* Operating System.
- interface between user & hardware.
- ∴ end user deals with Program / application.

Program / Application layer.
↓ Talks to OS
OS.
↓ does execution of program using hardware
hardware. --> (cpu, ram, disk & i/o devices)

So managing that hardware for exection is os do

- core of operating system is refer to as kernel

* OS functionility.
① Process management
② file & i/o management
③ memory management.
④ cpu sheduling.
⑤ hardware abstraction.
additional functionility
⑥ networking
⑦ security.

all These steps ① To ⑤ each OS does.
That is core os & That is kernel.

• linux Made ~~with~~ from Unix.

Unix slogen → Process have lives & files have spaces.

① every Thing is file in unix (Symbol)
* data file (-) * char device (c) = Transfer data Character wise (keyboard, mouse)
* directory (d)
* Pipe (p) * block device. (b)
* socket (s) (eg → pen drive, SSD, harddisk
* links (l) sector by sector data store.

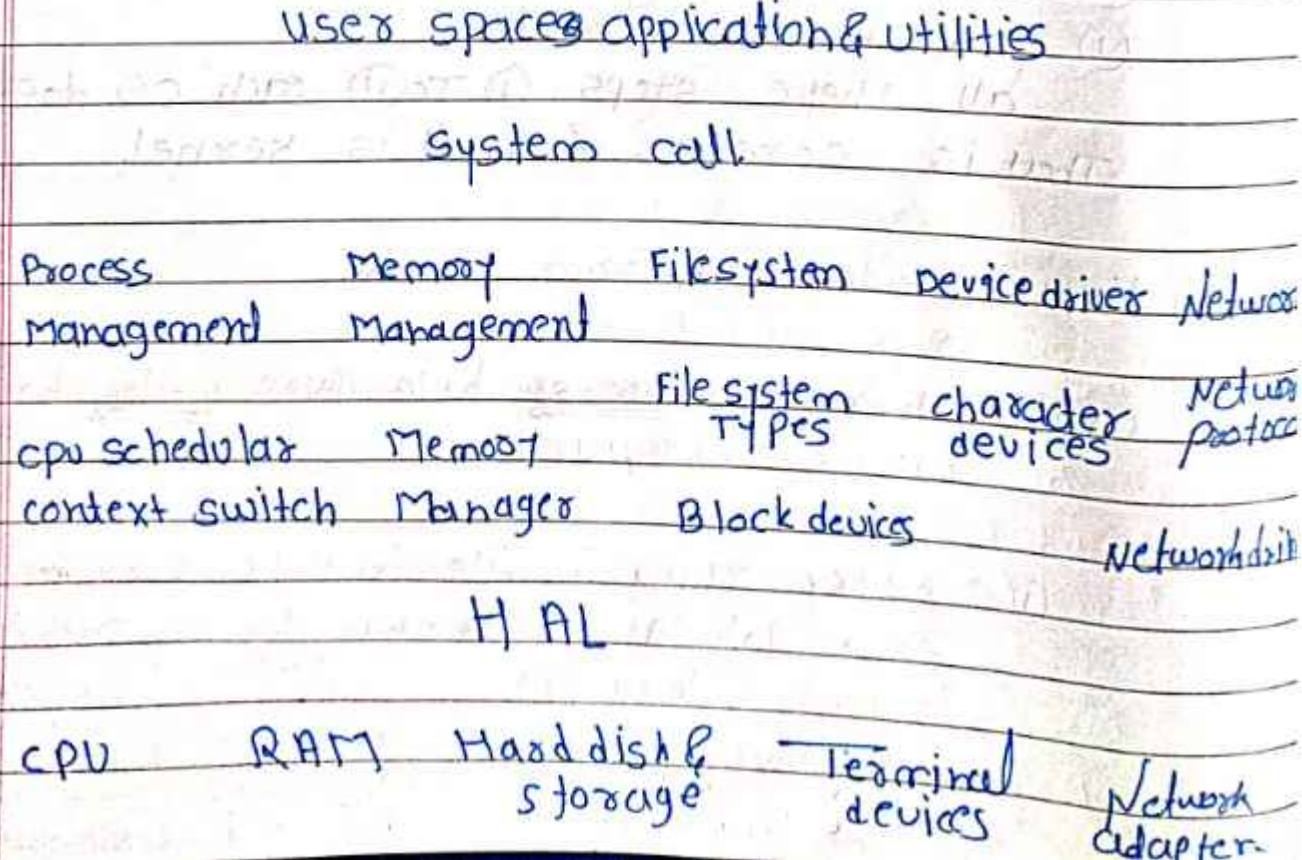
in block devices data store in sector.
1 sector = 512 bytes.

in unix everything you see is file & anything which is run it's process.

HAL → Hardware abstraction layer. (Hardware control)
eg if LG Harddisk & sony harddisk both different but internally manifacture wise but internally performing task is same / hat is called Hardware abstraction.

System calls = are the function expose by the kernel so that user process access the kernal functionality.

* Linux kernel architecture



File system type → windows have = NTFS, FAT etc.
Linux have = Ext3 Ext4 etc

* all our application running on user space & our application access any of the kernel functionality via system call

* System call internally executed via something called software interpet & that will help you to switch user mode & kernel mode.

① How To learn operating system.
  ① you should become end user
    - commands
  ⑪ after end user you become administrator
    - Shell scripts.
  ⑪ becomes a Programmer
    - handling system call
  ⑪ designer
    - as internal.

* linux shell
Shell is a programm that inputs command from end user & get them executed from the kernel

⑧ shell Types
① CLI shell
  - for eg without cmd.exe.
① GUI shell
  - for windows eg → explorer.exe
    it's a GUI part ← like in desktop what you see all these.

Note → window have command exe
& one more key: launch Power Shell

but for wind Linux.

CLI shell → bash → bourne again shell

GUI shell → linux : Gnome or KDE GUI

Linux have variety of shells
oldest shell = bsh → later rewrite bash.
csh / tcsh → ten x csh (for c langu)
Zsh → Z shell
ksh → korn shell (used in unix) mostly.

You have to learn bash shell

Shell present how much command → ls /bin/*sh

* Linux file system hierarchy

/ root or bgining of file system.

boot → here kernel & bootloader (grub)
bin → executable / command
sbin → it's a system command need admin priv
lib → libraries extension → (.so) shared object and device driver (.ko) kernal object
usr → installed Program/software.
etc → contain system configuration (app", system, hardware configuration)
dev → device file (character device file &

MCQ → kernel name of linux is Vmlinuz
MCQ extension of library in window is .dll
device driver .cis

windows Program file contain install software.

·Proc → Process (use to monitoring configuration of hardware or software.

·sys → device management
·tmp → Temporary file system (automatically lost data while shutdown)

mnt → mount point to see outher file system.
mounting → external drive or c drive or pendrive content look here is called content mounting.

root → directory for admin user

home
·user1
·user2

eg if username is sunbeam it's home directory is /home/sunbeam.

in linux "start with forward slash " / "
They all are absolute path.

special directories.
· . : current directory
· .. : Parent directory
· ~ : home directory. (your home)

* rmdir delete only empty directory.
* To delete use rm -r dirpath
  -r stand for recursive

* file system command

- ls dirpath } show directory content.
  ls -l dirpath }

- mkdir dirpath & rmdir dirpath
  make directory remove empty directory

- cat filepath } - see file content.
  cat > filepath } - write into file.

- cp filepath dirPath } copy file into given directory.
- cp -r srcdirpath destdirPath
  → copy directory into given destination director
  (copy whole directory)

- mv filepath dirpath → move file into given dir
- mv oldfile newfile → rename file.
  in linux, file/directory starting with "." is hidden.

  if you want To see Then use ls -a

q. How To see The calender.
q. How To read manual

eg terminal > ls -l -a
output is shown in terminal

Terminal > ls -l -a > out.txt
output is copied in file.

terminal > ls
terminal cat out.txt

Terminal > sort → it will sort
nitish
sandeep
amit.
ctrl + D:
input is taken from terminal.

q if you want to sort anything in file

→ eg sort < filename.txt
input is taken from file
input output is shown in terminal.

q take input from another file & output in another file
→ Terminal > sort <filename > outputfilename.
cinputfile > Result file

q reverse order.
Teaming > sort -r < truth's.txt

q Terminal > Sort -z > err.txt
err.txt created but empty.
error output (stderr) show on Terminal.

wc = word count.

output redirection

> = input.

< = output

what is number of

stdin → 0
stdout → 1
stderr → 2

in previous question we send error msg to file it not go.

So if we

* Terminal > sort -x 2> err.txt

error output (stderr) is writeen into file where it go.
terminal > cat err.txt.

* word count
* terminal > wc
# hello sunbeam dac students.
our exam on 5 Jan.
it show number of line number of word
number of charaders.

2 9 47

*. terminal > ls -l
Terminal > ls -l | wc
left command (ls -l) output is given as input to right command (wc).

"|" is pipe

Punch card => store data. in os befor ~~many days~~. OS. it's in binary form.
& light is passed Through it if light pass it's 1 or not is 0.
= (light = electricity).

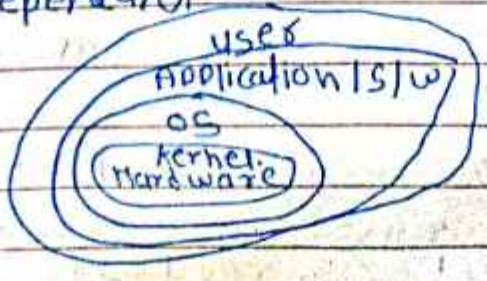
classmate Date 27-08/24 Page

Imp ① gain confidance in relative Path & ~~or~~ absolute Path.

