27/8/24

Page No.
Date

★ what is OS ?

- memory - It is place where executi-on of programs take place.
- Storage - It is the place where all the data is stored.

- It is a resource manager.
- It is a process mandger. - to supervise all task which is being executed by processor.
- It can take i/p from user and operate that i/p.

- OS is the interface between hardw-are and software.

★ How it is different from other application software?

- It is a hardware manager, no other app" is hardware manager.
- It has to store, It has to execute and give o/p on the hardware only.
- Hardware manager : It manages all the hardware resources or components of computer.
- OS is installed over harddrive.
- Applications are also installed over harddrive but under the layer of OS.

Page No.
Date

* when the OS was not there in 1990's, how the programs were executed / how the interface betn h/w & s/w used to happen?
- Batch OS was used at that time.
- There were no pre-installed program to perform a task.
we used to give i/p for program for it.
Eg. If we want to execute a Java program, then you have to write JVM also.
- Punching cards system was used. i.e. binary system to get the output.
Eg: After writing JVM, it was put on to the punching cards, in the binary form, which used to take more time (1 hr) to give the o/p.
Program written for punch cards ~~was in bina~~ should be written in binary.

- with the OS, efficiency (Through-put) is improved.
The system becomes multitasking with the help of OS.

- The ultimate thing to convert i/p to o/p, display the o/p was all done by the hardware.
- OS is always a hardware dependent, because, OS is a s/w

which needs to be installed.

- The output of the i/p can be stored on registers. also therefore, OS is hardware dependent.

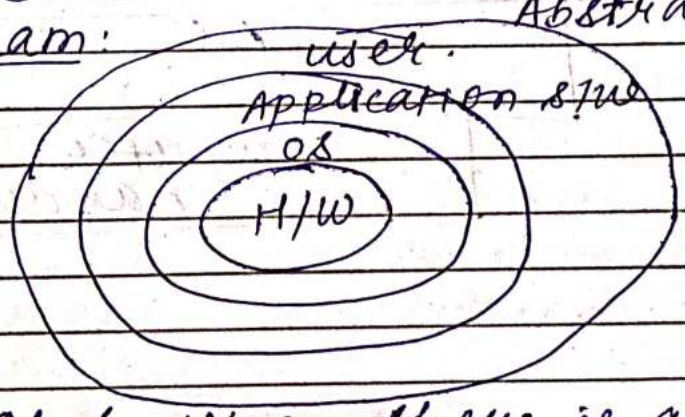
(Advantage).

- Cache is a very fast temporary memory.
- Registers are used to store the intermediate results (which is just required after a instn, it can store data & o/p also). which are very fast.
- Cache is a physical memory, it is not generated at runtime.

- SRAM and DRAM have the architectural difference which makes difference in their speed.

- o/p is displayed on the hardware i.e. desktop, printer, etc.

- OS is the first thing which gets loaded into the memory.
- Diagram:



Abstraction layer: Hiding internal information and just highlighted the setoff services

- Betn OS & H/W, there is a kernel which aligns instn of OS to H/W.

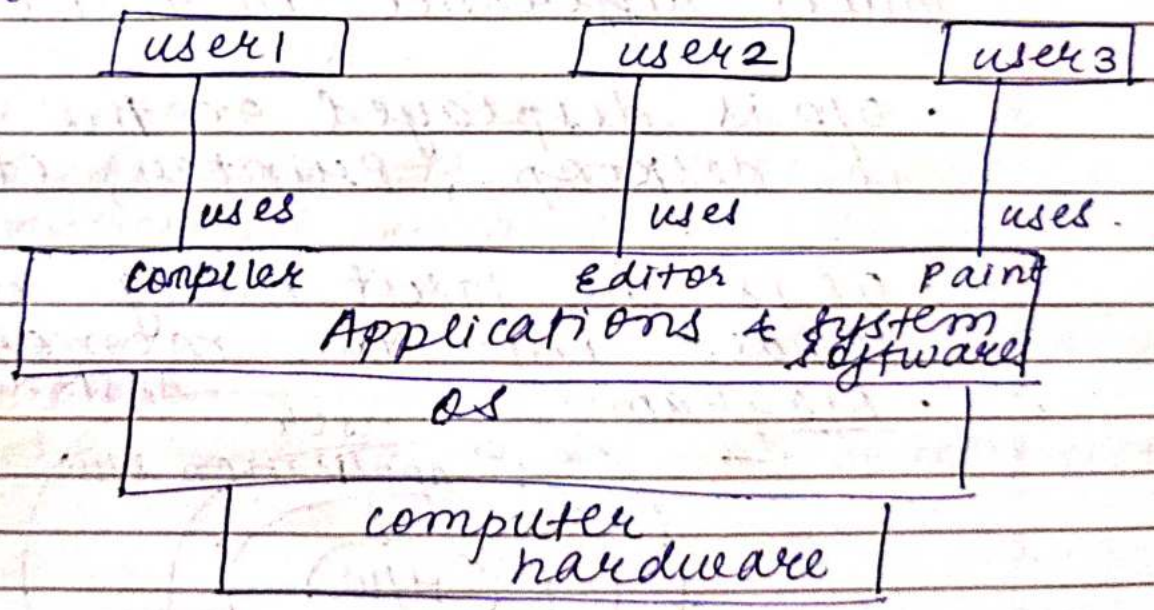
- we do not need to take help of OS to execute the Java program. we can directly execute with the help of kernel.
  JVM is used for executing the program.

- kernel works for OS. kernel collabo-rates with the OS.
  Instructions are given by the OS to the kernel.

- For using the Internet, we requi-re the internet driver.
  For -||- the devices, we require the device driver.

them have computer organization & content starts.

Diagram:



Page No.
Date

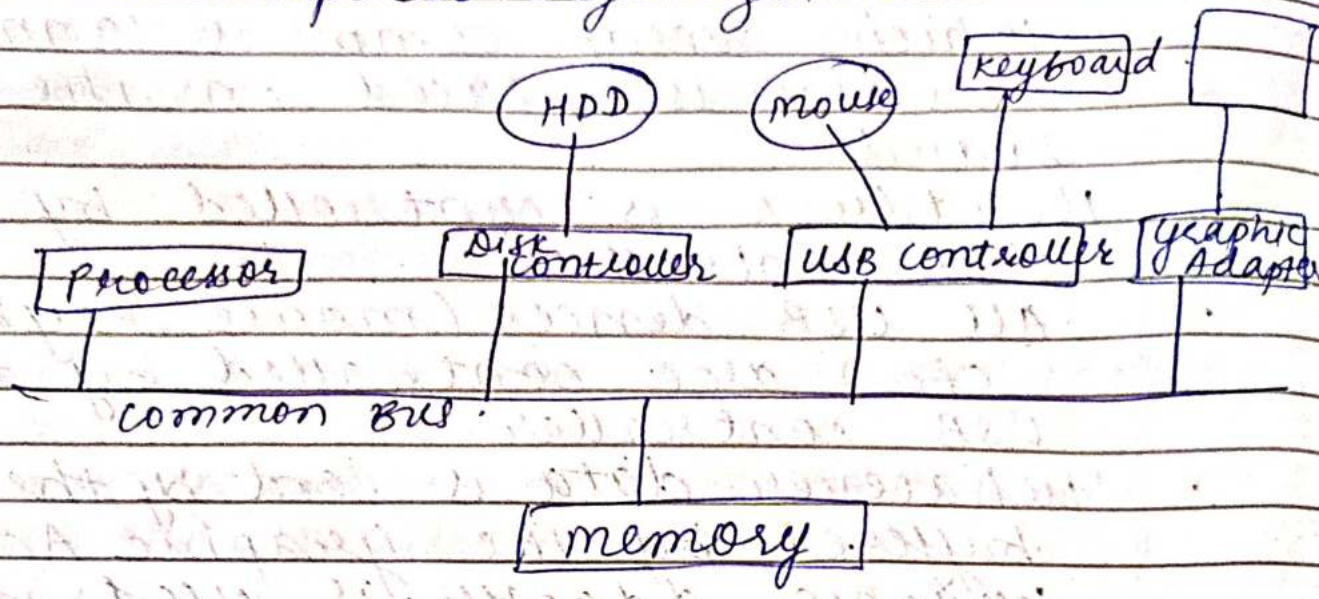
- There is a common bus through which every compⁿ is connected. This bus is placed on the Mother board.
- Harddrive is controlled by the disk controller.
- All USB devices (mouse, keyboard, etc) are controlled by the USB controller.
- whatever data is sent by the CPU, is buffered by the graphic Adapter. graphic adapter is used for buffering i/p and for managing the data ~~als~~ color (picture color) also.
- Processor instructs Disk controller to perform the instructions. Disk controller ~~is~~ ~~ts~~ knows which application is where because of the appⁿ stored on Harddrive.

- Reset bit in the Processor is used to delete all the trash data.
- from Sector 0, ~~on the BIOS in the ROM~~ we get the address of the BIOS to load it.
- BIOS is used to used boot loader for loading the particular OS.
- BIOS was installed on ROM ~~in~~ before 2000s.
- Instructions of OS (Program of OS) are stored on the sector 0.

booting related content.

