

Complete notes on

Operating System

Copyrighted By:-

Instagram:- coders.world

Telegram:- codersworld1

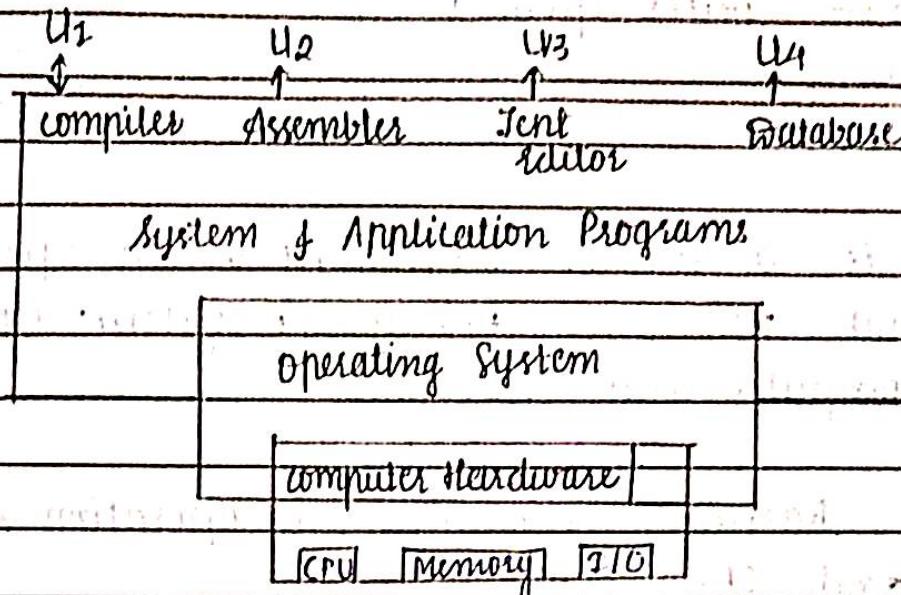
Written By:-

Sanskriti Satpute
(Software engineer)



DATE: / /

OPERATING SYSTEMS



Structure of computer system:

- Computer systems can be divided into types.
- 1) Hardware - It provides basic compiling resources. (CPU | memory | I/O) devices.
- 2) OS operating system: O.S. controls & co-ordinates the use of hardware among various app. & users.
- 3) Application programs : defines the way in which system resources are used to solve computing of users (compilers ..) (Assembler, editor, database)
- 4) Users : people, machine (printer, scanner or other computers)

DATE: / /

PAGE NO.:

OS prioritizes the work of application (eg - phone in better game)

Kernel -

- It is a control component of an operating system that manages operations system operations of computer and hardware.
- It basically manages the operation of memory and CPU type.
- It is more core component of OS.
- It acts as bridge between applied f data processing performed at hardware level using interprocess communication f system call.

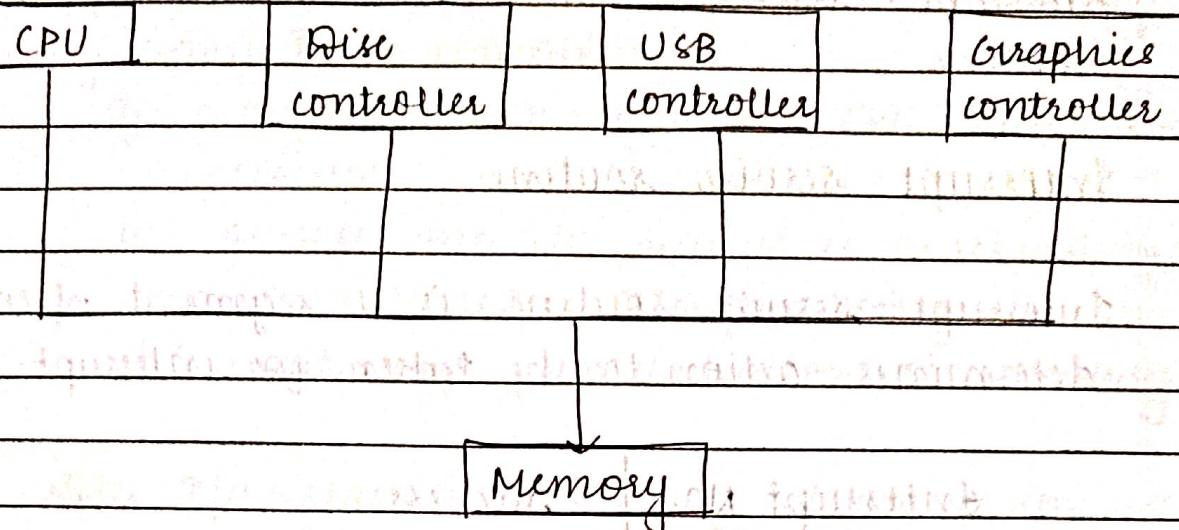
Bootstrap Program.

- It is a program which is loaded at power up or reboot.
- It is typically stored in ROM f EPROM
- It is generally known as firmware
- It initialises all aspects of system.
- It loads the operating system kernel f start its execution.

DATE: / /

PAGE NO. :

Computer system organisation



- 1) one or more CPUs, device controllers connects through common bus providing access to shared memory concurrent execution of CPUs & devices competes for memory cycles.
- 2) I/O devices & CPU can execute concurrently
- 3) Each device controller is in charge of particular device type
- 4) Each device controller has local buffer
- 5) CPU moves the data from I/O to main memory to/from local buffer
- 6) I/O is from device to local buffer of controller
- 7) Device controller informs CPU that it has finished its operation by causing an interrupt.

Interrupts

Interrupt transfer the control to interrupt service routine.

Interrupt service routine

Interrupt service routine is a segment of code that determines action to be taken for interrupt.

Interrupt No.	Address
0	003h
1	002h
2	hexadecimal
3	

Determining Type of Interrupt

- Polling -

same interrupt handler called for all interrupt; which then polls all device to figure out the reason for interrupt.

