2 -3 marks.txt

// sorts with respect to marks (85, 92, 78, 88).

* Check sort Sequence (-c): verifies that the file is sorted or not, if not sorted, the first out of sequence line is displayed.

cmd: $ sort -c +0 -1

* Using delimiter: Sort by default uses space/tab as seperator for fields but we can use other delimiter as well.

cmd: $ sort -t ":" students.txt

$ sort -t ":" +0 -1 marks.txt // sort based on
// first field.

Example file
101: Ravi: 85
102: Arun: 92
103: Kiran: 88
104: Sita: 78

(Delimiter = ':')

* Numeric Sort Using -n option.
  cmd: $ sort +1 -2n data.txt // skips the first field in // the sort and perform // numeric sort on second // field.

* Merge files (-m): combines multiple ordered files into one file that is ordered.
  cmd: $ sort -m file1 file2

* Example commands:-
  cmd: $ sort -t':' -u +1 -2 filename // It sorts // the file using delimiter ':' and w.r.t 2nd field
  cmd: $ sort -t':' +1 -2 +2n -3 filename // This is // multiple pass sort which means first sorts based on second field // denoted by "+1 -2" and then sorts numerically based on 3rd // field denoted by "+2 -3".

Translate command (tr command)
tr (translate) is used to replace, delete, or squeeze characters from input.

1) Convert lowercase to uppercase:
cmd: echo "hello" | tr 'a-z' 'A-Z' //output: HELLO
2) cmd: echo "abc123" | tr -d '0-9' //output: abc
3) cmd: echo "aabbbccc" | tr -s 'a-c' //output: abc
4) cmd: tr 'a-z' 'A-Z' < data.txt // converts the content of // file data.txt to upper case

Word count command
- wc -c filename // counts number of characters
- wc -w filename // counts the number of words
- wc -l filename // counts the number of lines

* computation and utility
* ~~Ext~~

Page No.
Date:

* Diff command.

~~It reports~~, It compares the two files, and reports only the lines that mismatch

Example:-

file1.txt: Apple
Mango
Banana

file2.txt: Apple
Grapes
Orange

cmd: diff file1.txt file2.txt

Output:
2c2
< Mango

> Grapes
3c3
< Banana

> Orange

Explanation:
- 2c2 means the 2nd word in file 1 does not match with 2nd word in file 2.
- "<" symbol indicates that it is file 1 ka word
- ">" symbol indicates that it is file 2 ka word.

* grep command:

① cmd: grep "error" log.txt
output: prints the containing letter sequence "error", even prints a word like erroneous as it has "error".

② cmd: grep -i "Unix" file.txt
consider file.txt:- Unix is powerful
I like UNIX
Learning unix shell
Universe is big.
output: Unix is powerful
I like UNIX
Learning unix shell

③ cmd: grep -w "unix" data.txt
output: learning unix shell.
// matches exact word unix only- does not // display unixshell

③ cmd: grep -c "success" log.txt
output: 3 // Assume that the log.txt contains 3 times word "success".

Page No.
Date:

④ cmd: grep -n "unix" data.txt
output: 1: Unix is powerful
3: Learning unix shell.

* Network utilities:-

(i) ping:
function: checks if a remote system is reachable and also provides time taken by package.
Cmd: ping google.com
Output: 64 bytes from google.com: time = 25 ms
64 bytes from google.com: time = 27 ms

(ii) host:
function: finds the IP address of a domain name
cmd: host google.com.
Output: google.com has address 142.250.183.14.

(iii) traepath
Shows the path taken by data packets to reach a destination.
cmd: traepath google.com
Output: 1: 192.168.1.1 ← your local router the first device your packet reaches
2: 10.10.1.1 ← your ISP's router (next network on the Internet)
3: google.com ← The final destination you wanted to reach.

(iv) netstate
Purpose: Displays network connections and open ports.
cmd: netstat -an
Output: tcp 0 0 127.0.0.1:80 LISTEN.

Page No.
Date:

(v) route:

Shows the routing table of the ~~table~~ system

cmd: route

Output: Destination Gateway Genmask

0.0.0.0 192.168.11 0.0.0.0

continuation of diff command:-

* cmd: diff -u file1.txt file2.txt

⇒ The output shows how file1 must change to become file2.

Assume file1.txt: Apple file2.txt: apple

Mango grapes

Banana orange

Output:

- Apple
- Mango
- Banana
+ apple
+ grapes
+ orange

cmd: diff -c file1.txt file2.txt

Output:

*** file1.txt

--- file2.txt

****

*** 1,3 ****

! Apple

! Mango

! Banana

--- 1,3 ----

! apple

! grapes

! orange