- Bios is the process before booting.
- Bios checks all the devices.

Basic Computer organization:



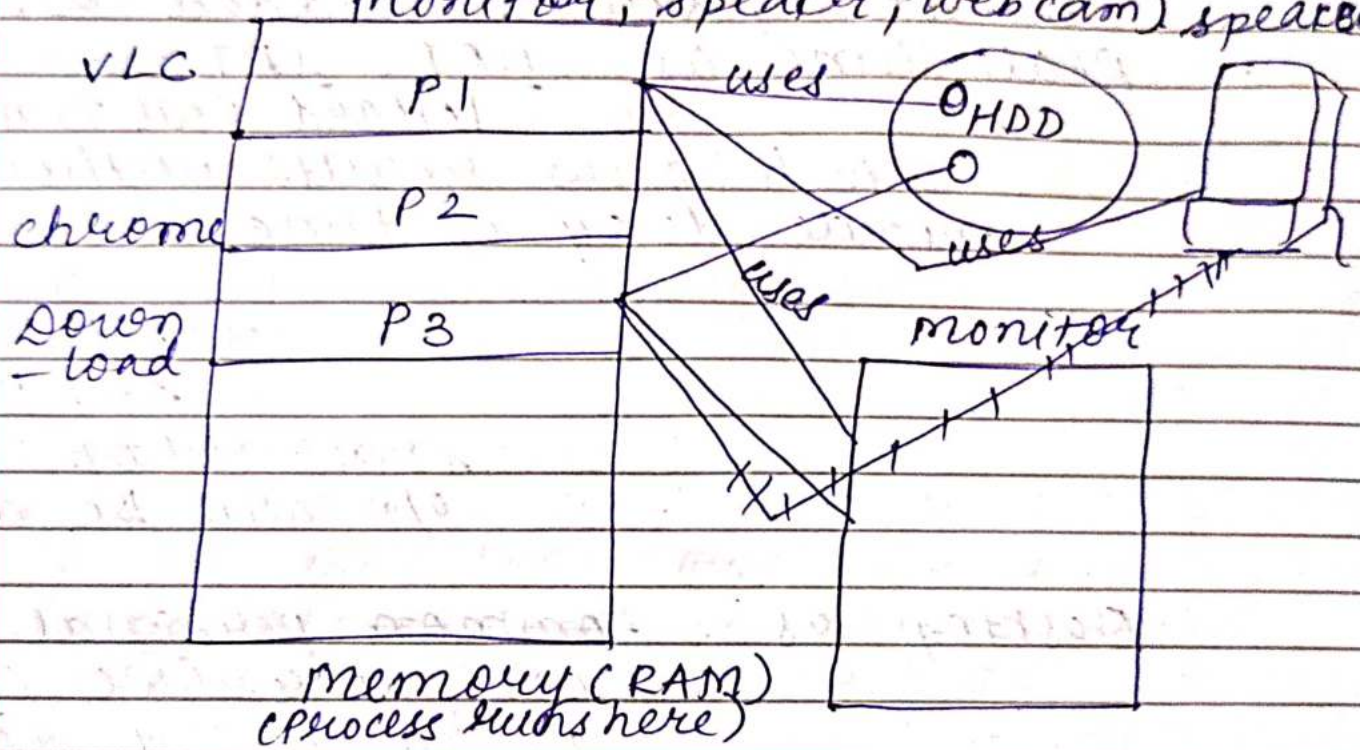
- By writing shell program, we directly connect with kernel i.e. we can directly instruct the kernel.
- Booting is loading all instructions from Hard drive to main memory.
- Cold booting: It involves starting a computer from a totally powered-off state. (Eg: when we start the com
- Hot booting: Warm booting refers to restarting the computer without fully powering it off (Eg. Restarting)

Page No.
Date

* Examples of well-known OS :
- Mobile OS : ~~Androot~~ Android, iOS, windows.
- Embedded OS : In this you can hardly install any application.
  Eg : Nokia phone, Router, washing machine, T.V. ~~&~~.
  OS. is punched on the H/w first only.
  we can upgrade (update) mobile OS but cannot update embedded OS.
- Real Time OS : HRT, SRT.
  (Hard Real Time)
  which gives results without minute delay of time.
  ↓ (soft Real time system).
  some delay in the o/p will be accepted
- Desktop OS : ~~Common~~ personal computer. (we can design ~~these~~ it according to our needs).
- Server machine OS : It is ~~a~~ a special to use or runs over the server machines to responds to millions of request at the same time.
  It has different h/w and s/w also.
  Distributed OS is used for load balancing to handle multiple OS. (having multiple distributed server).

* Functions of OS:

- Process management - e.g. zoom, we can use different features of the zoom at the same time.
- Memory management - All the data is stored / Executed in memory (RAM). Memory used by different applications at the same time.
- Device management - (HDD, Printer, Monitor, speaker, web cam) speaker



when processor shifts from one process to another ~~qui~~ quickly, it is known as ~~At~~ context switching.

- Disk management - Disk scheduling Algorithms.
- Network management - Ports, Network card / controller.

→ Port is used to connect external sources to the motherboard. (h/w port)

→ When we open different tabs, they run on different ports. (s/w port)

→ Interrupt is a quick signal to the processor to listen to the user. (i.e. another process requested by the user).

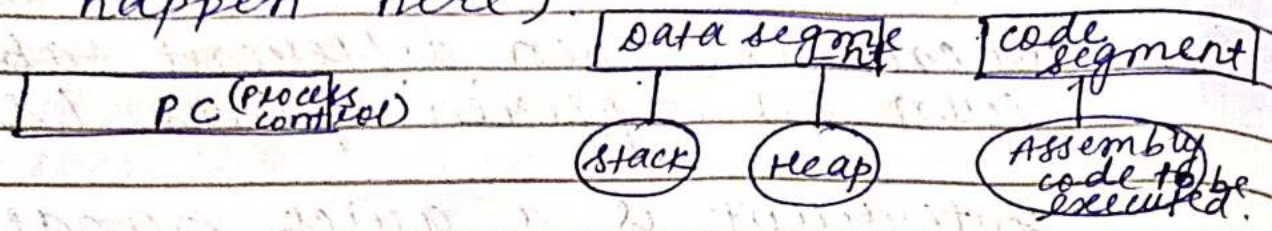
- File management
- Security management : Firewall, Antivirus, Antispyware, Ransomeware (used in banks).

↳ Checksum is the Integrity check for validating it, for finding out whether it is safe or not or it has any virus.
Virus changes the bits & bytes of the data. So Antivirus again corrects it.
Virus changes the bits & bytes because data is stored in bits & bytes.

* User space & the kernel space:

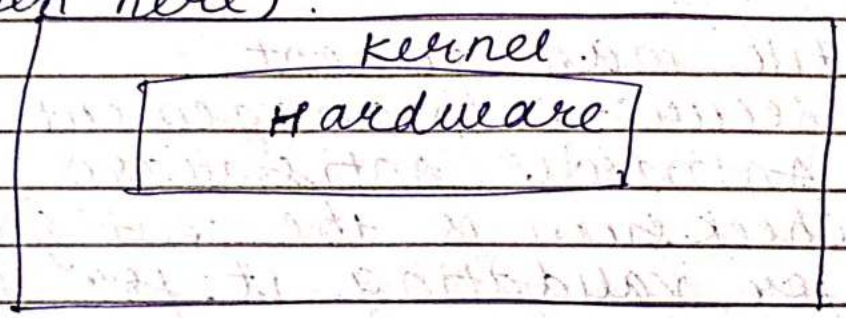
* Heap is a logical data segment).
* PC (program counter points to the next instruction)
* IR (instruction register contains instructions).

User space.
(All the process related things happen here).