- a) interrupt vector Table

Different interrupt handler will be executed for diff interrupt

DATE : _____

PAGE NO. :

I/O Structure

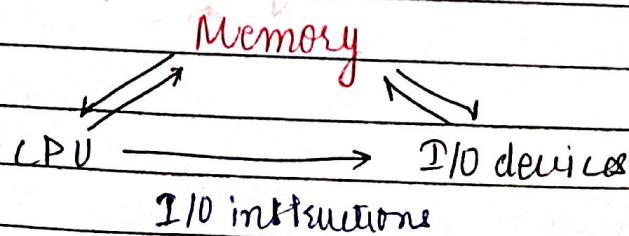
- 1] After I/O starts, control returns to user programs only upon I/O completion.
 - a) wait instruction - idles the CPU until next interrupt.
 - b) Almost one I/O request is outstanding at a time no simultaneous I/O processing
 - c) wait loop (concentration for memory access)
- 2] After I/O starts, the control returns to the user program w/o waiting for I/O completion.

System call - request to O.S to allow user to wait for I/O completion.

Device status Table

- a) contains entry of each I/O device indicating its type address of state
- b) OS indexes into I/O devices table to determine device status & to modify table entry to include the interrupt.

Direct Memory Access



- 1) Typically used for I/O devices with lot of data to transfer in order to reduce load on CPU
- 2) Device controller transforms the block of data from buffer storage directly to main memory w/o CPU interruption.
- 3) Device controller interrupts the CPU on completion of I/O.

Storage Hierarchy.

Registers

Cache memory

Main memory

SSDs

Electronic disc

Magnetic disc

Optical disc

Magnetic tapes

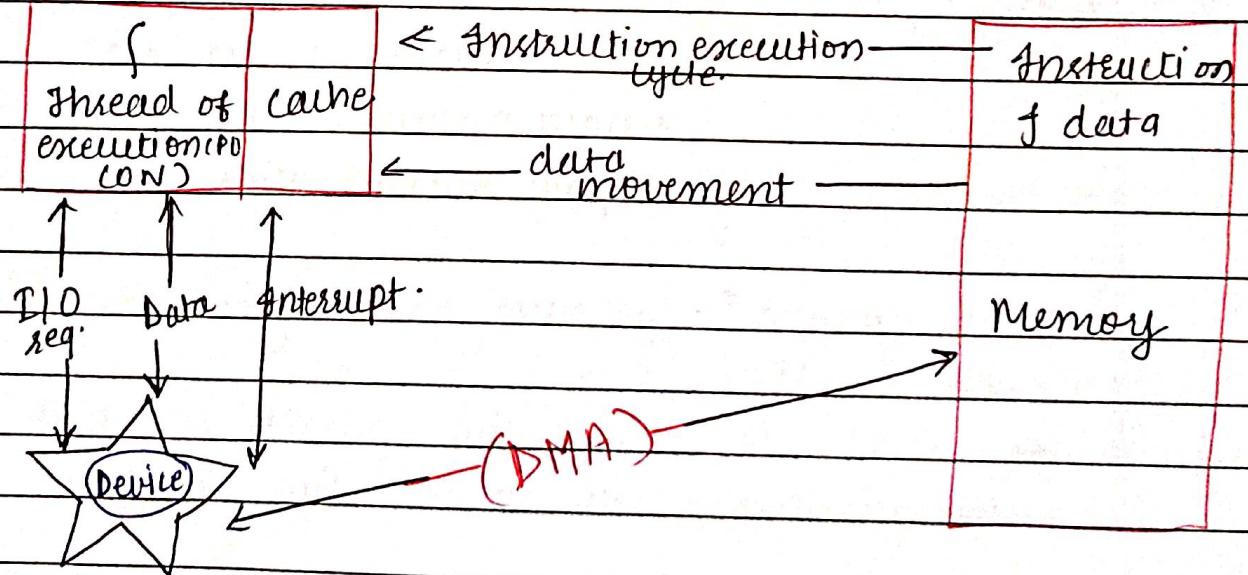
DATE: _____

PAGE NO.: _____

Caching

- 1) It is an imp principle performed at many levels in comp. (hardware, software, O.S.)
- 2) Information in use is copied from slower to faster storage temporarily.
- 3) Faster storage (cache) check first to determine if information is there.
 - a) If it is, informationalised directly from cache.
 - b) If not, data copied to cache & used there.
- 4) Cache are smaller than the storage being cached.
- 5) O.S manages the cache size & replacement policy.

★ Computer System Architecture



DATE : 20 / 1 / 22.

PAGE No. 5

Kernel entry points

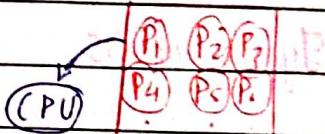
→ what are diff event in OS.

In computing environment event is an action or occurrence detected by the program that may be handled by the program.

System call

~~30/01/22~~ Types of operating system.

1) Multiprogrammed OS



→ Non preemptive

→ Idleness is decreased.

→ If CPU will execute a process, it will execute completely.

2) Multitasking / Time sharing → mostly used.

(Round Robin)

→ Time is shared, will execute the process for specific duration, will execute the remaining after.

→ not idle.

→ But response time is decreased.

DATE: / /

3) Real time OS Hard soft

— XOX —

21/01/22

Introduction to OS.

- An OS is a program that manages computer hardware.
- It also provides a basis for application programs and acts as an intermediary betⁿ computer users & hardware.
→ windows, linux, ubuntu, Mac OS x TOS, Andri

Types of OS.

Goals of OS

- i) Batch OS
- ii) Time sharing OS
- iii) Distributed OS
- iv) Network OS
- v) Real time OS
- vi) Multiprogramming
- vii) Multiprocessing / tasking

i) convenience

ii) efficiency

iii) Both

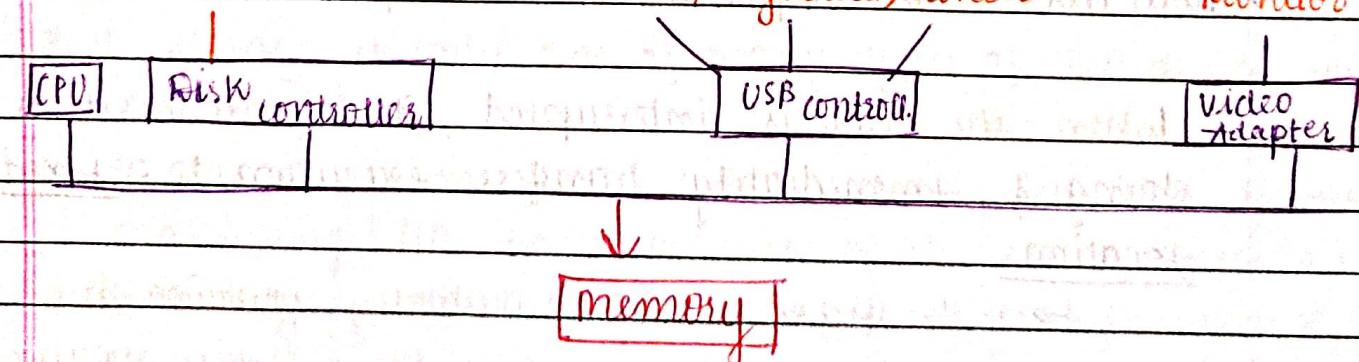
→ Functions.