* Day 1 Notes
- Operating System
- it's resource /Process Manager.
- it's Software devices (System Software)
- Take input from user & complete it from hardware. & gives output.
- before OS How it work.
  example of it → Java HelloJava you write.
  but you write JVM also for it To Process.
  & all ~~is~~ it happen with Punching card.
- using OS Throughput /effience is improve.
- OS is hardwared dependent.
  - os is software & software is need To install somewhere this is Hardware.
  - it require physical memory To execute.
  - cache → it's a highspeed memory
    (Taking instruction. is fast)
    - located in between CPU

Register : located directly inside CPU as Part of CPU core. & it's stored intermidiate result. it's faster than cache.

- & output is also given in hardware That's why it's hardware dependant.



* os does not directly instruct the Hardware. kernel : it's a unit of os but it take order from os

* booting.
- when you turn on PC special Pin is activate.

booting have two types
cold boot → directly Powerd on/set.
warm boot → Restart the PC

* Example of os.
also incloid Tablet.
- mobile os : Android ; iOS, window
- Embedded system os : you can hardly install somthin on it
Here certain functionality Pre define you can use it but not change it from root.
eg washing machine, AC, TV Remote.
you cuh set setting bud hot change.

- Read time os → result in Time bounded
(i) Hard real time → no minute delay eg launch SS space sat
(ii) soft real time(SRT) → it accept minor delay.
eg 9.10.10 we need.
9.10.20 it's ok.
(iii) Desktop os : common personal computer

- Server machine os it's special os used to design or runs over server machine to handle Multiple os. request.

Checksum = integrity TEST.

* Functions of OS.
- everything is performed in OS is process
  creating - Managing → Termination of process.

- Memory management
  - which appl^n ron on which area.

- Device Management (HDD, Monitor, Speeker etc)

* when one process is transfer from one process to another in hand second is called ~~inter thred commun~~ contex switching

* Disk management (Disk sheduling).

- network

external devices attach to your mother bord using socket

* if we accept the cookie Then it came while we Tern on interned anytime. (you see it in Taskbar)

- file management.
- security management
  - using firewall or Antivirus or codes which tells about bad code good code.

* User space & kernel space

kernel is actul driver of your Hardware.

* Memory

TYPES.

Primary
- RAM — SRM, DRM
- ROM - PROM, EEPROM. EPROM.

Secondary.
- HDD
- USB PenDrive
- optical
- CDS

PROM = Programable Read only memory.
EEPROM = Electrically Erasable programmble Read only memory.

volatile memory = Data is erase between These

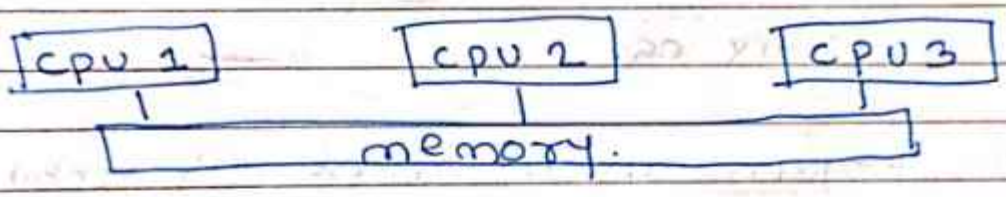
non volatile → Data Delete if you want to delete

* batch operating System
it executes jobs in batches without User interaction during Processing.
- if you increase the ram bigger than harddisk so the seek time is increase also.

* Multiprogramming OS
one process at a time.
capability that allows multipal Program to reside in memory at the same time. at same time & be executed concurrently.
- it maximize cpu utilization.

* Contex switch.
Temporary Store area in ram or Swap area it's ~~at~~ virtual memory.

* Multiprocessor OS
more than one processing element. but they work on same common virtual memory.



why multipal cpu.
① you want to handle more process
② you do not want contexswitch.
· we can achive maximum throughput.
~~number of Process completed by Pc~~
Throughput ⇒ How much instruction executed completely.

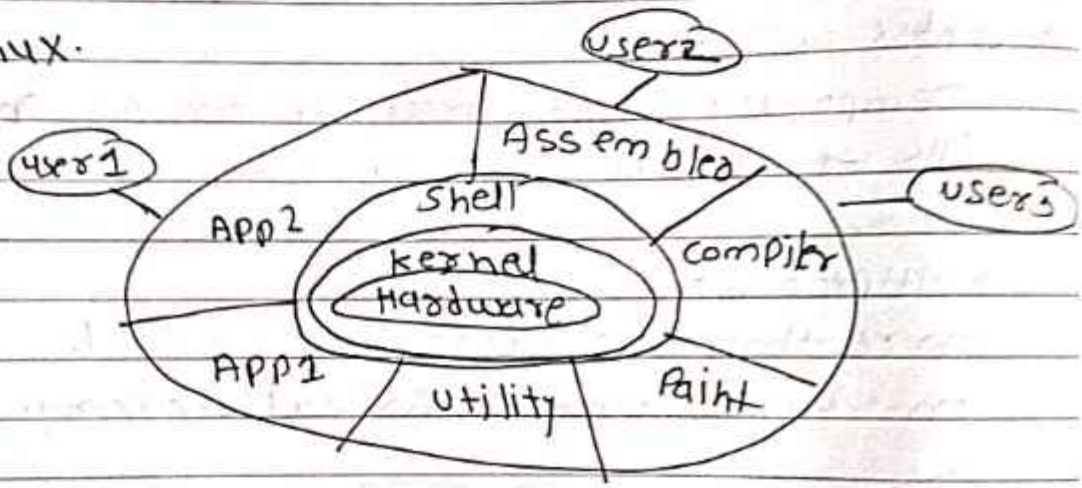
* q what is degree of Paralisum.
→ number of tasks or Processes that can be executed. simultaneously in a Parallel. Computing enviroment.
run multipal operations at the same time.

kernel is never stop.

* Distaibuted os
Distaibuted over a network.
eg Poletics (regional wise distribution country)

but all are connected.

* Linux.



Linux os.

- here Program directly work with kernal & connect with hardware.

- Application directly work with kernel.
- linux is open source operating system
  it's free To use & user can modified it according needs
- The founder of linux is Linus toravards availabal since 1991.

- Features of linux.
  - No cost / low cost
  - security
  - Multi user

Multitasking
Networking → every thing is defind as per Http
CLI as well as GUI
Better file system

- working with basics file system of linux.

/ is root directory.

i) /bin → user binary directory. / basic program.
ii) /sbin → system binary Program
iii) /etc → all configuration file here / login id part.
iv) /dev → device file.
v) /Proc → Process information.
vi) /var → variable file
vii) / tmp → Temp file
viii) / usr → user Program. (Application) we download latter
ix) /home → root directory for user or parent directory of user friendly
x) /boot → Boot loader file.
xi) /lib → System library.
xii) /user/share → App^n support.

* Advantage:-
- Easy To access

commands. it's a editor also.

i) nano filename = open Perticular file if not exits. Then create file

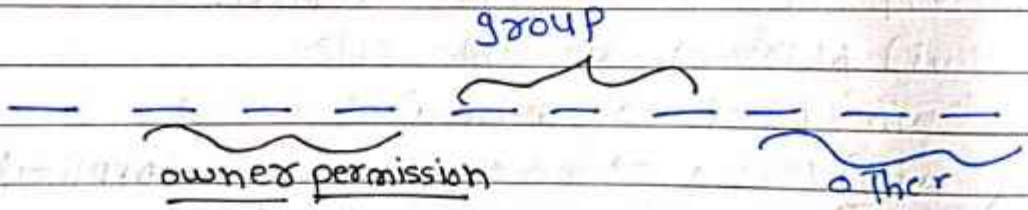
ii) Touch filename = create file only not read file

iii) Pwd : Present working directory.

iv) ls : it list out all the files & diretory.

v) mkdir = folder making. (color is blue)
create new directory.

vi) ls -l → permission owner subowner Time Date
drwxr-xr 2
10 dashes.



* Permision Type
r - Read
w - write
X - Execute.

* group (other user) user within system
~~only read~~.

* other → any user for remote location.

→ gives permission To file or remove /remdir.

* chmod ⇒ a +wx filename.

all author file + write & execute gives Permission.

u represent = owner
g = group
o = other
a = all

+ To give permission
- To remove Permission

by defaut owner get r/w permission. & other only read.

* rm filename.txt → remove.
→ change directory.
* cd Day-1/ → go To ED Day 1 directory
Shortcut cd Day + Tab.

* rmdir → delete directory. (Perticular)

rm filename → remove file
rm -r Directory → remove directory.

* Basic Linux command.

help : shows you basic command & their use.

man : shows you complete manual of that command or program.

ls : list all the folder & files of a directory.

cd : change directory (one folder to another)
cd directory name.

PWD : Print working directory

mkdir : mkdir directoryname.
If you want to move one file to dir
. MV filename directoryname/
mv is use to move

Cp = copy.

touch : touch filename → crete empty file

* User in Linux.

- Regular user → it see in own directory not in others.