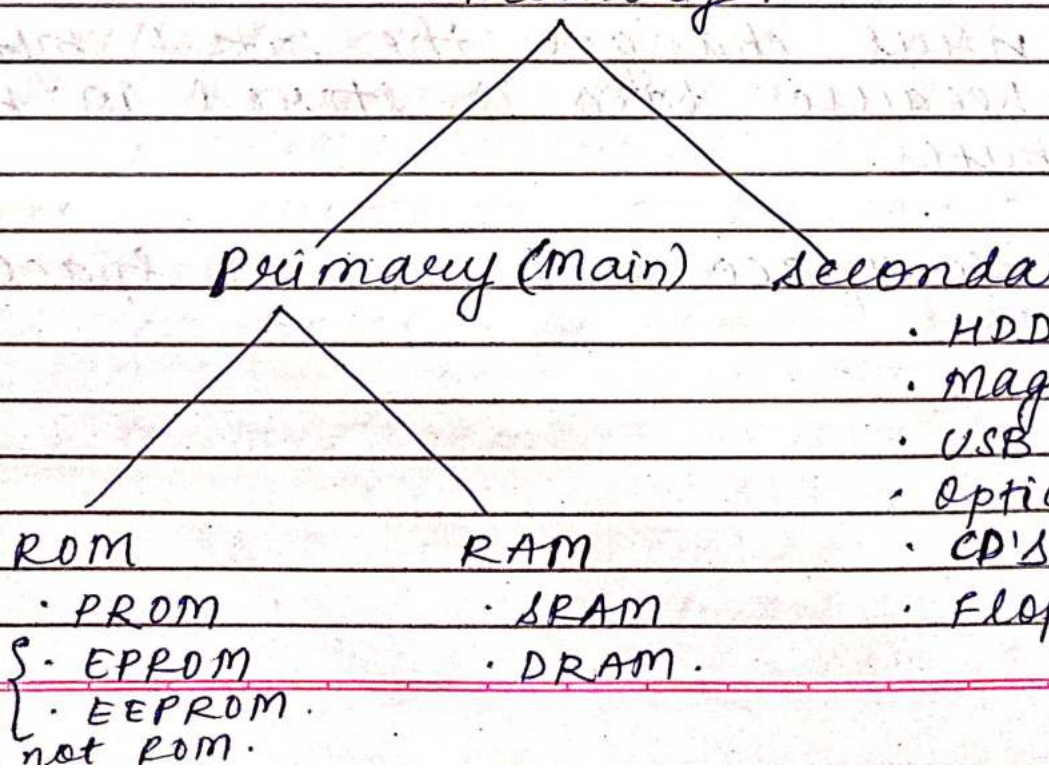


---

kernel space (All OS related activities happen here).



* <u>Memory Hierarchy</u>:

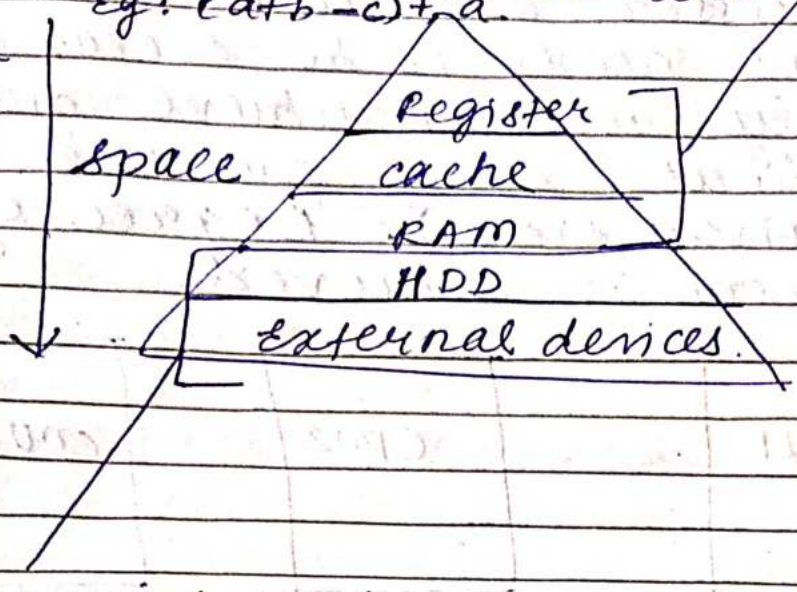


our system has one of these not ROM.

* General purpose register - used to store data & perform operations.
* Accumulator = Intermediate results are stored here.
Eg: (a+b-c)+a.

(Temporary results are stored in this)

volatile memory (Data gets erased when the system is turned off)



Non-volatile memory.
(Data will get erased until you explicitely erase it).

- BIOS lives on EPROM or EEPROM

* Types of OS:

(1) Batch OS: Batch OS is a type of operating system that allows multiple users to use it at the same time. It does not allow any direct communication between users, meaning that each user must complete their tasks before passing control on to the next user.

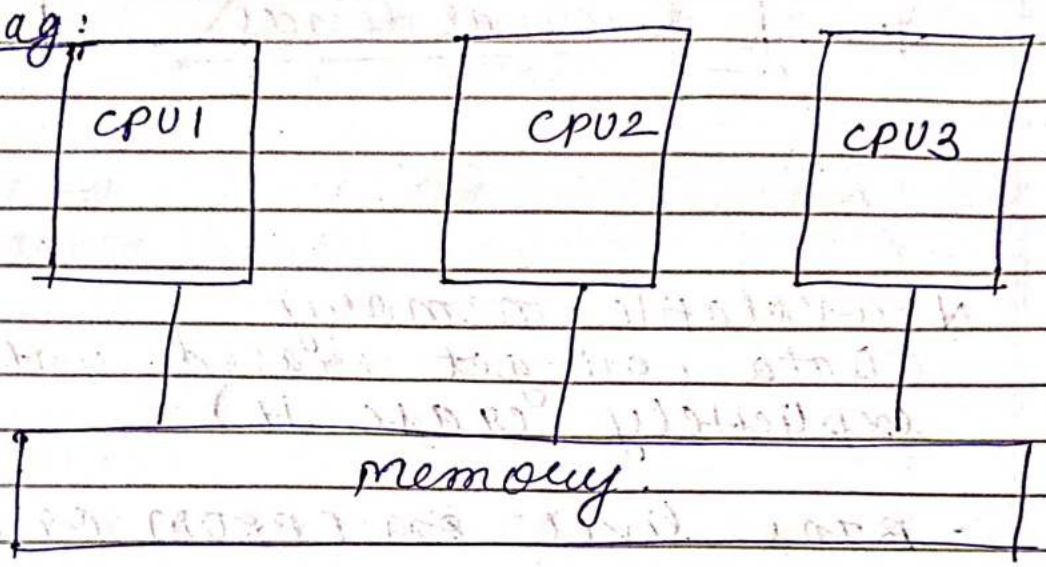
(2). Multiprogramming OS: OS which can handle more the 1 task by context switching them is known as multiprogramming.
Eg. we using zoom and Adobe Acrobat at the same time but ~~concurrently~~ simultaneously.

- If RAM memory is full for swapping, it uses virtual memory.

(3). Multiprocessor OS : It is used when processes needs to be executed concurrently. we can achieve maximum throughput (no. of processes executed in a given time). Degree of parallelism is achieved.

diag:

(MCQ can be asked on these)



- It can have shared or separate memory.

(4). Distributed OS : In distributed OS, multiple CPUs are utilized, but for the end users, it appears as a typical centralized OS. It enables the sharing of various resources such as CPUs, disks, network interfaces, nodes, and computer ~~nodes~~ across different sites, thereby expanding the available data within the entire system.

(5). Desktop OS :

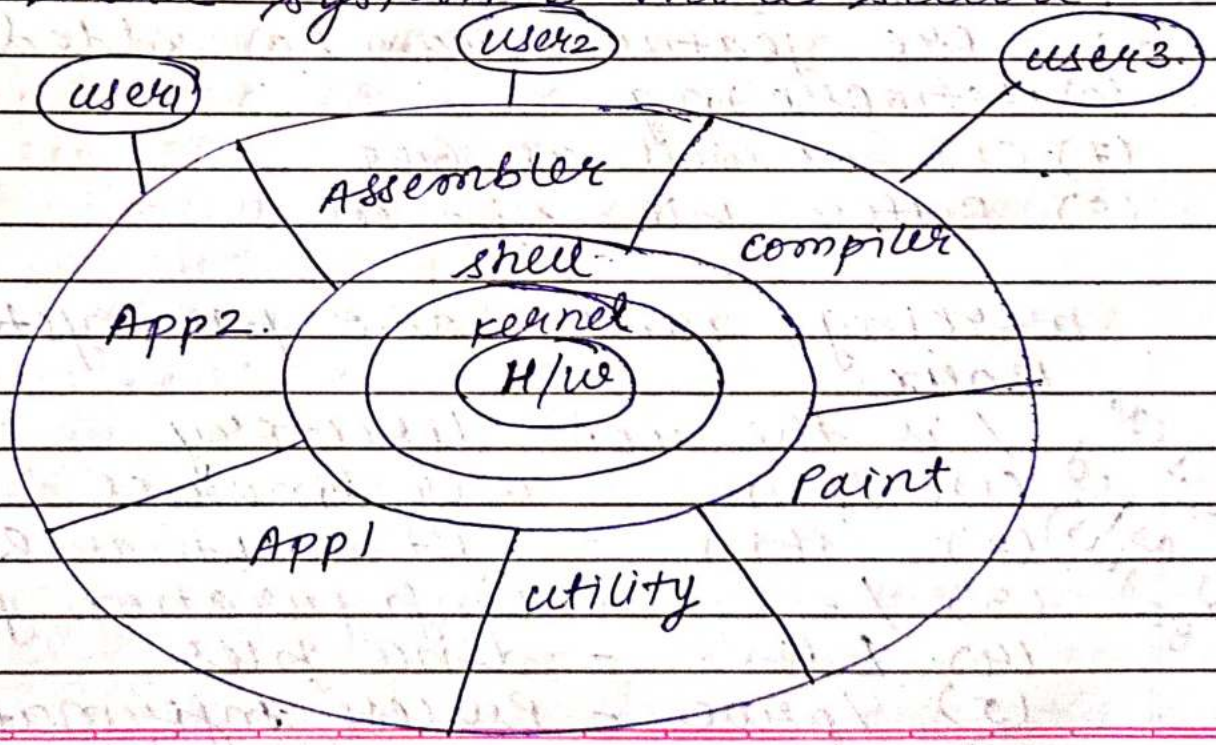
The desktop OS is the environment where the user controls a personal computer (Desktop, Notebook PC). It aids in the management of computer

hardware and software resources. It supports fundamental features such as fast scheduling, peripheral control, printing, i/p o/p and memory ~~location~~ allocation. This OS serves as a bridge betn programs & comp^n h/w.