- It is an interface betⁿ User & Hardware
- Allocation of Resources
- Management of memory, security etc.

Basic of operating system (computer system operation)

- A modern general-purpose computer system consists of one or more CPUs and a no. of device controllers connected through a common bus that provided access to shared memory.
- Each device controller is in charge of a specific type of device.
- The CPU and the device controllers can execute concurrently, competing for memory cycles.
- To ensure orderly access to all the shared memory, memory controller is provided whose fun' is to synchronize access to the memory.

Risks *Additional mouse, keyboard, printer and monitor*



Some important terms

- 1) **Bootstrap Program** - The initial program that runs when a computer is powered up or rebooted.
- It is stored in the ROM (secondary memory).
 - It must know how to load the OS.

DATE: 17/1/2023

- and start executing that system.
- It must locate and load into memory thru OS kernel.

(Heart of OS)

2) Interrupt

- The occurrence of an event is usually signalled by an interrupt from Hardware or software.
- Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually thru the way of system bus.

3) System call (monitor call)

- Software may trigger by executing a special op-code called system call.

→ How does CPU responds to interrupt

When the CPU is interrupted, it stops what it is doing & immediately transfers execution to a fixed location.

↳ The fixed location usually contains the starting address where the service routine of the interrupt is located. (place where what interrupt want to do is written)

Interrupt service routine executes

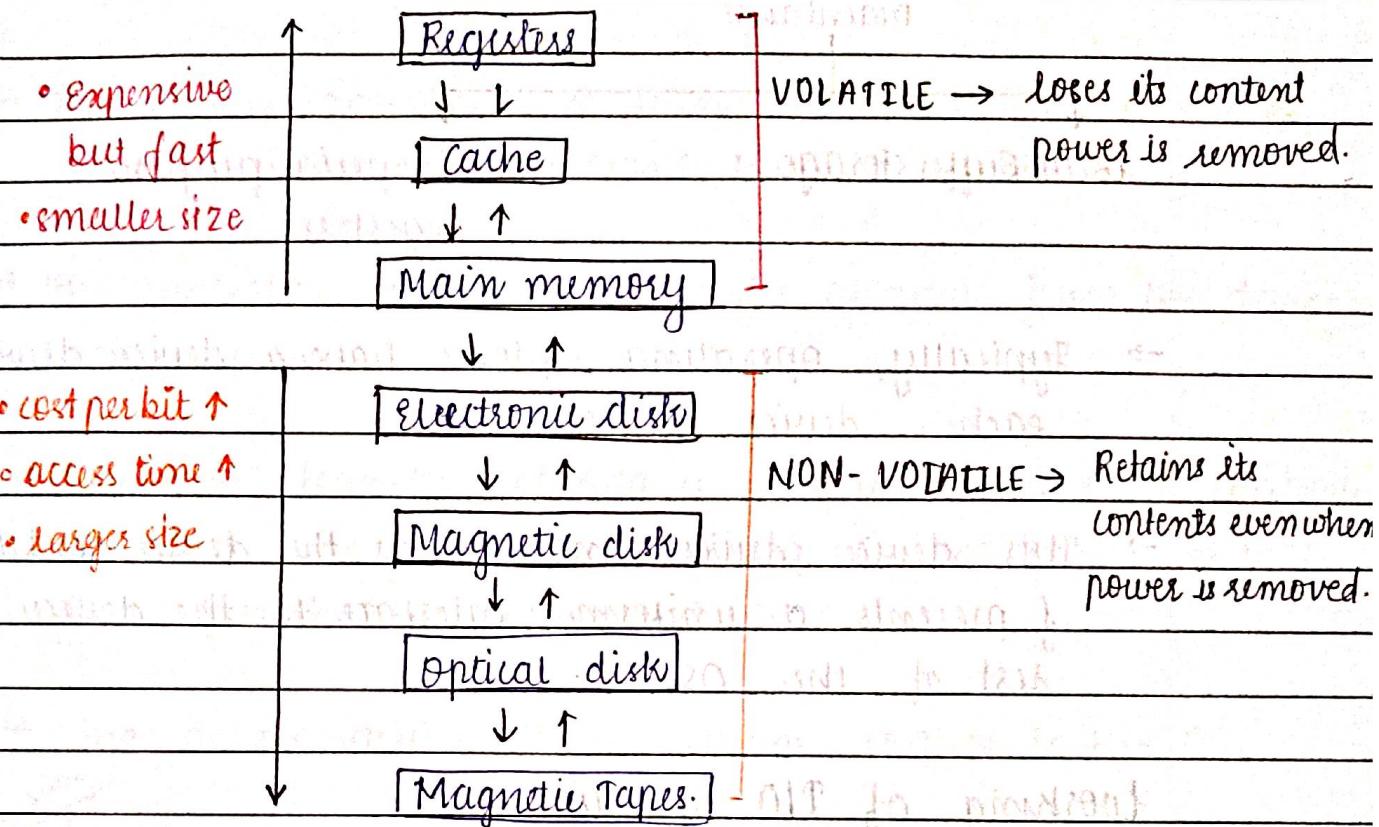
on completion, the CPU resumes the interrupted computation

DATE: _____

PAGE No.: _____

* special case
(NVRAM → non volatile)

Basics of OS [storage structure]



(I/O Structure)