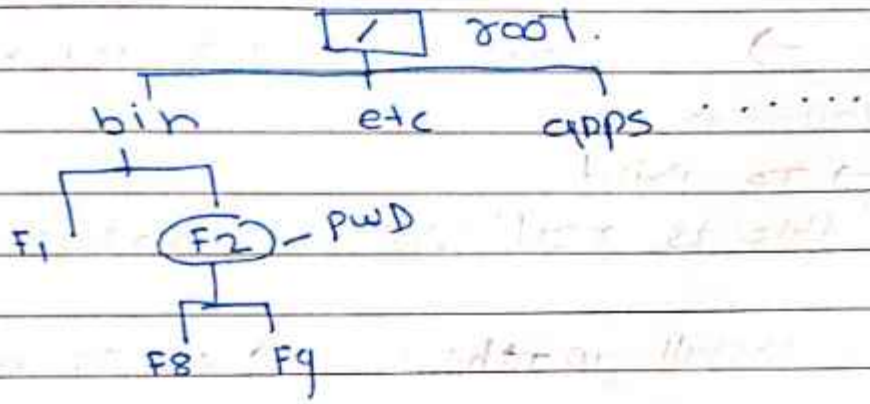
- Root user → full access of system it's a admin

if regular user get root user then you use all command but you need to use sudo

Sudo su → it's a super power. but not recommend because you mistakenly delete another user.

Service user → server runs for services.

* ABSolute VS relative path.



cd /bin/f2/fq it's Absolute
f2 cD f2 relative path.

* apt ⇒ user for update.
sudo apt-get update.
it use TO update only. not install

sudo apt upgrade
-for installation.

* as we know ls gives directory & file.
but if we use
ls -R it' gives director /subdirectory & files in sub directory also
ls -a → To see hidden file

* ls -last → list files of & current directory in the current directory.

-l → list files in long format.
-a → include hidden file.
-r → recursive The order of sort
-t → sort the files by time of modification

* history → it show command you run.

* echo → To print
* Printf "this is real" \n" \n → for next line

* sudo apt install apache 2 → use to install Apa

* top → Process who takes more resource.
* PS → all process run.

* PS -a → background Process.

* vim filename.
sudo su
#
regular user To super user.

Day 2 cdac malkeet singh

classmate Date 28/8/24 Page

* using man command you can read the manual.
  all linux command present in → 1st section
  library function → printf; scnaf, strn, → 3rd section
  system call → 2nd section of manual.

* Day 2 os Module Malkeet singh sir.
  Main user is doing the task.
* sudo adduser user2
  use added user user which I want add.
* adduser is command. → making the command.
* sudo chown user2 file.txt.
  change user → owner change.
* if you want to ~~read~~ see. you need ~~but~~ sudo ls in another directory.

  if you want change a user ⇒ su username
* chown = The change owner of the file.
  su = To switch the user from current to any specified user
* cat = To display content of the file on console.
* Tail = disply n line on console || by default Top 10
* head = disply n line on console || by default last 10

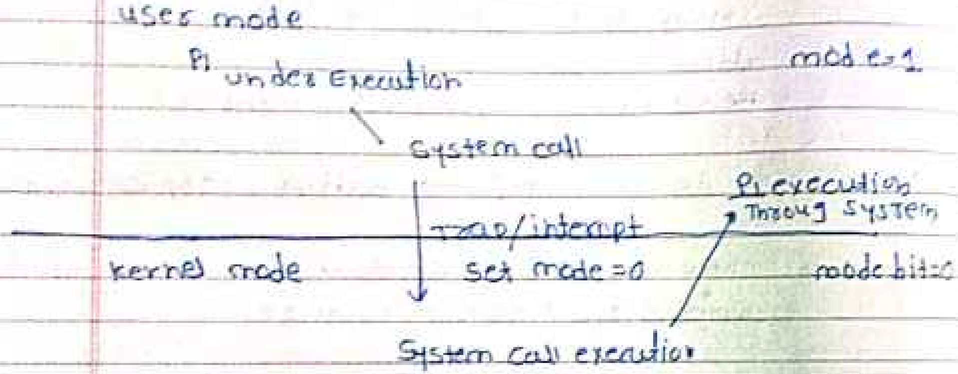
adduser = To add new user into system.
sudo =

* Execution always done in processor.

* Process Management.

System call → it interact with Hardware or kernel To access.

Processor work in Two mode.
user mode & kernel mode & This done by System call.



it's a process management

* when multipal call system have Then it's present in queue (& it use sheduling like FCFS)

* System call is generated when you want any I/o.

* System call ⇒ only method user & kernel communic.
Types
- File related calls : Read(), write(), Delete(), open() close(), create()

- Process Related : new(), fork(), Exit(), wait() Running() etc.

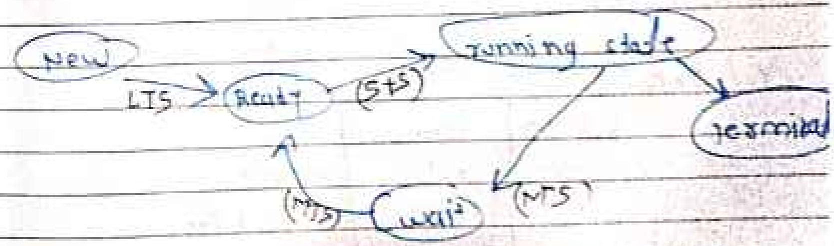
Device related call's : read(), ctrl ioctl(), etc
information related calls : meta data of something getid, gettime, sysdate,
communication related calls : wait(), signal() status()

Process : it's information or code or data which help the Processor to execute or complete The user task.

Process may have
i) codesegment : it have compil code
ii) Data segment : Data for execute
iii) memory structure.
iv) information segment

* for each process PCB is created

* Process life cycle.
New state → Process created.
ready state. → cpu shedule To Process
running state → Processor is executing The Process
Terminate state → complete.
wait state → if not cpu shedule.



Abort thing not resumed.
wait is resumed.

* who does the scheduling.
  5 Process schedular.
  i) LTS (long Term schedulr)
  ii) STS (Short Term schedular)
  iii) MTS (Medium Term scheduler)

if process waiting for io & io is assignd To ano & because of some high Priority Process have io & not assign Then it becomes starvation problem

Premitive scheduling have → starvation problem

* How does The Scheduling
Using Scheduling algorithum.

① Preem-Emptive Scheduling
→ one proces under execution. & cpu take high priority Process That is pre-emptive scheduling

② non Premptive. scheduling.
→ ex. reverse or upper it not assign complete it task

* PCB Process control block.

PID	1001	
	oxffff0	pointer
	Ready	State of Process
allocated register.	A, B, C.	
Program counter	address of next process	
list of allocated resource	Pointer to memory resource.	
	File. BT...	accounting information + Priority

* waiting of smaller Process To wait for execution That effect is called as convey effect.

Accounting information. have number of file open.

* scheduling algorithum
· FCFS.

Arrival Time. (AT)

Burst time. (BT) → Time to complete process in cpu.
compilation time. (CT)
waiting Time. (wt)

Turn around time. (TAT) = CT − At.

PID	At	BT	wt	Tat
P1	0	4	0	4
P2	1	6	3	9
P3	2	8	8	16
P4	3	2	15	17

Gantt chart.

P1	P2	P3	P4

0 4 10 18 20

Avg wt = sum of wt of All Process / Process.

waiting time = cpu Allocation time − Arrival time.

Turnaround time = Arrival time − complition Time.
BT − 0

Algorithms for scheduling [illegible]
Advantage & disadvantage.

SJF

Pid	AT	BT	wait	TAT	completion Time
P1	0	4	0	4	4
P2	1	[illegible]	3	11	12
P3	2	8	10	18	20
P4	3	2	1	3	6
			15 = 4 Avg	36	Avg = 9

Gantt chart

P1	P4	P2	P3
0 – 4	4 – 6	6 – 12	12 – 20

* Round robin algorithm.
a fixed slice of time is given to each an every process that slice of time is known as quantum.

PID	AT	BT	CT	wait	TAT
P1	0	6	10		
P2	1	2	4		
P3	2	6			
P4	3	8			

Slice = 2 quantum in milisecond.

Throughput = Number of Process completed / Total Time

* Arrival time = Process enter ready que.
* Burst time (duration) = Time required by a Process to get execute on cpu.
* completion = Point of Time process get executed
* Turn around Time = completion time - Arrival time. (Total Time we need)
* waiting time = Turnaround Time - Burst time

cpu bound → Tells first how much time cpu required to execute.

I/o bond →

* Responce Time = Time at which cpu get first time - Arrival time.

FCFS

Criteria = "Arrival Time" "Non Preemptive"

Pid	At	Bt	CT	TAT (CT-AT)	WT (TAT-BT)	RT
P1	0	2	2	2	0	0
P2	1	2	4	3	1	1
P3	5	3	8	3	0	0
P4	6	4	12	6	2	2

Gantt chart.

P1	P2	idel	P3	P4
0 – 2	2 – 4	4 – 5	5 – 8	8 – 12

time →

जिस process का burst time कम वो ...

* SJF Burst time Non Premtive — no interupt. (means no interupt.)

Pid	At	Bt	Ct	TAt (CT-AT)	wt (TAT-BT)	RT
P_1	1	3	6	5	2	2
P_2	2	4	10	8	4	4
P_3	1	2	3	2	0	0
P_4	4	4	14	10	6	6

Gantt chart

idel	P_3	P_1	P_2	P_4

0 1 3 6 10 14

Time →

BT P2 P4

* SJF Mode Preemptive

Process	AT	BT	CT	TAT	wt	RT
P_1	0	5	9	9	4	0
P_2	1	3	4	3	0	0
P_3	2	4	13	11	7	7
P_4	4	1	5	1	0	0

→ Gantt chart

P_1	P_2	P_4	P_1	P_1	P_3

0 1 4 5 6 9 13

✓ P_1 = ~~5~~, ~~4~~, ~~3~~, ~~2~~, 1
✓ P_2 = ~~3~~, ~~2~~, 1
✓ P_3 = 4, 3, 2, 1
✓ P_4 = 1

SJF	AT	BT
P_1	0	7
P_2	1	4
P_3	2	8

P	P_2	P_1	P_3

0 1 5 11 19.