(6) Server OS:

It is an ~~OS~~ operating system designed for usage on servers. It is utilized to give services to a large number of clients. ~~It~~ It is a very advanced operating system that can serve several clients simultaneously. It is more advanced operating system with features and capabilities needed in a client server architecture or comparable enterprise computing environment.

- Every OS has unix. Kernel is a unix system.
- Linux system is more secure



Page No.
Date

- In linux system, OS is there but it is not in the interaction.
- Kernel is a program which resides in the main memory from starting to end.
- Linux is an open-source operating system. It is available free to use and user can modify it according to their needs.

The founder of linux is Linus Torvards. Its available since 1991. An open source community is working ~~tow~~ behind the updation and upgradation of the linux code.

- Feature:

(1). No cost / low cost
(2). Multi-tasking
(3). security
(4). Multiuser
(5). Stable and scalable : It can grow, & more features can be added.
(6). Networking
(7). CLI as well as GUI.
(8). Better file system.

* working with basic file system of linux.

(mcqs can be asked on this)

/ is the root directory
(1). /bin - user binaries
(2). /sbin - system binaries
(3). /etc - configuration files
(4). /dev - device files
(5). /proc - Process information
(6). /var - variable files.

* pwd - stands for Print working directory.

(7). /tmp - Temporary files.
(8). /usr - User Programs.
(9). /home - Parent directory of user friendly directory.
(10). /boot - boot loader files.
(11). /opt - Apps.
(12). /lib - System libraries.

* Lab 1

· Commands:

(1). Pwd - shows the present working directory.
(2). nano file.txt - nano is the editor just like notepad. It creates new if it does not exist, if it exists, it opens that file.
(3). ls - It lists all the files and directories of current working directory.
(4). touch filename.txt - It creates new file.
(5). mkdir - It creates new folder.
Folder name is in blue color
File name is in white color.

nano	touch
It creates + displays file.	It only creates file.

(6). ls -l - It lists the total number of files.
(7). chmod - through this command, we can give permissions to the users or revoke (remove) all permissions.

Permissions given to the other users (other group) (user within the system)

- | r - r | - - r | - -

other Internet / OS user. (user from the remote location).

These permissions are for the owner.

Eg: chmod a+wx file2.txt.

giving

It represents all users.

u - represents owner.
g - represents group.
o - represents other.

Eg: chmod a-wx file2.txt.

Removed the permissions.

Eg: chmod u+x file2.txt.
chmod g+r file2.txt.
chmod g-rw file2.txt.
chmod a+wrx file1.txt (valid)

execute all this in the lab.

(8). clear - To clear all contents in the cmd.

(9). rm - To remove a file.

(10). cd - To go in another directory.
cd / - To go in inside directory
cd .. - To go back 1 directory
cd - To go to home directory.

(11). rm -r directory-name - To remove parent directory & its ancestors. (to remove file and recursive directory).

(12). rmdir - To remove a particular directory.

* If there is only 1 directory, cd / will work. If there are more than 1 directory inside directory then cd dir1/dir2

- We cannot create a new directory in the root directory.

(13). cd ../.. - It is used to move multiple levels of parent directories.

(14). cd ../../etc - It is used if we directly want to go in the another directory / folder.

(15). mkdir / dir1 / dir2 - It is used to create directory inside a directory.

(16). mkdir 'dir1 1' - To create directory with spaces.

similar eg:

```
mkdir "folder 1"
mkdir 'folder 2'
mkdir folder\ 3
mkdir "folder 4" "folder 5"
mkdir -p "folder 6"/"folder 7"
```

Here folder6 is the parent folder of folder7.

(17) ls > filename.txt - To create new file in a folder.

(18). cat filename.txt - To view the contents of the file.

(19). echo "This is a test file" - To echo a particular text.

(20). echo "This is file2 with content" > file2.txt - To ~~copy~~ put the content to file2.txt.

- sort fruits.txt | uniq --count - To count the number of uniq lines in file.

Note : If ~~already~~ there is already written content in the file2.txt, it gets replaced with the new content

(21). cat file1.txt file2.txt - To display the contents of both the files (Not executing).

(22). cat f* > combined text - To create new file which is a combination of contents of 2 files.

☆ <u>All ls commands</u> :

(23). ls -l - To list the number of files and folders. ~~in~~

- ls -m : List directory con separated by comma
- ls -Q : display directory con enclosed by quotation marks.
- ls -l : Display files in a long-u format.
- ls -lh : Display file size in human readable formats in KBs.
- ls -g : ~~ell~~ omits group ownership column.
- ls -F : Add a forward slash to the directories.

- < file2.txt tr '[:lower:]' '[:upper:]' > output.txt - To convert lowercase to uppercase and transfer it to another file.
- ls -i : Display inode number of files and directories.
- ls -a : Display all files including hidden files.
- ls *.txt : Displays all files according to the file extension.
- ls -la : Displays all files and directories in long list format
- ls -R : Displays files and directories recursively.
- ls -r : Sort files in reverse.
- ls -X * : sort files alphabetically by file extension.
- ls -n : List UIDs and GIDs.
- ls -ltr : It shows files in long listing format in reverse sorted by the modification time.
- ls -tl : It sorts the files from newest to oldest
- man ls : It gives all the details of the cmd i.e. name, synopsis, descriptn
- alias ls = "ls -l" : To make alias for the cmd to perform operatn of another cmd. Here, ls will perform operatn of ls -l.
- unalias ls : To remove aliases.

28/8/24. · grep - It is used to search a particular word in file or string.

★ commands :

(1). sudo adduser user2 - To add new user.

sudoers file all the usernames & permissions given to them.

drwxr-xr-x 2 market market 4096.

owner groupname.

· you can create as much as users and as much as groups.

(2). ~~chown~~ sudo chown user1 file2.txt.

- To change ownership of the file.

su - switch user.

· we need to give user permissions to list the file.
cmd - sudo ls.

· If we want to permissions to the outside user, then, use,
· when user tries to access another directory then use these commands,

(1). nano sudoers.
(2). sudo nano sudoers.
(3). cd.
(4). su user2
- Enter password.
(5). sudo - ls. (lists all the files)

Eg: user2 trying to access user1 files.

* Inside etc folder, sudoers file is there.

Eg: user2 @ CMKL - market : /home/market

user — Directory.

(3) rm - file1.txt - To remove a file.

(4) head -5 file1.txt - To display first 5 lines.

(5) tail -5 file1.txt - To print last 5 lines.

(6) head file1.txt - using only head command prints only first 10 lines by default.

MCQ questn can be asked on this.

(7) tail file1.txt - using only tail cmd prints only last 10 lines by default.

(8) user del username - To delete a user.

(9) Redirection: Output of one cmd is given to the next command is known as redirection.

(10) cat > file3.txt - To add data in the file.

(11) The eg. of Redirection:
head -5 file.txt > file4.txt -
To pass the first 5 lines of file.txt are passed to file4.txt.

> > [illegible] will overwrite the content

eg: till -s file.txt > file2.txt

(12) ~~=~~ - To append the file content to file2 content.

(13) whoami - To get the current user name.

(14) find . -type f -name "*.txt" - To get the full of txt extension.

(15) ~~cp~~ ~~To~~ cp file1.txt file2.txt - To ~~change~~ the file ~~same~~ from ~~one place~~ to ~~file2.txt~~ another directory.

(16) mv file1.txt ~~=~~ file2.txt - To change the filename.

(17) hostname -i - To get the ip address of the system.

(18) zip -r dest.zip ~~Src~~ folder - To zip a folder.

folder which you want to zip.

(19) unzip dest.zip - To unzip a folder.

(20) sed -i 's/This/Thus/g' : To replace a word with another word in file.

(21) ~~unzip dest.zip - 7 sed -n '4, 15p' file.txt~~ - To display first 15 line of the file.

* Every process has its usermode and kernel mode. execution in the user mode is done by processor.

* System calls:

[user mode]

P1 (under execution) mode bit = 1

[system call] mode bit = 1 P1 execution

Trap/Interrupt,

set mode = 0 mode bit = 0.

[kernel mode] [system call execution]

- System call executions happens here.
- when processor works for kernel mode, it works in the kernel mode

- If process P1 want to access any file or hardware, it will make a system call.
- CPU will shift from user mode to kernel mode.
- when CPU works ~~for~~ ~~t~~ in the kernel ~~mod~~ user mode, mode bit is 1 when it shifts to kernel mode, mode bit is set to 0.
- when there is a i/p, o/p call generated by a process, ~~a~~ it sends a Trap/Interrupt is generated.
- Processor works for both user mode and kernel mode.
- If there is a delay in fetching the data, IOQ is generated. IOQ (input output queue).