- A storage is only one of many types of I/O devices within a computer.
- A large portion of OS code is dedicated to managing I/O both because of its importance to the reliability and performance of a system & because of the varying nature of the devices.
- A general-purpose comp. system of CPUs & multiple device controllers that are connected through a common bus.
- Each

DATE: 11/11/23

Each device controller is in charge of a specific type of device

maintains
local Buffer storage

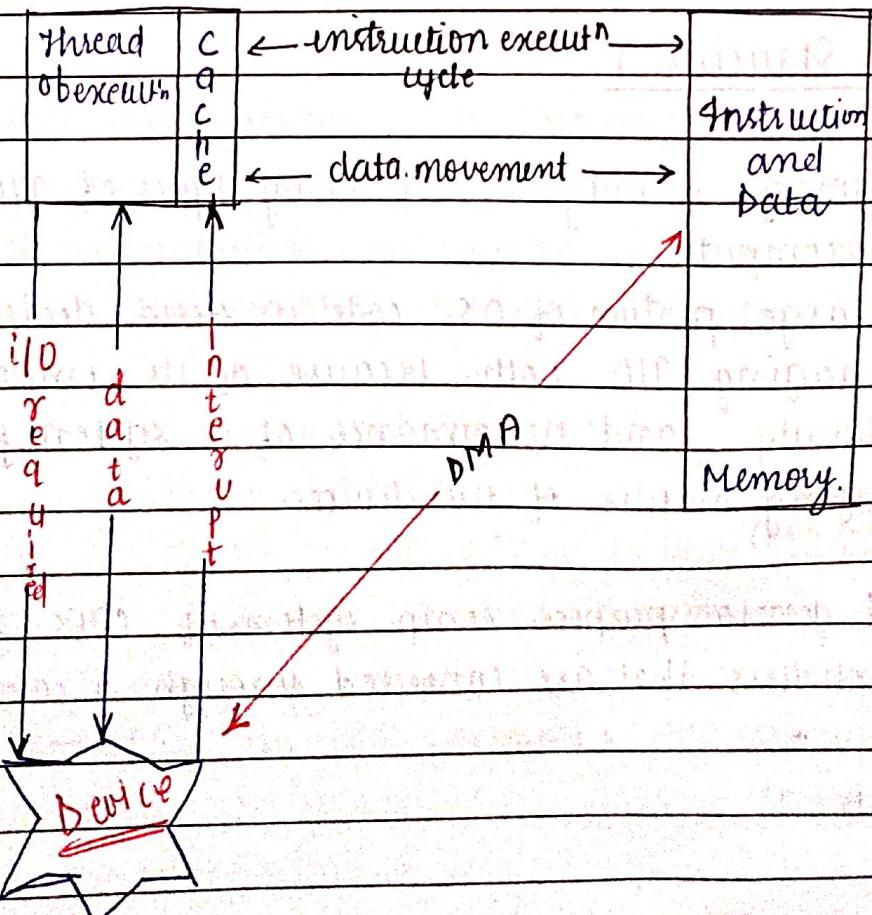
set of special purpose

Registers

→ Typically, operating systems have a device driver for each device controller

→ This device driver understands the device controller & presents a uniform interface to the device to the rest of the OS.

Working of I/O operation



DATE: _____ / _____ / _____

PAGE No.: _____

- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take.
- The controller starts the transfer of data from the device to its local buffer.
- Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- The device driver then returns control to the OS.

~~Dis Adv.~~

This form of interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement.

- To solve this prob. Direct memory Access is used.
- after setting up buffers, pointers and counters for the I/O device, the device controller transfers entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU.
- only one interrupt per block, to tell the device driver the op. is completed.
- while the device controller is performing these op. the CPU is available to accomplish other work.

DATE : 20/11/2021

PAGE NO. 1

Computer System Architecture

Types of computer system on no. of general purpose processors:

1) Single Processor System.

→ one main CPU capable of executing a general purpose instruction set including instructions from user processes.

→ Other special purpose processors are also present which perform device specific tasks

(microprocessor chip in keyboard)

2) Multi Processor System / Parallel or tightly coupled

→ Has two or more processors in close communication sharing the computer bus and sometimes the work, memory and peripheral devices.

Adv.

→ increased throughput (measure of perform. of system)

→ economy of scale. (as here sharing of resources)

→ increased reliability.

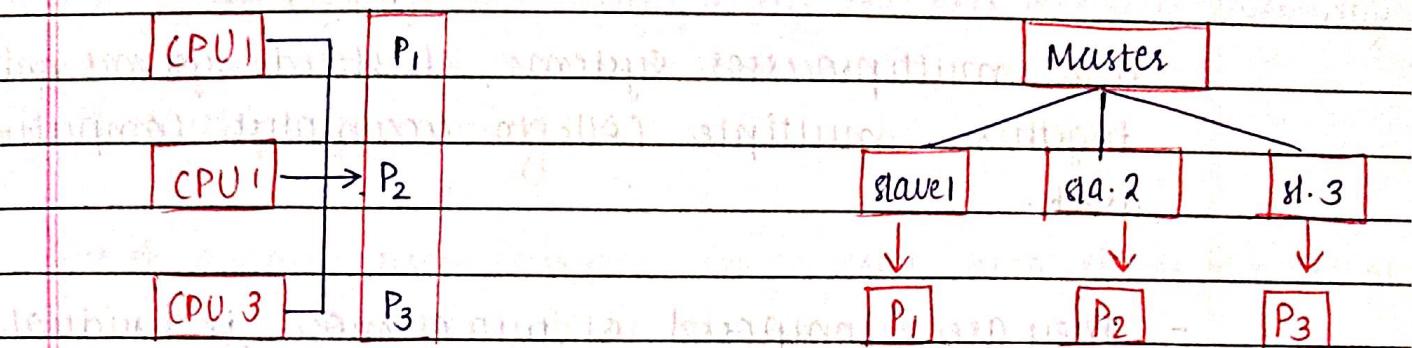
DATE: _____

PAGE NO.: _____

→ Types of multiprocessor system.

• Symmetric

• Asymmetric Multiprocessing



→ all the processors are similar to each other. Master → slave approach.

→ commonly used

→ all processors share physical memory.

The master allocates

processor schedules &

allocate the work to the worker processes.