P1 1 2 3 4 5 6 7
P2 1 2 3 4
P3 1 2 3 4 5 6 7 8

* loss of memory inside because fixed size These is memory fragmentation.

fixesize Partision

memory is of computer system canbe devided into block of fixed size or variable size called ~~fixed size~~

Suppose each block is 4byte

Fixed size Partisioning : memory into fix size block but in fixed size if block is 4 byte & Process is get memory if 3 Then 1 byte wast & Process must have small or equal To block size. ~~if not~~ is internal fragmentation.

* variable size Partitioning.

invariable size partitioning ~~if block size same.~~ ~~& we can not assign~~ These during contigious. memory allocation The external fragmentation can happen.

* internal fragmention happen anywhere. block size is ~~also~~ small Here.

* if (internal fragmentation) not have solution.

* solution of (EF), also known as defragmentation. compaction = algorithm in EF all Process shift one side & blank in another

* but it's Time consuming, CPU Time required is more.

* First fit → first block according To it size it's equal or greater.

sudo → admin user.
in sudoers file all the sudo commands are given.

Most IMP Page

Hostname of server

* cdac @ Gaurav →
user name of current user

+ if you want to create one user use.
  * sudo adduser username
  * sudo adduser user2

+ if i want to chang owner.
sudo chown user2 file1.txt
which owner you give — which file

- we we create user in eg cdac then group is also cdac &
- if we create user using sudo then it also go in cdac group.
you can create as much as user & as much as group you want.

- Su → switch user — su user3
in these user. is → Permission denied.

① user 3 is in cdac directory of cdac use can not change list out file & directory. (you can not access because these directory does not belong to you.
for these we give sudo command to that user.
sudo ls.
How user given sudo permission. → in file
so go to root user. if you want to give permission to the user you need to give permission in sudoers file. that file in etc directory.
cd / nano cd /etc ; ls , then sudo nano sudoers then privileges add user name in privileges
Beware of using sudo command. / sudo rm -

Assignment 1.

a
→ i) Pwd
ii) mkdir my Assignment
iii) cd my Assignment.

b
→ cat > file1.txt or touch file1.txt
display content = cat filename

c → create new directory.
mkdir linuxAssignment/docs

If you want multiple sub directory.

-mkdir -p Parent/child d/sub_c_1

d → copy file.txt file into doc directory & rename 2 times
→ cp file1.txt doc/file2.tx

e
→ chmod 744 file2.txt
owner r, w, x
group r
other r

F
→ ls -l l see all content.

File searching.
g → find file with extension (.txt)
→ a → search for file with extension. .txt

find . -type f -name "*.txt"

'.' current directory.
-type f files (not directory)
-name "*.txt" = txt extension file.

2) Display lines containing a specific word in file.
- grep "word" filename

System info.
H. current system date & time.
→ simply write date

Networking.
i ip address of system
→ ip addr show

To check Ping
Ping remote servername.
Ping google.com.
& ctrl+c To stop.

j file compression.
a → docs into zip.
zip -r docs.zip doc → directory name
-r → recursive.
docs.zip → newly created zip file.
docs → directory name

& To extract. mkdir extracted-documents.
unzip doc.zip -d extracted_docs

k file edting.

→ open file1.txt in editor & add some text to it.

a→ nano filename.

b→ sed -i 's/welcome/gaurav/g' file1.txt

-i → sed file edit conformed

s/welc/gaurav/g → change welcom to gaurav.

g is to change occurence.

file1.txt → file name where you want To change

Q2

a

→ inpute file data.txt

* Sunbeam 2 Day.

Touch f1.txt f2.txt file.txt

ls

How do i see only txt file.

ls *.txt ∴ * = any number any character

am looking for all .txt file starting with F

ls F*.txt

only Print that file having only one letter after F

~~ls F*~~ : ls f?.txt ? = any single charater

what about Three

ls f???.txt

These is refered as shell wild card character.

*	any number of character.	(% sign)
?	single character.	(_) in RDBMS

IMP

Redirection

Inpute redirection = instate of taking input from terminal we take inpute from file.

output redirection = instate giving outpu To terminal gives to file

~~Err redirection (2>)~~

command > outfile → it replace.

command >> out file → if you want append

cat : it's simpul command Take input & gives out same time in terminal. (echo The same)

cat > text.txt

so now output is go to These file > output redirection

Hell dac

welcome dac

Ctrl + D

who = how many user on our system

cat > text.txt

linux common.

ctrl + D — Then old data gone That data over ride. So That use >>

cat >> text

read manual of command

ctrl + D //Append it

Err redirection.

command 2> errfile

2 because it's std err 2

eg sort < in.txt > out.txt 2> err.txt.

(I) sort
(I) read command in.txt
(II) give output out.txt
(IV) if err keep it err.txt

in redirection data Travel file & Program & Program & File

* **Pipe**

1^{ST} command inpute output is 2^{nd} command output Then we use Pipe.

who | wc — output of who in Pipe; Take inpute from Pipe

(it's use in inter Process comm in Java also)

echo$? → gives previous comand execute or not.

0 = Success

1-255 - failure.

*. Test command exactly same These phenere.

Pipe

who give output | Take inpute.

is left side

terminal ls/home → it successful

but if ls/hom → it fail

but in future if you know How to check our command is sucessful or not.

→ echo $? → $? (special shell variable)

→ it Tell previous command Pass or fail.

output 2 it's fail

any non zero number it failure.

0 = successful.

- terminal > man 1 test

terminal > test

Test feature- we can check equality of number using test (int)

- we check it file or directory.

- file is executable or not

Terminal > Test 12 -eq 12 #

echo $? # 0 Then true.

Tes 12 -gt 12

echo $? #1 false.

in shell Test command run automatically.

* if first command is successful Then & then execute Second command.

command1 && command2

eg Terminal Test 12 -eq 12 && ls/home
successful && gots 2
& it give second cond^n output.
Test 12 -gt 12 && ls/home.
failed

In 2^st eg is ls/home execute because Test command is execute.

not second because Test is fail.

use. → if you have path Then check is it file or directory.

check for directory ⇒ test -d /home && ls/home
(if /home is directory (-d) if it's director Then list it content) Then it execute.

* we can use it for or also.
if first command is fail Then Try second one

Test -d fruits.txt || cat fruits.txt

* comman &.
Puting & in any command in last.
it called asynchronous command execution.

By Default, shell weighs for commad to be completed.

MCQ for These concept.

— in home directory we find file.txt present or not

eg find ~ -name "file.txt

* Try in GUI of linux not in command line.
Terminal > firefox.
(I) # firefox open but shell prompt is not available
(II) # Shell is waiting for firefox to complete.
once a firefox is closed, shell Prompt will appear again.

now
firefox &
shell Promped open ~~but~~ shell Prompt is also open
These feature is asynchronous.
- Here shell doesn't wait command to execute.

* Regular expression.
How to write.

grep → GNU Regular Expression Parser.
grep -E egrep → Extended GNU regular expression Parser.
grep -F fgrep → fixed GNU regular expression Parser.

Terminal > cat > food.txt
this
biscuit
is not
tasty,
but
that
cake.
is
relly good
ctrl D.

Print all the lines which contain is

grep "is" food.txt
command content to find file

at the beginning is

grep "^ is" food.txt
^

is at end
grep "is$" food.txt
$

only is present.

grep "^is$" food.txt.
^ $

* Regix wild card Characters
1) $ → end with
2) ^ → start with

example 2.

Terminal > cat > bug.txt

bag
beg
bug
big
bog
bg
b*g
ctrl D.

* . any one character → regular expression wildcard
* [ ] → it called scansed

cat > bug.txt
find = all line in between b & g any
one character is there

grep "b.g" bug.txt
. means any one character

any one alphabet.
if find b anything between b&g. | grep "b [a-z] g" bug.txt | NOT a-z it print [b*g]
i want only with (a i u)
grep "b [aiu] g" bug.txt

find "b*g"
terminal > grep "b * g" bug.txt

1) \ = "\" remove special meaning of wild card character.

2) using fgrep.
no meaning of special character is given.

fgrep "b*g" bug.txt
it search given word as it is .. no wild card meaning.

* Regex wild card chaster.
. → any one character
[ ] any single char in given scan set.
a) [^] not in scanset.

grep → work with only basic wild card character
egrep → work with extended wild card character.

eg3 cat > big.txt (Here we check repetation of previous character means for(i))
bg
big
biig
biiig
biiiig
biiiiig
biiiiiig
biiiiiiig

* How many time i repeat.

Terminal > grep "bi*g" big.txt

Zero (0) or multiple character occurance of i

* zero or one occurance of i
= Terminal > egrep "bi?g" big.txt

? = 0 or one occurance
it's a extended regular expression.
so use egrep.

* one or more occurance.

bi+ egrep "bi+g" big.txt

* exactly Three occurances.
egrep "bi{3}g" big.txt

* Three or more occurances. → bi{3,}
egrep "bi{3,}g" big.txt

* 3 or less occurance

egrep "bi{,3}g" big.txt

* 3 TO 5 occurance
egrep "bi{3,5}g big.txt

Note when we use egrep we use extended wildcard character. They work with egrep.