- If 1 process is accessing one hardware resource and if another process tries to access that, it goes in IOQ.
- After process execution, ~~after~~ i.e. after accessing the h/w resource, mode bit is again set to 1.
- System call modifies the mode bit.
- Interrupt is specially related to h/w. System call is specially related to s/w.

* Types of system calls:

(1). File related calls : Read (), write(), Delete (), open (), close (), create()

(2). Process related calls : New (), fork(), Exit (), wait (), running (), etc.

(3). Device related system calls : read(), write() , ioctl, etc.

(4). Information related calls : getpid(), gettime (), sysdata , etc.

(5). communication related : wait(), signal (), status, etc.

- Eg. In java, ~~w~~ when we execute a file, source file is given ~~to a~~ as a process but processor will use executable file by generating it to give the output.

- Process : It is the information or code or data which help the processor to execute or complete the task.

It consists of :

(1). code segment : It consists of compiled code or instructions to be executed.

(2). Data segment : It consists of data required for the execution.

(3). Information segment : It have meta data about the system variables or system which help in the process execution.

(4). Memory segment : Heap & stack.

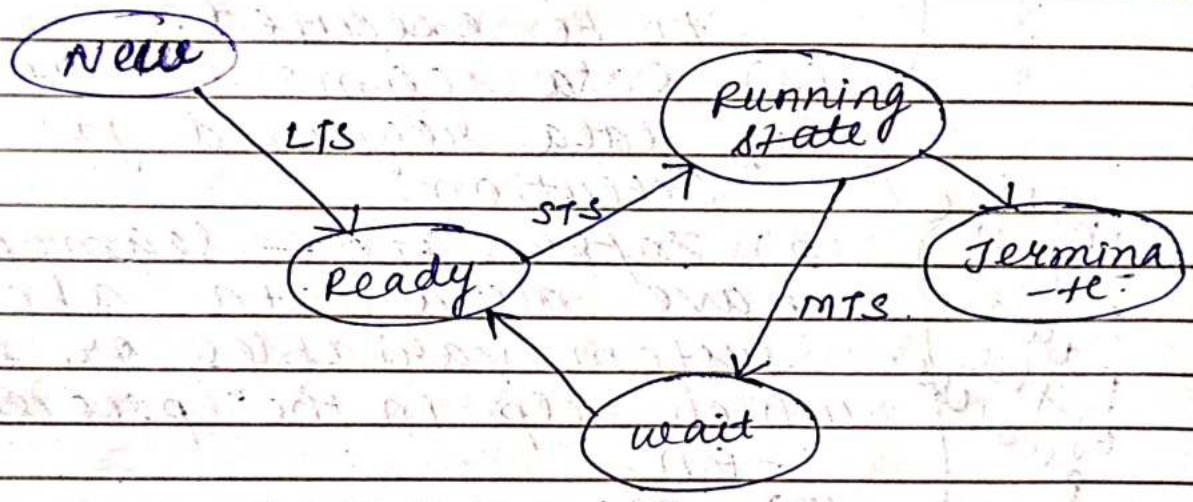
These are the parts of the process.

- Information of the process is stored in PCB. Every process has its own PCB.

* CPU is more faster than i/o devices. that's why ~~so~~ it directly puts Process in waiting queue instead of telling them to wait.

* <u>Process life cycle</u>:

(1). New state :
(2). Ready state (Ready queue)
(3). Running state : Processor is executing the process.
(4). Terminated state :
(5). wait state :



· schedulers are special programs which do the work of scheduling or putting process in different states.

* STS (Short Term scheduler):
It does the work of deciding which process to execute next.

- Process schedulers:

(1) Long Term scheduler (LTS):
when we know Process will need to wait for more time the LTS is used

(2) Short Term scheduler (STS):
when we move a process from Ready to Running state and when CPU needs to be given to a process for short time STS is used.

(3). Medium Time scheduler (MTS):
when a Process is moved from Running to wait state. or when process needs to wait for some time then MTS is used.

- when a process waits for long time in the waiting queue & does not get the CPU, it leads to starvation.

- scheduling Algorithms:

(1). Pre-Emptive scheduling:
If 1 process is under execution and if CPU takes another higher priority process for execution then it is called Pre-Emptive scheduling.

(2). Non-Pre-Emptive scheduling:
If 1 process is under execution and if any process with lower priority & higher priority cannot interrupt the process 1, then it is called Non pre-emptive scheduling

★ PCB (Process Control block): It is a special block which contains metadata about the process.

PID	1001
Pointer	0xfff0
State	Ready
Allocated registers	A, C, D
PC (Program counter)	0xdff0
Allocated Hardware	KB, MS, PR
Accounting Information	file.txt, abc.txt
Priority	10
	⋮

Pointer points to the code to be executed.

PCB P1

· Attributes of PCB:

(1). Process state: The state of the process is stored in the PCB which helps to manage the processes and schedule them. There are different states of the process which are 'running', 'waiting', 'ready' or 'terminated'.

(2). Process Id : The OS assigns a unique identifier to every process as soon as it is created which is known as Process ID, this helps to distinguish between processes.

(3). Program counter : while running processes when context switch occurs the last instruction to be executed is stored in the program counter which helps in resuming the execution of the process from where it left off.

(4). Registers : Registers are used to store data created during the execution of a task.

(5). Memory limits : This field contains information about memory management system used by OS. This includes page tables, segment tables, etc.

(6). List of open files : This information includes the list of files opened for a process.

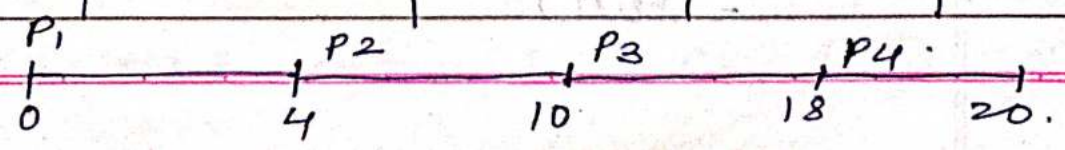
☆ Non-Pre-Emptive scheduling :

Types:

(1). FCFS (First come First come) : The process which comes first is given the CPU first.

(Questions will be asked in this exam)

PID	Arrival Time	Burst Time	wait	TAT.
P1	0	4	0	4
P2	1	6	3	9
P3	2	8	8	16
P4	3	2	15	17



waiting time of process =
CPU Allocation - Arrival time.

Avg waiting time = $\frac{\text{sum WT of all processes}}{\text{No. of processes}}$

$= \frac{26}{4}$ 6.5

$= 6.5$

(2). Shortest Job First (SJF):

PID	Arrival Time	Burst Time	CT	wait	TAT
P1	0	4	4	0	4
P2	1	6	12	5	11
P3	2	8	20	10	18
P4	3	2	6	1	3

Gantt chart :

P1	P4	P2	P3

0 4 6 12 20

(Starting with P1 because there should be a process for comparison).

(3) Round Robin Scheduling :

This algorithm is better than first come first serve.

Starvation issue will not be here.

In round robin algorithm, a fixed slice of time is given to each and every process.

PID	Arrival Time	Burst Time	CT	Wait Time	TAT
P1	0	2	2	0	2
P2	0	4	10	6	10
P3	0	6	14	8	14
P4	0	2	8	6	8.

Gantt chart :

P1	P2	P3	P4	P2	P3	P3

0 2 4 6 8 10 12 14

Avg. WT = $\frac{20}{4} = 5$ Avg TAT = $\frac{48.5}{4}$

* cp + mv imp^n commands - practice

(2) Pre-Emptive

(a). Priority scheduling :

PID	Arrival time	Burst time	Priority	CT	Wait	TAT
P1	0	2	7	12	10	12
P2	0	4	5	10	6	10
P3	0	6	1	6	0	6
P4	0	2	9	14	12	14

Gantt chart :

P3	P2	P1	P4

0 6 10 12 14

Avg WT = $\frac{\cancel{28}\ 7}{\cancel{4}} = \underline{\underline{7}}$

Avg TA = $\frac{\cancel{40}\ \cancel{42}\ 10.5}{\cancel{4}} = \underline{\underline{10.5}}$

16
12
28
28
14
40

(b).

PID	Arrival time	Burst time	Priority	CT	Wait	TAT.
P1	0	2	7	12	10	12
P2	1	4	5	11	6	10
P3	2	6	1	8	0	6
P4	2	2	9	14	10	12

Gantt chart :

P1	P2	P3	P2	P1	P4

0 1 2 8 11 12 14

Avg WT = $\frac{\cancel{26}\ 6.5}{\cancel{4}} = \underline{\underline{6.5}}$

Avg TAT = $\frac{\cancel{38}\ \cancel{9.5}}{\cancel{4}\ 1}\ \frac{\cancel{40}\ 10}{\cancel{4}\ 1} = \underline{\underline{10}}$

28
12
40

* Efficiency of the ~~Algorithm~~ is decided based on how many processes it completes in 1 min.