* → many processes can run simultaneously
($N \text{ CPUs} \rightarrow N \text{ Processes}$)

→ Multiprocessing adds CPUs to increase computing power.

* → one CPU sits idle while others overload

→ M-P can cause a system to change its memory access model from uniform memory access to (NUMA)

→ UMA is defined as the situation in which access to RAM from any CPU takes the same amount of time.

* All multicore sys → multiprocessors (not vice versa).

DATE: 10/10/2023

PAGE NO.: 10

- Blade servers → multi processor board, I/O boards & network boards are placed in the same chassis connected via interconnect.

3) Clustered system.



- like multiprocessor systems, clustered systems gather together multiple CPUs to accomplish computational work.
- They are composed of two or more individual systems coupled together.

→ Provides high availability.

→ Can be structured:

asymmetrically



- One machine in Hot-standby mode.

- Other runs application.

✓ (more efficient)

symmetrically



- Two or more host run application.

- Monitor each other.

- Other forms are parallel clusters → allows host to access multiple storage units simultaneously

allows host to access the same data on shared storage

Clustering over WAN

DATE: / /

PAGE NO.:

Operating System Structures

- (Multiprogramming & Multi-tasking) → commonalities
- Operating system vary greatly in their makeup internally

i) Multi programming.

- A single user cannot, in general, keep either the CPU or the I/O devices busy at all times.
- Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

Operating sys.
Job 1
Job 2
Job 3
Job 4
512M

job 1, job 2
job 3, job 4,
job 5

Job pool

- CPU doesn't remain idle

- Efficient.

- Multiprogramming systems provide an environment in which the various system resources (for ex- CPU, memory, and peripheral devices) are utilized effectively but not provide interaction with computer system.

DATE: _____ / _____ / _____

iii) Time Sharing (Multitasking)

- CPU executes multiple jobs by switching among them (virtual memory)
- Switches occur so frequently that the user can interact with each of program while it is running.
- Time sharing requires an interactive (or hand on) computer system, which provides direct communication betⁿ the user of the system.
- A time shared system allows many users to share the computer simultaneously.
 - It uses CPU scheduling & multiprogramming to provide each user with a small portion of a time-shared computer.
 - Each user has at least one separate program in memory.
 - A program loaded into memory & executing is called a "PROCESS"

Operating System Services

- An OS provides an environment for the execution of programs
- It provides certain services to program and to users of those programs.

DATE: 10/10/2023

PAGE NO.:

iii) Time Sharing (Multitasking)

- CPU executes multiple jobs by switching among them (virtual memory)
- Switches occur so frequently that the user can interact with each program while it is running.
- Time sharing requires an interactive (or hand on) computer system, which provides direct communication bet'n the users of the system.
- A time shared system allows many users to share the computer simultaneously.
 - Uses CPU scheduling & multiprogramming to provide each user with a small portion of a time-shared computer.
 - Each user has at least one separate program in memory.
 - A program loaded into memory & executing is called a "PROCESS"

Operating System Services

- An OS provides an environment for the execution of programs
- It provides certain services to program and to users of those programs.

DATE: _____

PAGE NO.: _____

ii) User Interface: (feel, memory, program) available

→ interaction b/w user & computer

ex - command line, graphical

iii) Program Execution

source code → compiler → object code → executor → output

iv) I/O operations

ex - keyboards → CPU → monitor

v) File system manipulation.

create, delete, modify, search, access restriction of file.

vi) Communications

vii) Error detection

viii) Resource Allocation

ix) Accounting

x) Protection & Security

User Operating System Interface

- i) Provide a common-line interface or command interpreter that allows users to directly enter commands that are to be performed by the operating system.
- ii) Allows the user to interface with the O.S via a graphical user interface or GUI.

Command Interpreter

- Some OS include the command interpreter in the kernel.
 - Others, such as Windows XP & UNIX, treat the command interpreter as a special program.
ex → in windows → code prompt
 - On systems. multi command interpreters to choose from, the interpreters are called as shells
- Eg:-

Bourne shell, C shell, Bourne-Again shell, Korn shell.

example:

We want to perform a task → create a file

first approach → code included in command interpreter itself
→ that code will be executed.

- * Command interpreter itself doesn't contain any code. But the codes are written in certain program

Second app

command interpreter calls the program where code is written to create it.