* Find "cake" or "biscute" or "good" from food.txt
Here is no pattern

command → egrep "(cake|biscuit|good)" food.txt.

it show me whole line

* How many Time pattern is repeated.
Previous example in bug.txt

grep -c "b[a-z]g" bug.txt

-c is use

* -n Tells at which line data repeated.

grep -n "b[a-z]g" bug.txt

regular operation commonly use to
operation.

* multipal file in a folder & you want to see

grep -h "big" *.txt

= find big in txt files

* directory/subdirectory : search.

grep -R -h "big" ~/directory Path

* regular expression wild card.

* = 0 or more occurance of Prev char

+ = 1 or more

? = 0 or one occorance

(w1|w2|w3) → find a word from w1, w2, w3

\ → remove special meaning of wildcard character.

Buiding regex exact 10 digit number

" ^[0-9]{10}$"

* vi editor question came in exam.

google vi editor mcq & read for exam
that guranteed comes.

* vi editor. vim → vi improved

Developed by Bill Joy

he is also Sun microsystem founder

Student → working on BSD unix & he can rewrite in vim editor

* operation command

create/open file → vim filepath

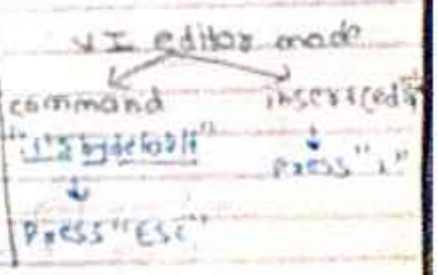
write/save → :w

quit → :q

write & quit → :wq

quit without saving → :q!

save quit all file. → :wqa



* ## vi editor

content in .vimrc file are automatically executed every time vi editor is started. it located in user home directory

* Here follow space otherwise error arrows). hidden file

Terminal: vim ~/.vimrc

set number
set tabstop=4
set shiftwidth=4
set autoindent
set nowrap
set autowriteall
syntax on
esc :wq

OS Day 3 CDAC-M Rishabh Singh

* real country effect
  extendability
  Round robin disadvantage
  examples of algorithms

fork → fork is use to create child process from the parent, execute concurrently.

what fork does → what a child process does it does exactly the same for which we have defined the parent process.

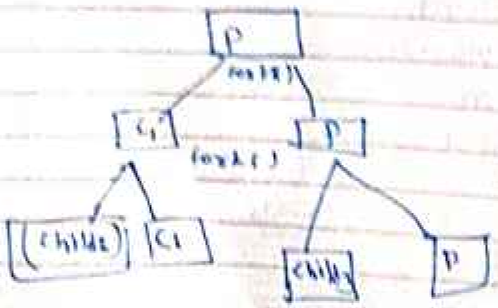
* execute c program
  Name program.c
  gcc -o program program.c
  ./program child ric

  using fork.

⇒ but which execute first we cannot say.

code: int main()
    fork();

fork



MCQ exam

n = 2 process

How many
Time rep
repeat
2^n
$2^n = 4 \cdot 2$

$(2 \times 2 \times 2) - 1$

How many child process make (number n = 1)

$2^n - 1$

eg $2^1 - 1 = 1 = 2$, n = 3

n = number of fork used

if fork use 3 time,
$(2 \times 2 \times 2) - 1 = 7$

-1 if child not created.
0 if child created.
1 for parent process.

* what is Zombie process.

The parent executed & it's still in run but wait for child to execute. Then parent becomes Zombie.

* orphan process.
  Parent execution is completed & the child process is orphan process.

* Paging
* deviding the process by pages in modules & putting them in main memory. Then these is called Paging.

  insted of loading whole executable code into main memory it just load only only needed pages.

  that is better use of memory & it gives maximum throughput.

= Pages have same size
= Frame size == page size.
due to this no internal as well as external fragmentation

* ~~contiguous length~~

program divide pages
memory divide frame
* Fixed size paging.
* Size of frame equal to size of page.

- The size of pages ~~is unique but all~~ have same size

* we require a separate memory logical & physical memory
logical = addressing used by program (virtual)
physical = actual address.

* use of paging
all programs should be in [illegible]

- all computer systems is byte readable [illegible]

0 1
2 3
4 bit
address

* The physical address is also change according to pages

* CPU request mapping

CPU generate logical address
we load pages to [illegible]
but the [illegible] of the [illegible] in memory

This done by MMU (memory management unit)

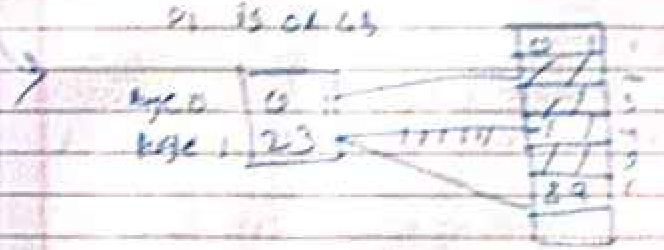
[illegible]
generate table & it page table generate for the process.

Page table -> [illegible] mapping of pages & frame

* The technology page . is 64 bit that called mapping

CPU [illegible]
logical address

P1 is of 4 [illegible]



Page table | Frame
Page 0 | Frame 0
Page 1 | Frame 1

* If CPU demand any page & it not found in main memory than called Miss.
& find is hit.

LRU = Least recently use
M.R.U = most recently use

* what is virtual memory

1 in context switch process Temp save into main memory.

& that memory is virtual memory.

- virtual memory is Twice of your ram.
- Here execution is not done. only store running Process.
- it flash out all info when you shutdown.
- virtual memory also present in harddisk etc

* Page fault.
  if Page is not into frame where CPU look for it is called page fault.

* if Page is not in the frame is page fault.

if you have more [illegible] you can.
32 bit system → [illegible] address can save.
64 bit system → more memory [illegible] to save data.

→	2	3	2	4	3	5	4	3	2	7
frame 1	7 8	8	8							
frame 2	4 3	3	3							
frame 3	2 2	2	2							
frame 4	6 6	5	6							

miss & hit calculation.

* Paging numericals do you own.

* when you want to write more than one command all at once then shell programming is used.

* nano sh2.sh

```
echo Enter your num
read Name
echo "Welcome, $Name"
```

bash shp2.sh.

*

```
echo Enter num 1
read num 1
echo Enter num2
if [$num 1 > $num2]
Then
    echo " num 1 is greater"
else
    echo " num2 is greater"
```

hacker rank 2 - 05.

Q Display The line from 12 TO 22 of text.txt
⇒ cat 2 text.txt
1 To
22 line.

command: head -n 22 text.txt | Tail -11
(pipe)

Q. last 20 character of input file.
→ tail -c 20 file.txt

Q in given fragment of text, replace all sequence of multiple spaces with just one space.
→ echo "Hello   llo
  wor   ld
  how   are you." | tr -s ' '

Q sort decending

sort -hr text.txt

Q find file
in file only print line those which have that The those word.

grep -i -E 'That|The|those' file.txt

i = for insensitive
E = Extended

command what is sed, tr

yy = yank.

* vim editor command.
vim hello.h hello.c — open file (multiple)
* vim command mode (Esc)
:ls
:next — go To next file
:wqa Prev
:wqa — write quit all
:qa — quit all

vim -O hello.h
open file in vertical tabs. ? -O (capital)
open file in horizontal tab ⇒ -o (small)

To switch between tabs use ctrl + w w

*. copy past commands

include <stdio.h>
include "hello.h"
int () {
  Print (" VI has two mode insert mode & command mode");
  return 0;
}
int sum (int a, int b) {
  return a+b;
}

yy To copy (go in That line Then yy)
p for Past p

(To copy line (multiline)
:6,9y
6 To 9 copy. & P

hacker rank 2 os:

Q display the line from 12 to 22 of text.

⇒

cat > text.txt
1 to
22 line.

command: head -n 22 text.txt | tail -11
pipe

Q. last 20 character of input file.

→ tail -c 20 file.txt

Q in given fragment of text, replace all sequence of multipal spaces with just one space.

→ echo "hello llo
wor ld
how are you. | tr -s ' '

Q sort decending

sort -hr text.txt

Q find file
in file only print those which has that the those word.

grep -i -E 'that | the | those' file.txt
/
i = for insensetive
E = Extended

comment what is sed, tr

yy = yank.

* vim editor command.
vim hello.h hello.c open file (multiple)
vim command after (ESC)
:ls
:next go to next file
:wqa prev
:wqa write quit all.
:qa quit all

vim -O hello.h
open file in vertical tabs. ? -O (capital)
open file in horizontal tab ⇒ -o (small)

To switch between tabs use ctrl + ww

* copy past commands

```
include <stdio.h>
include "hello.h"
int () {
    Print (" Vi has two mode insert mode & comand mode");
    return 0;
}
int sum (int a, int b) {
    return a+b;
}
```

yy To copy (go in that line type yy
& for past p

if you copy line (multiline)
:6,9y
6 to 9 copy. & P

* copy Past in vi editor

yy → copies current line
4yy → 4 line from curser
:6,9y → 6 T 9 copy line
P → Paste

yw → copies current word.
3yw → copies 3 from curser.

u → undo
ctrl+R → redo

replace y with d is use for cut

if we want to jump line number
using +36
go To

logical = address
physical = memory

FIFO.