* Memory Management (RAM):-

- The memory of the computer systems can be divided into block of ~~fix~~ fixed size or variable size. This process is known as fixed size partitioning and variable size partitioning respectively.

Fixed size partitioning: Here, the memory is divided into fixed size blocks where all the blocks are of size like either 2 bytes, 4 bytes, etc. The process which will get the memory may be small or equal to the partitioning block size. As the process size may be smaller than block, it gives rise to internal fragmentation.

- There will be no internal fragmentation in variable size partitioning.

Variable size partitioning: Here the memory is divided into variable size blocks ~~in~~ in which blocks may have size 2 bytes, 3 bytes or any size.

In variable size partitioning, during contiguous memory allocation some time total memory available cannot be given to the new process which leads to external fragmentation.

Page No.
Date

* Compaction :
- Compaction is a technique to collect all the free memory present in the form of fragments into one large chunk of free memory, which can be used to run other processes.
- It is time consuming, waste of CPU cycle and execution.
- It can be done only when the relocation is dynamic & done at execution time. Compaction cannot be done when relocation is static and is performed at load time or assembly time.

Before Compaction:

* ~~First fit, Best fit, and Worst fit~~ =
Before compaction, the main memory has some free space between occupied spaces. This condition is known as external fragmentation. Due to less free space betn occupied spaces, large processes cannot be loaded into them.

occupied space
free space
occupied space
occupied space
Free space.

After Compaction:
After compaction, all the occupied space has been moved up and the free space at the bottom. This makes the space contiguous and removes external fragmentation.

* Main memory means is allocated with the processor, so moving instructions into and out of the processor is extremely fast. Main memory is also known as RAM (Random Access memory). This memory is volatile. RAM loses its data when a power interruption occurs.

Compaction continued...
Processes with large memory requirements can be now loaded into the main memory.

occupied space
occupied space
occupied space
Free space
Free space

* First fit, Best fit and Worst fit:
First fit: First algorithm searches for the first free partition that is large enough to accomodate the process. The OS starts searching from the begining of the memory and allocates the first free partition that is large enough to fit the process.
Best fit: It searches for the smallest free partition that is large enough to accomodate the process.
Worst fit: It searches for the largest free partition and allocates the process to it.

29/8/24.

Page No.
Date

* Process is a representation of user's action which he is doing on the physical ~~RAM~~ memory i.e. RAM. - True

* The OS doesn't want CPU to remain ideal at any time so we implement CPU scheduling through schedulars.

* Scheduling Algorithms take care that CPU gives the maximum Throughput.

* FCFS algorithm can lead to starvation which is known as convey effect.

Round Robin Algorithm:

* ~~SJF algorithm will never~~

* · once the process is completed partially, the CPU temporarily puts process from RAM to Harddrive which is known as swap out. and again bring the process in RAM for execution. which is known as context switching. This is the biggest disadvantage of Round Robin Algorithm.

· There is only 1 solution to starvation i.e. Round Robin Algorithm.

* · Response Time:
Process getting CPU for execution is known as Response Time.
In Round Robin scheduling, when a process is given some time, every now & then, only first response time is considered.

- Completion Time : Completion time is ~~to~~ when a process finishes execution and is no longer being processed by the CPU. It is the summation of the arrival, waiting and burst times.
- Waiting Time : This is a process duration in the ready queue before it begins executing.
  WT = TAT − Burst time.

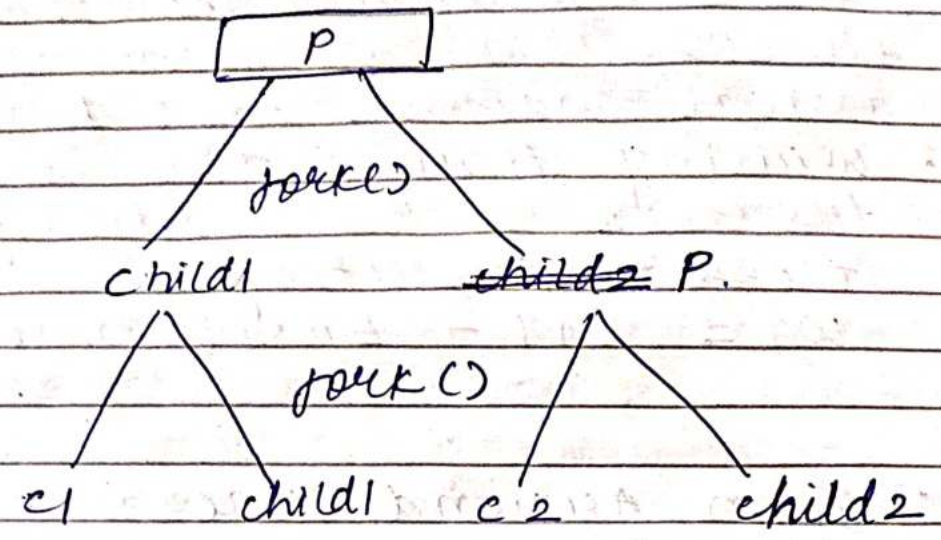
- Turn Around Time :
  Formula:
  ~~for~~ TAT = Completion Time − Arrival Time.
  The time elapsed between the arrival of a process and its completion.

* · Process creation using fork :
  fork() : It is method / system call which is used to create child process of the current process being executed.

- What a child process does?
  It does exactly the same for which you have designed the Parent process.

- We can prioritize a Parent process & the child process by using multithreading.
  If we wish a child process to do a different task than parent, it ~~is~~ ~~to~~

can be achieved using multithreading



No of ~~of~~ time, process will be be executed $= 2^n$
$= 4$.

No. of child processes $= 2^n - 1$.
$= 2^2 - 1$
$= 4 - 1$
$= \underline{\underline{3}}$

* ~~+~~ · If child process can't be created, it is represented by -1.

· If child process can be created, it is represented by 0.

· If the parent process can be created, then, it is represented by +1.

* If the parent process is executed, if it is ~~destroyed from~~ in the memory just because its child is getting executed it is known as zombie process.

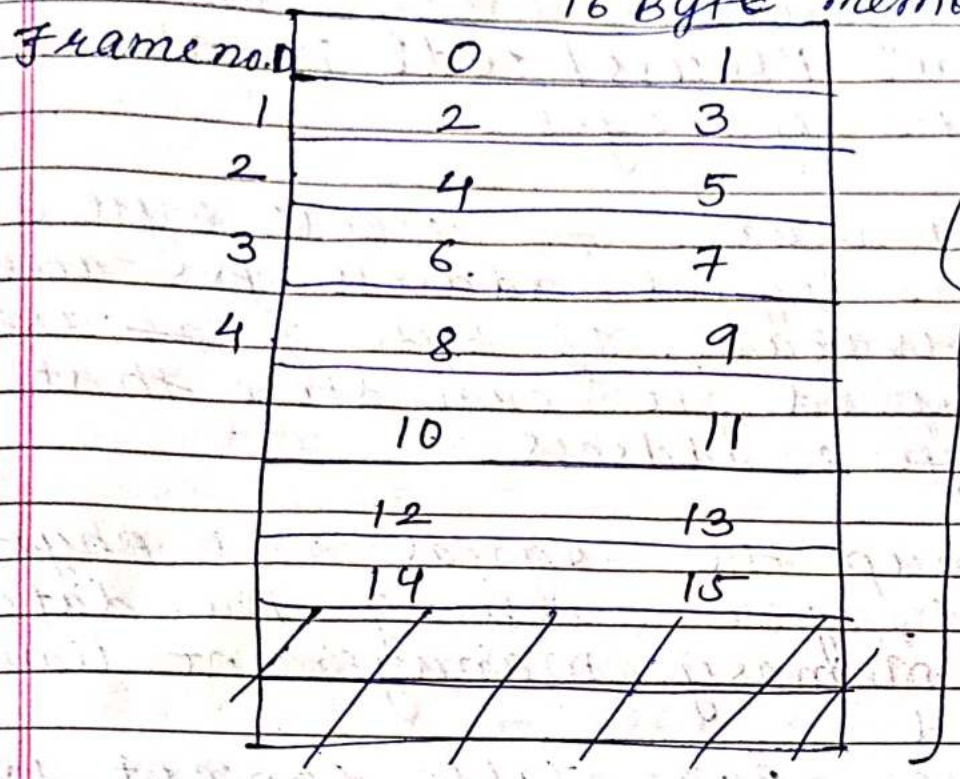
Orphan process: A process whose parent process no more exists. i.e. either finished or terminated without waiting for its child process to terminate is called as orphan process.

* ~~Variable~~ Paging:

- Dividing the process into no. of pages into Hard drive (secondary memory) and calling the pages (by) in main memory (RAM) is called paging. In this, we will load only demanded pages into the RAM.
- Paging will remove external fragmentation.
- Dividing the process, just before entering into the RAM is not logically possible.
- Dividing the process where it is actually i.e in Hard drive is known as paging.
- main memory where the program/process needs to executed is divided into frames.
- To support paging, the size of pages must be equal to the size of the frames.
- size of all the pages is equal. All the pages must be divided into equal size.
- Paging is the solution to internal fragmentation as well as external fragmentation.