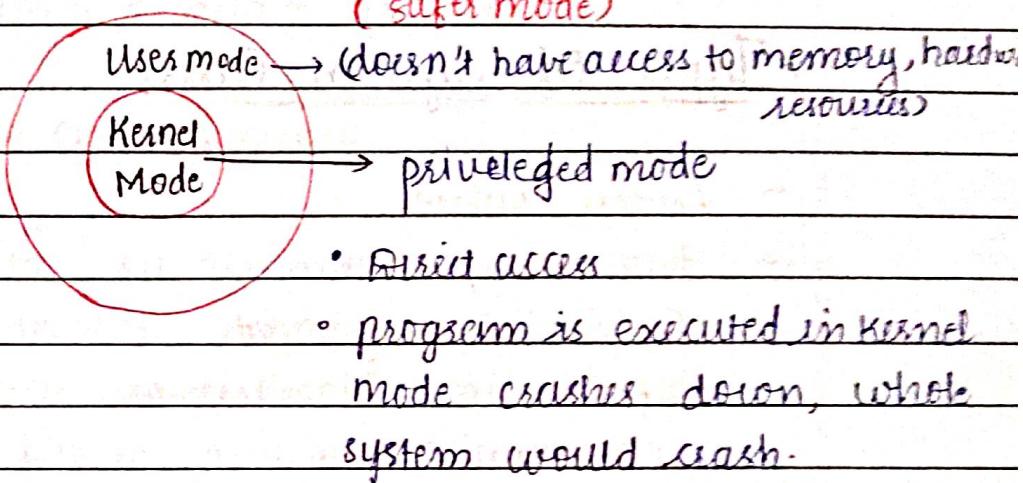
DATE: _____

PAGE NO.: _____

System Calls

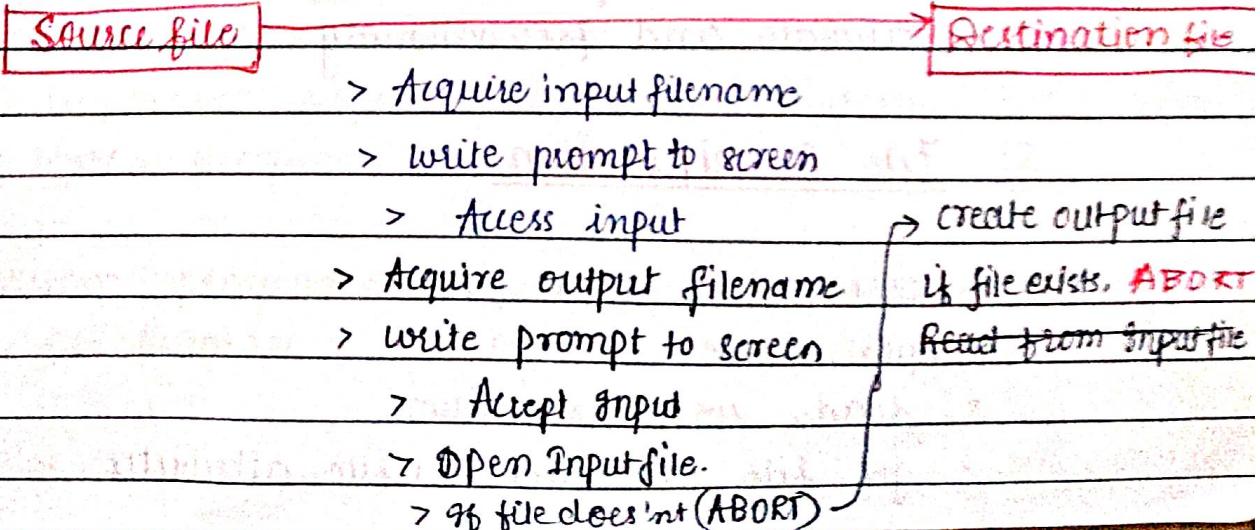
System calls provides an interface to the service mode available by an operating system.

(safe mode)



- System call is a programming way in which a computer program requests a service from the kernel of the OS.
- These calls are generally available as routines written in C and C++.

Ex:-



loop Read the input file.
until Read fails write to output file
 close output file
 write completion to screen
 Terminate normally.

> Types Of System Calls.

(5 types)

- Process control
- File manipulation
- Device management
- Information Maintenance
- Communications

1] Process Control

- end, abort
- load, execute
- create process, terminate process
- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory.

2] File Manipulation

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set file attributes

DATE: _____

PAGE No.: _____

3) Device Manipulation

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices.

4) Information Maintenance

- get time or data, set time or data
- get system code data, set system data
- get process, file or device attributes
- set process, file or device attributes.

5) Communication System Calls

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

System Programs

- > an important aspect of a modern system is the collection of system programs.
- System programs provide a convenient environment for program development and execution.

DATE: / /

PAGE NO.

- Some of them are simply user interfaces to system calls others are considerably complex.

ii) File management

- > create
- > copy
- > print
- > delete
- > rename
- > dump
- > list & generally manipulate files & dirs

iii) Status Information

Ask the system for -

- Date, time
- amount of available memory or disk space.
- Number of user
- detailed performance
- logging & debugging information etc.

iv) File modification:

- > several text editors may be available to create & modify the content of files stored on disk or other storage devices.
- > There may be also be special commands to search contents of file or perform transformation of the text.

DATE: _____

PAGE NO.: _____

iv) Programming - language Support

- compilers
- assemblers
- Debuggers (such as GDB)
- Interpreters (such as C, C++, Java, Visual Basic and PERL)

v) Program loading and execution

Once a program is assembled or compiled, it must be loaded into memory to be executed.

The system may provide:

- Absolute loaders
- Relocatable loaders
- Linkage editors and
- Overlay loaders.

Debugging system for either high-level language or machine languages are needed as well.

vi) Communications

These programs provide mechanism for:-

- ii) Creating initial conn' among processes, user f com. sys.
- iii) Allowing users to send messages to one → another comp.
- iv) To browse webpage
- v) To send electronic-mail messages
- vi) To log in remotely or to transfer files from one → another.

DATE: 2021/10/1

PAGE NO. 1

Operating System Design & Implementation

• Design goals.

1st problem: Refining goals & specifications

- choice of hardware

- type of system.

Beyond the high design level, the requirement may be much harder to achieve.

Requirements

• User goals:

- convenient to use

- easy to learn & use

- reliable, safe & fast

• System goals

(Designer, eng. seq.)

- easy to design, implement

- maintain, operate

- It should be flexible, reliable, error free & efficient

Mechanisms & Policies



(How to do things)

(determine what will be done)



* one imp. principle that, the separation of policy from mechanism.

Implementations

- Once an OS is designed, it must be implemented
- Traditionally, OS have written in assembly language
- Now, however the are most commonly written in higher level languages such as C or C++.

Advantages.

- The code can be written faster.
- It is more compact.
- It is easier to understand.
- It is easier to port.

Disadvantages.

- Written in assembly lang. can't operate on many devices.

Ex: MS-DOS in Intel 8088 A.I. available only on Intel family of CPUs.

Structure of OS.

Simple Structure

Application program

Resident Sys. Program

Device drivers

ROM BIOS device drivers

DATE : 10/10/2023

PAGE No. 1

Monolithic Structure. → (UNIX)

The users

shells and commands

compilers and interpreters

system libraries

System-call interface to the kernel

Kernel

signal, terminal
handling

character, I/O sys.
terminal devices

file system

swapping block I/O

file system

disk & tape drives

CPU scheduling

page replacement

demand paging

virtual memory

Kernel interface to the hardware

terminal controllers
terminals

device controllers

disk & tapes

memory controllers

physical memory

diff. maintain & implemented.

Layered structure.



Adv.

→ Every layer have diff. func.
so debugging is easy.

Disadv.