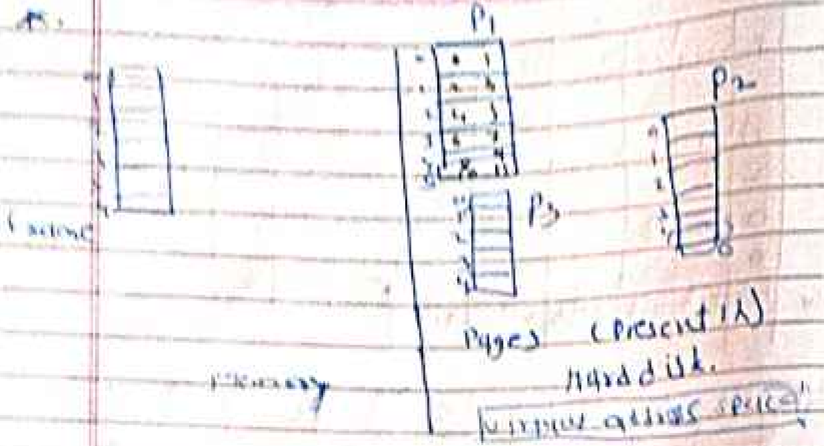
Eg.

P1	0	2	1	2	0	3	5	3	2	1	0
frame 3			1	1	1	1	1	1	2	2	2
2		2	2	2	2	2	5	5	5	5	0
1	0	0	0	0	0	3	3	3	3	1	1
	miss	hit	miss	hit	hit	miss	miss	hit	miss	miss	miss

	0	2	1	2	0	3	5	3	2	1	0
4						3	3	3	3	3	3
3			1	1	1	1	1	1	1	1	1
2		2	2	2	2	2	2	2	2	2	0
1	0	0	0	0	0	0	5	5	5	5	5
	M	M	M	M	H	M	M	H	H	H	M

* Belady's anomaly. when we increase the number of frame the page fault ratio is decrease in case of first in first out.

virtual memory is present in hard disk (hard memory)



- virtual memory is illusion to the computer system you can execute the process which is larger than physical memory (RAM)

- instead of loading full process loded to RAM instead it loded in VM in hard disk. & These process in stored in pages. (fixed size of pages)

If page size is 2B.

q - Bit addressable

q 1GB system

it physical memory also deviding into frame Then paging use.

* System byte addressible.

0 - 16 = 4 bit

0 - 16 = 16 bytes

what is byte addressable.

logical address | page no | offset |

Page offset which bite you want to read

MMU → convert logical address to physical address.

Page number	
0	
1	
2	
3	frame 1

IMP:

(6) if we have logical address space 4GB & page size is 2 Byte how many pages generated

16 bit required 4 byte

1 GB ⟶

16 को दिखाने के लिए 4 byte चाहिए है
(4 bit को दिखाने के लिए कितना चाहिए)

4GB के 2 byte part. कितने होंगे
So To

(16 को दिखाने के लिए 4 bits चाहिए
तो इतने दिखाने के लिए कितने bit चाहिए.

eg 4 GB (4 × 1024) = MB $\underline{50}$ × 1024 = kB
× 1024 bits
____
2

Suppose SRC $\underline{567889}$ 2 is for page

567 884, Thalwsh space we make

How many bits we require To create The

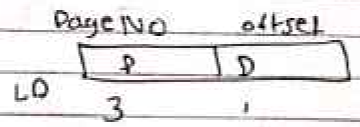
How many we required

~~we have To enter~~

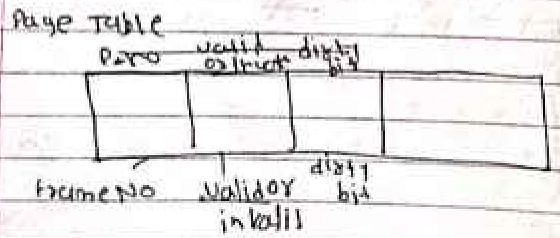
* Page table also present in ram.
  Page Table give frame address.

  if Page present Then ok.
  but if not Then it is a **Trap**

eg CPU
Process 1



Page Table



delete Temp file program loding will Take Time.
because it load Total Process.

* Translation look aside buffer is nothing but cache.

* If Page is Present Then it value 1 if not 0

* in mainmemory came & cpu change it Then it's dirty bit.
  if during The execution The page content is updated & Then MMU so chang 1 To dirty bit. & it value chang in logical memory also

Notes **IMP**

* hardware required for Paging (virtual + cache)
* Translation look aside buffer.
  - it find Physical address for given logical address.
  - in paging we load only few pages in main memory of perticular process. & Entries of that Process made in MMU Pagetable is present In Physical memory so To find a frame MMU look first at Pagetable & according To Page table entry it access the given frame no & These Process is actually access The ram twise. so To reduce These excess Time as use Translation lookaside buffer in cache

Virtual memory-
  It's a memory present in hard drive. which work like Physical memory to entertain large Process. whoes size bigger Than main ram
  - in VM Process devided into pages & os load pages from V.M To Physical memory on demand of cpu as known as demandpaging.

- While working with virtual memory pages are sent as per their logical order.

- While taking the pages from virtual memory to physical memory the pages get physical address.

- If process page loaded from virtual memory to physical memory is called Swap in.

- If page shifted from physical memory to logical memory of replacement process that is swap out.

- Dirty bit
  It is a bit in page table which describe whether the page content is updated during the execution or not. If for specific frame content is updated then content is set to 1 otherwise is always zero.

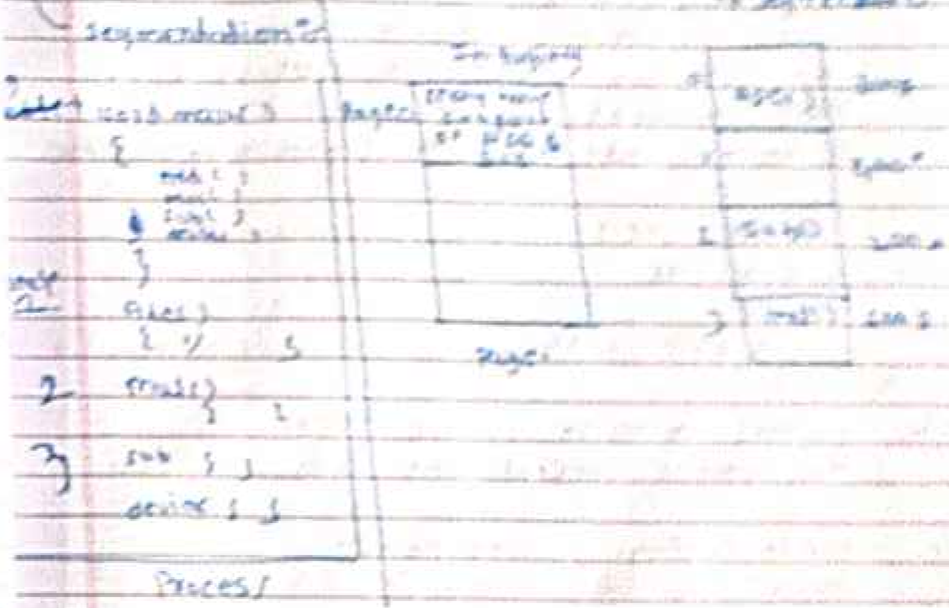
* Shared pages & reentrant code

  It is some technique making pages which content of one or more process is shared page.

Paging has that code is divided in pages without knowing of structure.

Instead of divide process of fixed size Paging divide process in variable size Partitions Segmentation.

Segmentation:



* Segmentation is a technique of dividing a process into variable size Partition known as Segment. Segmentation also presents virtual memory.

* To handling segmentation memory should be variable size. Partitioning.

Here mapping also Do with more.

Paging is done based on OS & segmentation is more useable.

Large Page Table ⟹ If Page table size is more than frame size Then you should be maintain another pages That is called multilevel Pages & 2 level paging.

echo = " " or no need.

-it no need for extension

* Shell Programming
- shell is interface between user & kernel.
- An user can interect with shell by using Shell command or shell script /Program.
- it take inpute from user & pass it to To kern

* list out shell
  cd /etc
  ls
  cat /etc/shells.

*. no need To assign variable Type.
The variabl present until shell on.

create directory. shell Program.
nano file1.sh.

* #!/bin/bash. → first line. enter in "

echo " Enter your Name"
read str   read input from user
echo welcome, $str1

execute that shell
↓
bash file1.
Shell name. filename.

-eq = equal
-gT = greater.

nested question for lab)

* echo "enter num1"
read num1
if [$num1 -eq 5]   if _ [space space] space
Then
echo " num is equal"
else
echo "number is not equal"
fi   end of fi

* echo " enter num1"
read num1
echo " enter num2"
read num2
If [$num1 -gt $num2]
Then
echo "num1 ig greater"
else
echo "num2 is greater"
fi
execute bash file name.

*. cat p3 > p4   (If else
nano p4

echo num1
read num1
echo num2
read num2
echo num3
read num3

if [$num1 -gt $num2]
Then. if [$num1 -gt $num3]
Then echo Num1 is gretest
else echo num3 greatest
fi   - fi close.

if( )
{ if( )
  else
}
else
{ if( )
  else
}

Degree of multiprogramming

fi
else
if [$Num2 -gt $Num3]
then echo num 2 is greater.
else echo Num is greatest.
fi
fi

* loop.

x = /100
for i in 1,2,3
echo $x

rbash => restricted version of Bash the limits the user's ability.