* Paging & segmentation are types of virtual memory.

16 Byte memory.



Fixed size Partitioning.

- If the computer memory is designed to have frames of 2 bytes of each, then, process of 4 bytes is divided into 2 - 2 bytes.
- Computer memory is byte-addressable. (Entire memory is divided into bytes)

Process i is 4 B.

Page 0	0	1
Page 1	2	3

~~(0, 1st, 2nd & 3~~

(0th, 1st, 2nd & 3rd byte).

* Address ~~stored~~ in the main memory is physical address + Address in secondary memory is logical add

- The physical memory where we shift our Process/code is also divided int bytes of Pages.

- If CPU wants to access Byte 3, CPU uses logical address for accessing the program. If Byte 3 ~~ao~~ area is occupied, it can store that data at Byte 8 address.

- To map the logical and physical address, for accessing the data, MMU (memory management unit) is used.
- It uses Page Table concept for mapping.

Page Table:

page0	frame1
page1	frame4

* Aging: It is a scheduling technique used to prevent starvation in operating systems. It involves gradually increasing the priority of processes that ~~having~~ been waiting for a long time. It increases the chance of them getting the necessary resources to execute. Thus, it reduces the risk of starvation.

* The OS is stored on the hard disk, or hard drive of a computer. However, to speed up the startup process, the OS is occupied into RAM (Random Access memory) when the computer starts. This is done by ~~to~~ BIOS (Basic Input output system).

* <u>Virtual memory</u>:

- For storing temporary data/process/program, virtual memory is used.
- This virtual memory is present in our hard drive.
- This is volatile memory. If you switch off your PC, the data gets erased from the virtual memory.
- Paging is done in virtual memory.
- while working with virtual memory, pages are saved as per their logical address.

* <u>Page Fault</u>:
- when CPU requests a page and if it is not there in the memory, then it is a miss of a page and it is known as page fault.

* More the number of address bits, the memory can be divided into more partitions.

- while loading the pages from virtual memory to physical memory, the pages get their physical address.

· Buffer is device specific
Eg. Printer will have its own buffer,
Desktop will have its own Buffer.

* <u>Shell scripting:</u>

· when we want to use more than 1 st Linux command (shell command) more than once, it is known as shell scripting.

Eg. of Linux command,
ls ; cal ; date

· It is a very powerful scripting.

· Study different types of shell:

MCQ's can be asked on this

(1). Bourne shell (sh)
(2). C shell (csh)
(3). Korn shell (ksh)
(4). Bourne Again shell (bash)
(5). Z shell (zsh)
(6). Tenex C shell (tcsh)
(7). Debian Almquist shell (dash)
(8). Friendly Interative shell (fish)

30/8/24.

☆ Logical address is a virtual address created by the CPU of the computer system, used as reference to access physical mem loc.

Eg: Page Replacement Algorithm: FCFS (FIFO).

P1 → 0 2 1 2 0 3 5 3 2 1 0.

· If it is not mentioned that if page is loaded or not loaded, then we have to assume that frames are empty in the question.

Frame3			1	1	1	1	1	1	2	2	2
Frame2		2	2	2	2	2	5	5	5	5	0
Frame1	0	0	0	0	0	3	3	3	3	1	1
	M	M	M	H	H	M	M	H	M	M	M

· If the page is not available in the main memory, then we have to load it from the hard drive.
· For replacing the frame, the frame which comes in first, will go out first.

· FIFO and LRU are ~~kind of same~~ little bit same Algorithm.

· Miss Ratio $= \frac{8}{11} \times \cancel{300}\ 27.25$

$= \frac{\cancel{2400}\ \cancel{218.1}}{\cancel{11}} = \underline{218.00\%}$

$\frac{\overset{5}{\cancel{2}}\overset{4}{\cancel{7}}.25 \times 8}{}$

Hit Ratio $= \frac{3}{11} \times 100$

$= \frac{\cancel{300}\ 27.25}{\cancel{11}}$

$= 27.25\%$.

* Eg with 4 frames:

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

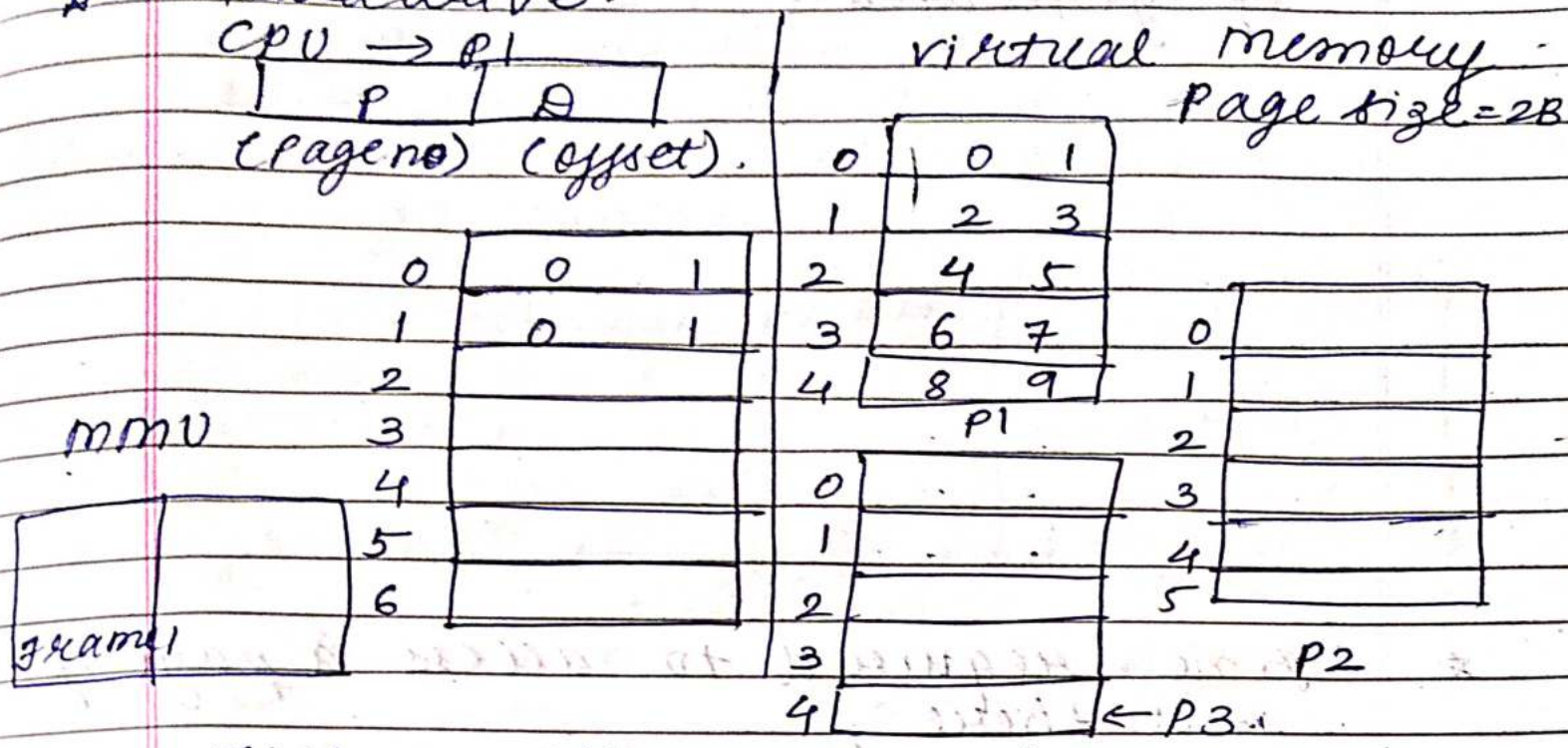
* Balady's Anamaly:

In the first In First out Algorithm, when we increase the no. of frames, the hit ratio is decreased, is stated by balady's Anomaly.

Solution: Don't give large number of frames to main memory.
This is experienced in FIFO (First In First out), second chance algorithm and Random page replacement algorithm

* RAM is always smaller than your Hard drive. therefore, no. of frames cannot be bigger than pages.

* Hard drive:



- It is an illusion to the main memory that it can execute a process which is larger than its memory.
- So, the process is divided into pages in the virtual memory and then put into the main memory as per the requirement.

$$\begin{array}{r} 45 \\ \times 8 \\ \hline 120 \end{array}$$

- CPU: Logical Address:

Pageno.	Page offset
3	1 (2,3), (0,1)

- Page number is equal to the frame number. because, it is just like a RAM, just execution doesn't happen here.

★ · Time required to access a page if it is there,

x – To see if page is there.
x – To read the page.
$= x + x$
$= 2x$.

· If page is not there, time required is x – To ~~to~~ see if page is there or not.

· Page table is present in the RAM also.

· To do the mapping of the pages from pages to frames, Translation Look Aside Buffer is used. It is a cache memory. It is limited. Frequently used pages are kept here.