Deciding the position of
layer

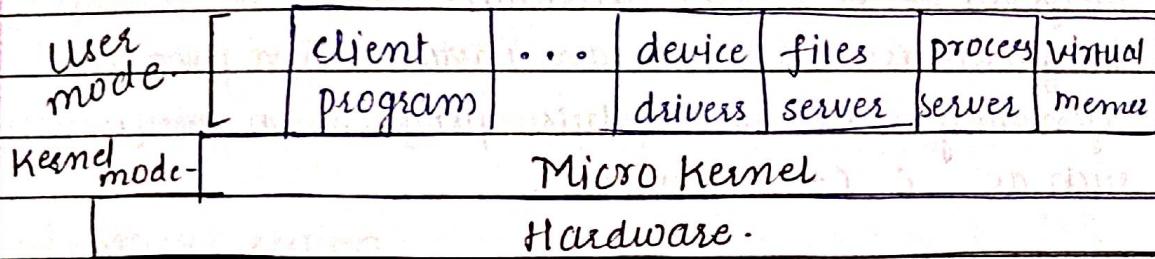
as layer can use layer
below it

DATE : _____

PAGE NO. : 10

- not very efficient
- as it has go layers by layer one by one.

Microkernels



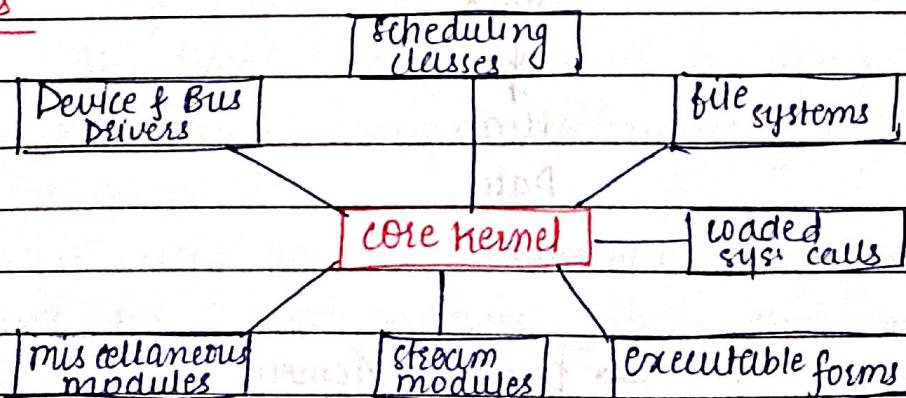
Adv

- To Remove non-essential component from system. (only core implemented others implemented as system programs).
- most of the programs are in user mode.
- Crashing of entire program won't happen.

Disadv

- Since the communication needs to be done always, system overhead will take place.

Modules



→ resembles micro-kernel

→ each module can communicate directly

→ loaded directly. no system overhead.

→ By far The Best.



DATE: 27/01/22

PAGE NO.

Unit - 2 Process & It's Scheduling

2.1] Process

- Program is a under execution.
- Program is consider as an instance of a program
- Generally user writes their program in specific language such as C, C++, Java etc.

Program → compile → execute → Process

- Program consists of multiple threads, where thread nothing but light weight process.
- Program is a passing entity but process is an active entity when it loads into memory.
- Several process may be associated with same program e.g. opening up several instances of

Stack



Heap

Data

Text

→ Process in Memory



Your sections are -

i) Text

- consists of source code.
- current activity are represented by value of program counter and content of processor's registers.

ii) Stack section

- stack memory hold temporary data such as func parameters, return address, local variables

Ex - int main()

```
{ int a, b;
    sum();
}
```

void sum()

```
{
    int s1, s2;
}
```

when sum() is executed, the memory will be cleared from stack memory f would go to main memory.

[stack]

- In Iteration - no new stack memory will form
- In Recursion - new stack memory is formed

→ stack is used for local variables, space on the stack is reserved for local variables when they are declared.

Space is freed up when variables go out of scope.

DATE : 20/11/2023

PAGE NO. 1

iii) Data section.

- Global variables, static variables which needs to be allocated & initialized prior to executing main program.

iv) Heap section

It is used for dynamic memory allocation, and is managed via calls to new, delete malloc, free

2.2) Process States

- During the process execution, it changes its state from one state to another state
- The state of process is defined in part by the current activity of that process.
- State of process represents its current status
- Each process may be in one of the follⁿ state.

DATE : _____ / _____ / _____

PAGE NO. : _____

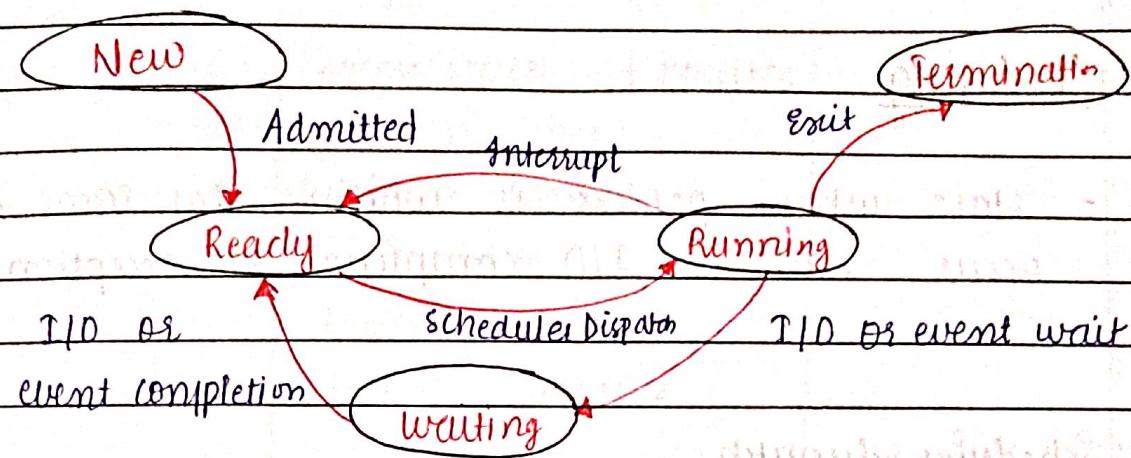


Fig: Process state

i) New

This is the first state of process in which process belongs when it is created.

Ready

This is the second state, in which process belongs when it is ready to execute. Scheduler may pick up the processes from ready queue and submit it to CPU for execution.

Running

This is the state where actual execution of process is carried out. From running state process may go to one of following state.

- ij) Terminated state: If process successfully executed
- ii) Ready state: If interrupt occurs
- iii) Waiting state: When any I/O event occurs.