* Throttling ~~is related to degree of multiprogramming~~
→ Throttling used to limit the rate at which a system or process consume resources.

* Shared Pages.
Shared Pages are memory pages. That are shared between multiple process on a system. ; They stored in ram

it reduce the memory footprint of system as multiple process use the same data

Round Robin question. Time quantum, Preemptive

PNo	AT	BT	CT	TAT	WT	RT
P1	0	5				
P2	1	4				
P3	2	2				
P4	4	1				

TQ = 2.

Gantt chart | P1 | P2 | P3 | P1 | P2 | P1 |
0 2 4 6 8 10 11

P1	1	~~2~~	~~3~~	~~4~~	~~5~~
P2	~~1~~	~~2~~	~~3~~	~~4~~	
P3	1	2			
P4	1				

Gantt chart → P1 P2 P3 P1 P4 P2 P1

Ready que
~~P1~~ | P2 | P3 | ~~P4~~ | P4 | P2 | P1
0

Priority. ProcNo AT BT CT TAT WT

10	P1	0	5/14	
20	P2	1	4/13	
~~30~~	P3	2	2/11	
40	P4	4	1	

Higher the num Higher Priority.

Gantt chart | P1 | P2 | P3 | P4 | P3 | P2 | P1 |
0 1 2 3 4 5 6 9 13

P1

Gantt chart = | P1 | P2 | P3 | P3 | P4 | P2 | P1 |
0 1 2 3 4 5 8 12

number sum 1 to 10 (Fibonacci) (type ...)
using for. 1 to 50 (Prime) (greater)
[for else] - if else
→ odd in natural num. / even odd.
~~find duplicate character~~

*. for loop. bash shell = for control by variable

for loop syntax
```
variabl = 0
for a in 1,2 3 4 5 6
do
echo $a
done
```

→ *.
```
#!/bin/bash

a = 0
~~sum=0~~
for a in 1 2 3 4 5
do
echo $a
done.
```

→ sum of
```
a = 0
sum = 0
for a in 1 2 3 4 5
do
~~echo $a~~ echo $a.
sum ='expr $sum + $a'
done.
echo sum is, $sum
```

operator -gt, -eq, -le, -lt, -ge

for model exam. 10 question

* Practice on
- sum of n odd & even numbers.
- To find a prime number is or not
- Fibonacchi.
- factorial.
- leap year
- Armstrong number
- Table multiplication.

* while syntax condition is given is [ ]

```
sh
while [ ]        it con true
do
 statement.
 increment statement.
```

print it 10 time.
```
#!/bin/bash
a = 0
while [$a -lt 10 ]
do
echo $a
a = 'expr $a+1'
done.
```

for increment we use. a = 'expr $a =
a = 'expr $a+1'

for execute shell [./filename] also use There

result of the command.

→ ~ goto home directory.

* until

```
#!/bin/bash
a=0
until [$a -lt 10]
do
echo $a
a=`expr $a + 1`
done
```

* wildcard character
  - `*` any number of characters
  - `?` one number

→ touch ... files file4 sell sets ...
ls fil*

in nano file1
  hi ...
  h ...
  hello ...

IMP points

→ line search inside any file.
→ starting with some character & end with that character

→ cat file

cat file2 | ...
grep ^h..n

① A ... with ...
② starting
③ $ use for end

→ cat file1 | grep "h.*n$"

wildcard

line starting of file
cat file1 | grep ^h
→ searching & sorting.

* How find file start with
ls f*

** command line argument
create file
echo hello
echo hi
echo ok
echo bye

bash file

while running ... give ...

bash ... ...

hello
hi
ok
bye

... if you want to add ... then you should ...

eg.
#! bin/bash.

echo Number of Parameter , $#
echo script name , $0
echo hello , $1
echo hii , $2
echo ok, $* all Parameter.

Num of Para , 3
script Name , 3
script Name , P4
Hello , malkeet
Hii Sandeep
ok , Rohit
Bye , malkeet sandeep Rohit

* $# = No of Parameter Passed.
* $0 = script Name
* $i = where i can be any number, it will return that Parameter
* $* = Give all Parameter Passed.

Test command.

is file Present or not test.

test -f abc.tx.
Find & is Prest.
Test command related to file.

Pratice on True else

| in unix True = 0 false = 1 |

Test for equal or not.
nano x=100
y=200
if test $x -eq $y
Then
echo " x and y are equal"
else
echo x and y are not equal.
fi

file is parameter not
echo

IMP.
* Test script accept
Two word from command line & check
Those word is greater then or not.

bash P0 #! /bin/bash mobile Prob
if test $1 > $2
then
echo x is greater y
else
fi
bash P0 malkeet Sandeep

* if you contain you have to use mathematical operator
(= =)

Things to do
Integer vs string number checker using Test

* Please use -eq , -gt , -lt , -ge , -le then operators with Test command

- The whole universe is automata
& it works on regular.

→ a(a+b)* → * any number . regular expression.

Please. >, <, =<

* either string / empty or not.

① Take two string from command like.
& compair questions.

Regular exp.

grep "s → filter pipelin & get.
The word which start
with s.

grep ^n-*t$: it will ~~print~~ filter.
The pipelike & get the
likes which are starting
with n & ending t

ls f* = all file starting with f

q input which take only letter A or a.

*. Arthematic expression
-$a +$b
$a - $b
$a / $b
$a * $b

compli- two num - comment & like $1 $2 classmate
Print * pattern.
Fiboracle series.

Passing some command line 3/4

nano p7.

#!/bin/bash.

echo "Enter a num"
read num1
echo "Enter a num"
read num2

result = `expr $num1 * $num2`
echo Result $res
bash p7.
100
200
Result 300.

in mathematical expression *

+ networking command: it's used to take control of
Telnet → remote server
Ftp → it allow user to send or recived file from remote server
ssh → it's use to take actrol of server or machine
stfp → it's use to upload or download from remote server
finger

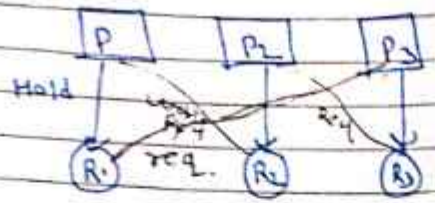
You can save path direcly by 'Path' command
Path = "program files / Java / jdk / bin"

system variable like = PS1, PS2 etc How to set them

multilevel Aging

in Pendrive it 8gb we not get whole space because it another Pages is covers by Pages.

*



- Deadlock is situation where one or more Process holding by some resources but wanting another process Then To execute This which is hold by another Process.

Necessary conditions of deadlock

* Mutual exclusion.
Process are shared
any Process request any resource
= if one resource any Process have Resource. Then another Process request but only holding Process have only write to execute.

* No Prompt tion
if a Process is holding a resource no other Process can Prempt it.

* Hold & wait
Process are allow to make request for another Process resource while Holding a resource

circular weight
while holding & wating for resources there should a circular wait.

* Deadlock Handling Technique.

1) Deadlock Avoidance or ignorance.
if problem came in 5yeas why i should write code. if we write

*) Deadlock Prventaion.
i) No mutual exclusion
or Pre-emptive or No Hold & wait & No circular wait.
ii)

* Deadlock Recovery. (self study)
i) Kill a process
ii) Add multipal instance of Process (means increase The Processes) if we increase Resource Then our cost increase.
iii) Resource allocation graph
iv) Bankers algorithm.

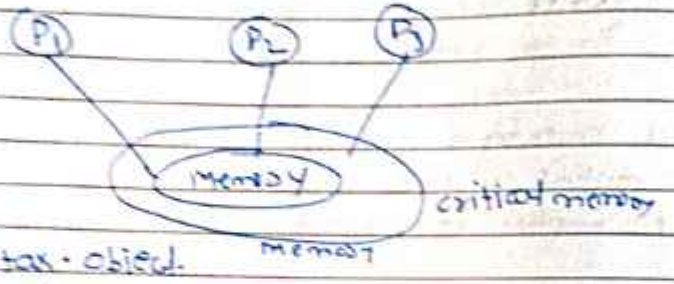
* Starvation have hope that process cpu ged.

* Dead lock is infinite Wait.

* Process synchronization.

if you having critical resource in which you have to allowed process one by one. that is called synchronization.

* mutex = locked



Mutex - object.

Semaphore → variable which denoted by S

wait ()
Signal()

if p_i has raise wait for P_i resource it means lock the R_i

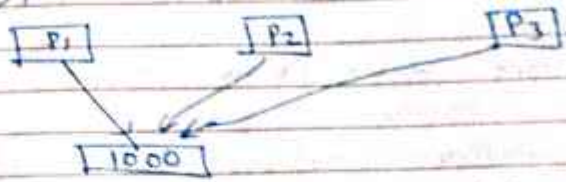
```
wait (Mutex) lock.
{
   Read → CS
   write → CS

   signal (Mutex) key
}
```

what is buffer

Mutex →
semaphore → semaphore type (read it)

semaphore (S)



it use both wait & signal but with S

wait(s)

eg
```
wait (S)
{ while (S >= 0)
   S-
      CS - Read
      CS - write
   } signal (S)
if any time Access
{ S ++
}
```

Semaphore we to maintain by wait & signal of Process.

* Producer consumer problem.

```
void Producer( )
{
  int itemP = new item()
  while (1)
  {

  }

Produce item stored in
STACK.
```

```
void Producer( )
{
  int itemC;
  while (1)
  { while.

  }
```

(*) Mutex & semaphore
- initial value
- which method increases the value of semaphore

case 1 No interruption in producer or consumer method

case 2 if producer is interupted while updating count
Here race condition occure.