* swapping out a page means sending it back to its original place and it is updated.

CPU → P1.

P	D

(Page no) (Offset).

LA = 3 1

Page Table.

Frame no	valid/Invalid	Dirty bit		
Frame1	1	1		
Frame7				

Translation Look Aside Buffer.

page-3	Frame1
page 9	Frame7

- CPU can directly access Translation Look Aside Buffer for accessing a page. It will increase the efficiency and throughoutput of the CPU.
- If the page is not found in the TLB then, time required will be more because, first CPU has to look for TLB and then in the Page Table to see if the page is there or not.

* process execution [illegible] than the process loading time.

- During the execution, if the page has been updated then dirty bit is set to 1. or ~~else~~ otherwise 0.
- when a page is swapped from the page table for adding a new page at that place, the removed page is the updated version placed at its original place.

- TLB:
- It is a page Table represented in cache memory to fasten the process of finding physical address for a given logical address.
- In paging, we load only few pages in main memory of a particular process. Entries of that pages is made in MMU page Table which is actually present at physical memory. To find a frame MMU first look into page table and then according page table entry it accesses the given frame no. This process is actually accesses the RAM twice. So to reduce this access as use Translation Look Aside buffer in cache memory.

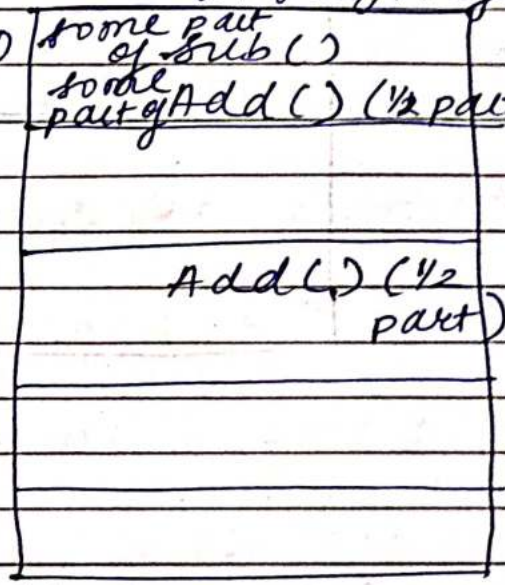
☆ Shared pages:
- If the content of 2 or 3 processes is kept in 1 page then, it is called shared pages.

☆ Invalid ~~bit~~ / Valid bit:

☆ Segmentation:
- To overcome the problem of paging, segmentation is used.

```
void main() page 0
{
Add();
Sub();
Div();
Mul();
}
Add() - Module1
{
  // some code
}
Sub()
{
  // some code (Module 2)
}
Mul()
{
  // some code (Module 3)
}
```

In paging



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

* wildcard symbols:
  (initial of a file1, file2, file3, ---)

(1). ls file*, ls s*

(2). cat file1 | grep ^h.*n$

caret symbol

to indicate which is starting with this (^) and ending with ($).

(3). cat file2 | grep ^h (It is case sensitive).

(4). question mark ?

(5). Asterisk *

(6). square Braces [ ]

(7). curly braces { }

(8). sign of exclamation [!]

(9). Back Slash (/): It acts a separator in the command.

Page No.
Date

* Shell Meta characters:

(1). Pipe (|) - It allows us to sort the particular in our given way.
e.g. cat file2 | grep ^h
It displays the content starting with h (case sensitive).

(2). Redirection (>): It allows you to use output of one command as an input to the other command.

(3) Tilde (~): It is used for going back to the user's home directory. cd ~ and cd both are same.

(4). Dollar symbol ($): It is also used for matching the end of the string for a line in a file.

(5). Caret (^): It is used to match the starting of the string within a given file.

* Command line Arguments:

(1). $# : No. of parameters passed.

(2). $0 : Script Name.

(3). $i : i can be any number, it will return that parameter

(4). $* : give all parameters passed.

Eg : # /bin/bash.
echo Number of param , $#
echo Script Name , $0.
echo Hello, $1
echo Hii , $2
echo OK , $3
echo Bye , $*.

(Test -f file.txt)

(1). Test cmd : To test if particular file is there or not ~~or~~
Eg : test -f abc.txt.
If it is there, it will result with 0. If it is not there then it will return 1.

Eg : #!/bin/bash (For Numbers)
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.

Eg : #!/bin/bash
x = Prashansa
y = Prashansa
if test $x == $y
then
~~eho~~ echo x and y are equal
else
echo x and y are ~~equal~~ not equal

($?) cmd -
/gt cmd -

* In unix / Linux, if the result is true, it returns 0 otherwise it returns 1.

* In other languages ie. c, Java, if the result is true, it returns 1 otherwise it returns 0.

(2) Test -d cdac - It is used to test whether this directory is there or not.

(1). Accept 2 words from cmdline and decide whether they are equal or not (Prg).

(2). Try -eq, -gt, -le, -ge, -lt, -z, these operators with test command.

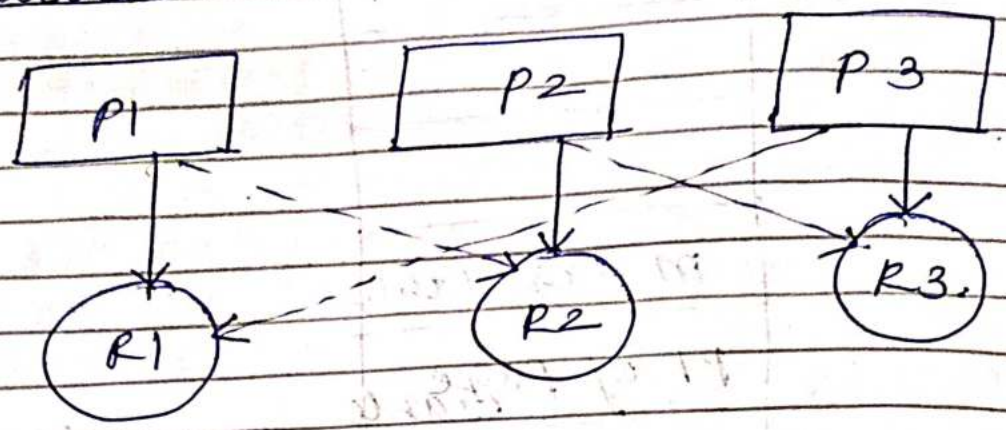
(3) Factorial of a program.

(progs can be asked in the exam)

* a
aa, ab
aab, aba, abb
aaab, aabb, abab, abbb, etc..
a + aa + ab + aab + aba + abb...
a (1 + a + b + ab + ba + ...)
ab (a+b)*

(Theory of computation)

* Deadlock :



- Process P1 is holding Resource R1 and requesting for R2.
  Process P2 is Holding R1 and requesting R3
  Process P3 is holding P3 and requesting for R1
- None of the processes will complete the process until they get the requested process.
- So, it becomes a circular deadlock condn.
- Deadlock is a condition where os can't continue the execution because processes demanding those processes resources which are holded by the processes. So the none of the process can continue its execution hence the deadlock occur.

* Necessary conditions to avoid dead-lock:

* Mutual Exclusion :-
- Mutual exclusion is a property of process synchronization that states that "no 2 processes can exist in the critical section at any given point of time."
- Mutual exclusion in OS locks is a frequently used method for synchronizing processes or threads that want to access some shared resource.

* No-Pre-emption : If a process is holding a resource no other process can pre-empt it.

* Hold and wait : Processes are allowed to make request for other resources while holding one.

* Circular wait : while holding and waiting for the ~~process~~ resources there should be a circular wait.

* Deadlock Handling Techniques:
(1). Deadlock avoidance or Ignorance: It can be avoided by restarting the system or shutting down the system for sometime. We can write
~~(2). Deadlock Prevention~~: a code for deadlock for avoiding deadlock but if deadlock occurs very rarely then writing a code is not efficient thing for CPU.

* Questions can be asked on semaphore. in CCEE

* Producer - Consumer Problem:

0	Item 1.	$n = 6$. count = 1.
1	Item 2	count = 2
2	Item 3	~~cou~~ count = 3.
3		
4		
5		

· code of producer:

```
void producer ()
{
int itemP;
while (1)
{
  Itemp = new Item();
  while (count == n);
  Buffer [in] = ~~ter~~ Itemp;
   [in] = [in+1] mode n;
   count = count +1;

  }
}
```

* count operation: count is loaded from main memory to registers. Registers set instructions for increm-enting the count variable.
After updating the variable, it is again loaded into main memory. (same vice-versa process happens for decrementing process).

(Basic operation of count variable in every prog)

- ~~void consumer ()~~ code for consumer:

```
void consumer ()
{
 int item c;
 while (1)
 {
 while (count = =0);
  {
   while (count = =0);
   Item c = Buffer [out];
   out = [out+1] mod 4;
   count = count - 1;
  }
 }
}.
```

- False condⁿ occurs because of the interruption.
- This problem can be solved by no interruption in Producer or consumer problem and by adding critical section. which will ~~add a lock~~ can add a lock to the running producer / consumer code.
- This interruption leads to Racer condition.