Solution is for race condition is mutex semaphore you have to make memory in critical section.

- Process synchronozion is the only solution to solve race condation.

Command & shell programming Practic

Q. loop 1 TO 99 only odd number.

```
for i in {1..99..2}
do
echo $i
done
```

Q what is shebang
→ #!/bin/bash

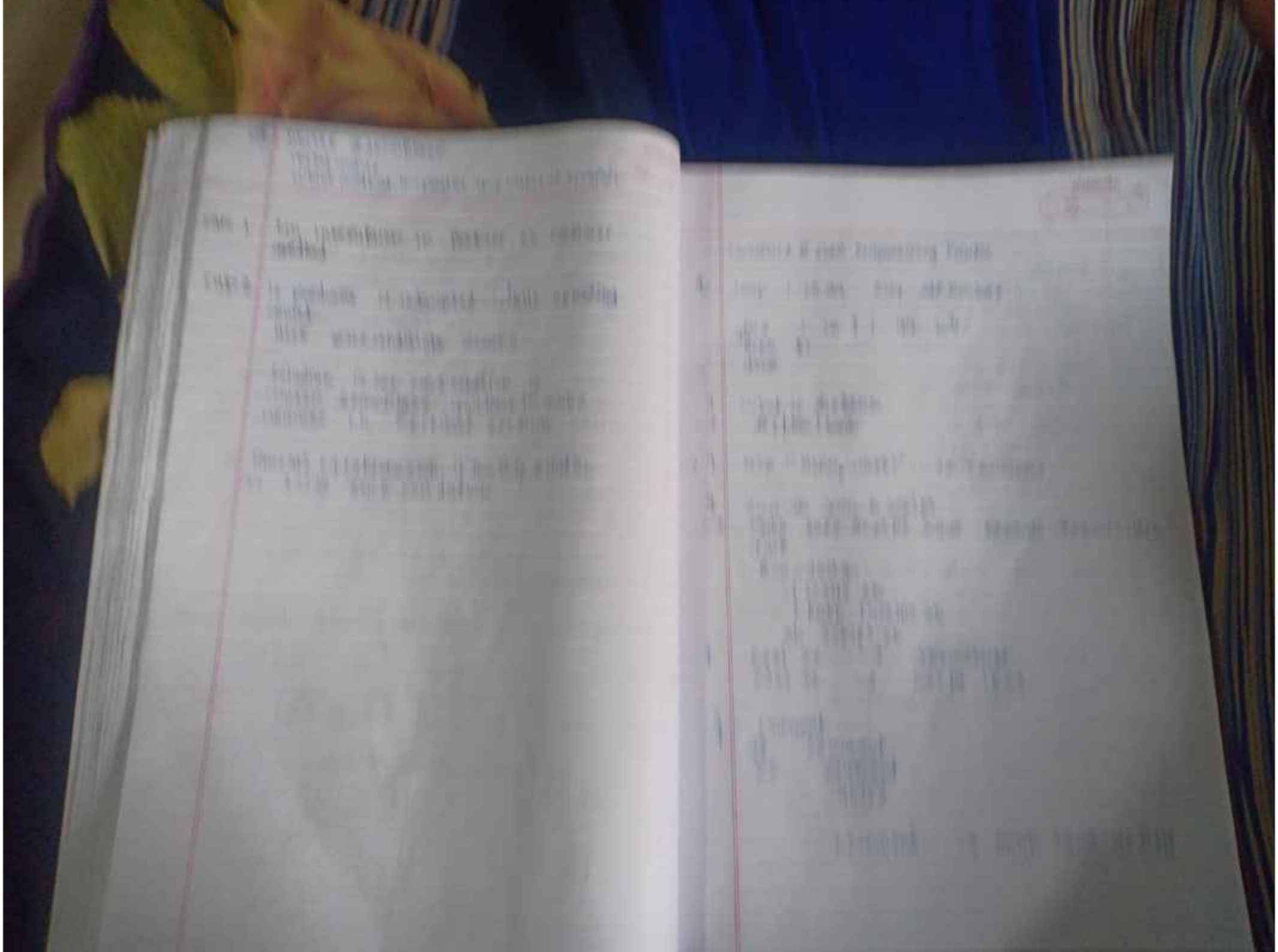
→ Q echo "Hello, word!" TO Terminal

Q How to run a script
→ Make sure script have execute permission.
rwx
- Run using.
./script.sh
/Path/Script.sh
sh script.sh

Q Ctrl+c → Terminate
Ctrl+z → Stops ipos

Q comment.
→ # comment
<< comment
.....
youre.

comment. → same name as first



q variable.

→ VARNAME = value.

var_name = $(hostnam)

q → Print → echo $varname.

q constant variable

→ readonly var_name = "Hi"

q Take user input.

→ read <var_name>

→ echo "what your name"

q read name.

or

read -p "what is your age" age (variable)

echo "name is $name & age is $age"

q if else.

```
if [$age -eq 18]
Then
    echo "your vote now"
else
    echo "NO"
fi
```

q Elif

read -p "Enter your country" country

→

```
if [ $country = "India" ]
Then
    echo "you are indian"
elif [ $country = "Nepal" ]
Then
    echo "you are Nepali"
else
    echo "you are from Earth"
fi
```

q operator.

→ Equal = -eq |==

great Equto = -ge

less Than eq = -le

not eq = -ne |!=

greater than = -gt

less Than. = -lt

q Check some file exits or not.

if [-d foldername] If folder exits.

[! -d foldername] If folder not exits

if [-f filename] if file exists

if [! -f filename] if file not exists

*) default case.

Switch.

```
echo " choose option"
echo " a = To psee The corrend date"
echo "b = listall The files in corrdr"

read choice
case $choice
        a) date ;;
        b) ls ;;
        *) echo "No valid input"
esac
```

* loop

(I)
```
for i in 1 2 3 4 5
do
    echo "Number is $i"
done
```

(II)
```
for j in Raju Sham Baburao.
do
    echo " name is $j"
done
```

(III)
```
for p in {1..20}
do
    echo $p
done
```

* iterate values from file

→
```
items = " /home/Paul/file.txt

for item in $(cat $items)
do
    echo $item
done.
```

*. while
```
count = 0
num = 10

while [ $count -le $num ]
do
    echo " Number at $count"
    let count ++
done.
```

*. a=to until → goes till it's condition is false
execute
```
a = 10
until [ $a -eq 1 ]
do
    echo $a
    a=`expr $a -1`        no yapa
done
```

* infinite loop.
```
while true
do
    echo "Hi"
    sleep 2s,
done
```
=> break - to stop the loop

=> continue - To stop current iteration of loop & start next

*. Print sum of first n so natural number.

```
Sum = 0

for i in {1..20}
do
    Sum = $Sum
        Sum = $(expr $i + $Sum)
done.
echo $Sum.
```

*. fibonache.

n = Total number you want to go.
a, b first two num.

```
for ((          ))
do
 echo " $a"

 fib = $((a+b))
  a = $b
  b = $fib
done.
```

code

```
n = 10
a = 0
b = 1
for ((  a+b  i=1 ; i<n ; i++))
do
    echo "$a"
    fib = $((a+b))
    a=b
    b=fib done.
```

```
read a b
if [ $a -gt $b ]
Then
    if [ $a -gt $c ]
    then
     echo " $a is greater"
    else
     echo " c is greater "
    fi
else
    if [ $b -gt $c ]
    Then
     echo " b is greater"
     else
         " c is greater
     fi
fi
```

*. Compair two string using the Test command
Take input in command line

```
#!/bin/bash

if test "$1" = "$"
Then
echo " string a is equal bigger"
else
echo " string b is not equal bigger"
fi
```

Run command line bash file.sh Str1 Str2

* Print Star Pattern.

```
*
**
***
****
```

```
row=4
for (( i=1 ; i<=row ; i++))      # no space.
do
    for (( j=1 ; j<=i ; j++))    # no space no $.
    do
    echo -n "*"
    done.
    echo ""
done.
```

* File start with letter f1
  is ff

* Find Previous command run or not
→ echo $?
  if output 0 Then run.
  if non zero not run. (between 1-255)

* Check integer is equl or not.
① Test 12 -eq 12 → exit 0
② echo $? → 0 True

* String is empty or not.

```
String=""

if ["$string"=""]
Then
    echo "string is empty"
else echo "not empty"
fi
```

file of particular name present or not
→ Test -e filename && echo "file.txt Present" || echo not exits.

for directory.
Test -d filedirectoryname && echo "file.txt present" || echo "not found"

* multiplication Table

```
Table → 5.
for i in rang {1..10}
do
    result=$((a * i))
    echo "$result"
done
```

* leap year.
  condion. n-number year %. 4=0 | year %.400=0
  year %.100 !=0

```
#! /bin/bash.
year=2024
a=$((year %. 4))
b=$((year %. 100))
c=$((year %. 400))

If [[ a -eq 0 ]] && [[ b -ne 0 ]] || [[ c -eq 0 ]]
Then
    echo "$year is leapyear"
else
    echo "year is not a leap year
```

* factorial.

```
read n
facto=1

for ((i=1 ; i<=n ; i++))
do
    fact=$((fact*i))
done
 echo " $fact "
```

* Sum of n even number

```
#!/bin/bash
read n sum=0
for ((i=1;i<=n,i++)
do
        even=$((2*i))
        sum=$((sum+even))
done
 echo "$sum"
```