DATE: 11/11/2023

PAGE NO.: 1

Waiting

- State where process is waiting for some event to occur such as I/O completion or reception of a signal.

Scheduler dispatch:

- Scheduler dispatches from ready state to running state.

Terminated.

Process enters into this state when it finishes execution.

2.3] Process Control Block.

- The OS needs some information related to process, which is stored in data structure called process control block.

- Each process in OS represented by process control block.

- PCB is also called Task control block

- PCB contains many pieces of information associated with specific process.

DATE: _____ / _____ / _____

PAGE NO.: _____

Process state	Pointer
Process number	
Program counter	
Registers	
Memory limits	
list of open files	

Fig: Process Control Block

ii) Process State

The state may be new, ready, running, waiting and halted.

iii) Program Counter

The counter indicates the address of next instr to be executed for this process.

iv) CPU registers

The registers vary in number and type of depending on computer architecture.

It includes accumulators, index registers, stack pointers and general purpose registers.

v) Process Number

It represents identifier of the process

DATE : / /

PAGE NO. :

v) Memory management information.

It include information as value of base and limit registers , the page tables or segment table based on memory system.

vi) Pointer

It include pointer to next PCB that is pointed PCB of the next process to run.

vii) I/O status Information

It includes the list of I/O devices allocated to process a list of open files

2.4]

Process Scheduling

3.

- Process scheduling is the technique adopted by OS which governs the order in which process need to be exercised.
- In order to achieve multiprogramming environment in OS which governs we require that some process is always running , which helps in maximizing CPU utilization.
- To achieve this CPU need to switch from one process to another process
- To keep CPU busy we need some entity that assign Job's to CPU and helps in decreasing idle times of CPU & increasing CPU utilization.

DATE: _____

PAGE NO.: _____

Scheduling Queue.

- This is the data structure which keep set of processes
- scheduling queue refers to queue of processor or devices.
- when the process enters into system then this process is put into queue.
- O.S maintains diff. types of queues which are as follows:

i)

i) Job Queues.

Initially when processes enters into system they are always put into queue called as Job queue if consists of all processes in the system.

ii) Ready Queue.

The processes that are residing in main memory and are ready and waiting to execute are kept on list called ready queue.

- This queue is generally stored as linked list.
- A ready queue header contains pointers to first and final PCB in the list.

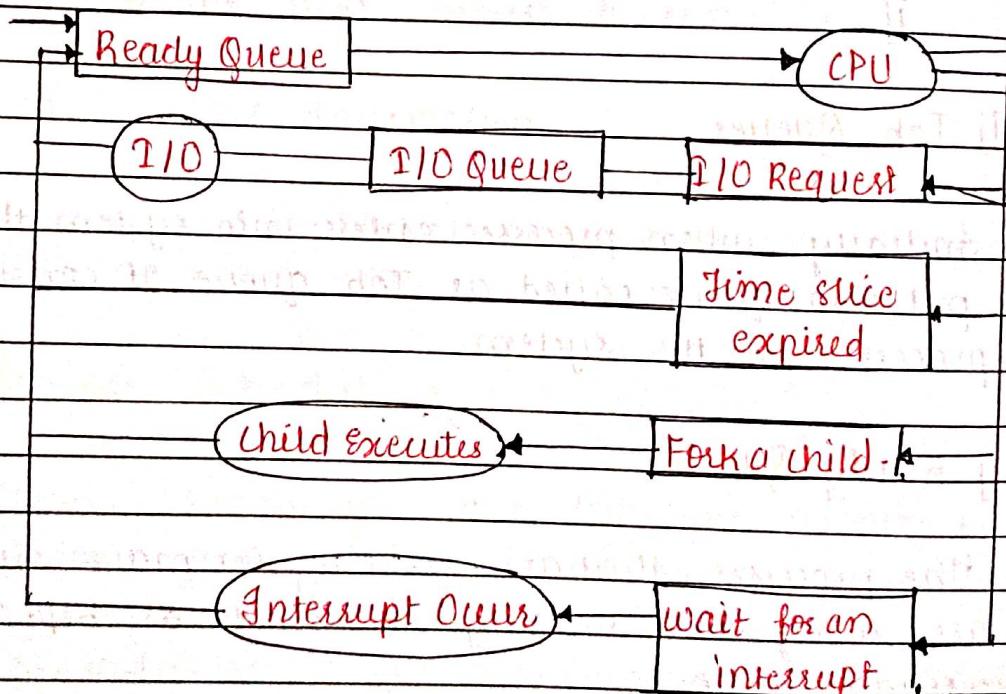
DATE: 22/1/21

PAGE NO.:

iii) Device queue.

The list of processes waiting for a particular device is called device queue. Each device has its own device - queue.

- A common representation of process scheduling is a queuing diagram.



In above fig. rectangular box: represents a box queue : as ready queue. and device queue.

The circle represents the resources that serve the queue and arrow indicate flow of processes in the system.

Schedulers.

- Scheduler is an entity responsible for arranging the processes in proper order and then submitting it to CPU for execution.
- Scheduler arranges the processes into an approximate seq.
- As process migrates among various scheduling queues throughout its lifetime, this task of migration is carried out by scheduler.
- Scheduler is an intermediate level of scheduling.
- There are diff types of scheduler

i) Long Term Scheduler.

- a.k.a Job scheduler.
- responsible for selecting the process from Job queue to ready queue
- Simply the scheduler picks the process from secondary memory and load it into primary memory.
- works as a gate keeper
- Also called as admission scheduler.
- It is very important in the large system like super comp.
- It also controls the degree of multiprogramming
- Unix system doesn't make use of long term scheduler.
- Long term scheduler executes less frequently.

ii) Short Term Scheduler or CPU scheduler 10

- It select the process from ready queue that are ready to execute and submit the process to CPU for execution.
- This scheduler temporarily remove process from main memory and submit it to CPU.
- Short term scheduler make scheduling decision more frequently.
- This scheduler can be a pre-emptive that meant it is capable of forcibly removing process from CPU when it decide to allocate CPU to another process.

iii) Medium Term Scheduler

- It select the process from ready queue that are ready to execute and submit the process to CPU for execution.

This scheduler temporarily remove process from main memory and submit it to CPU.

Short term scheduler make scheduling decision

iii) Shortest Job First scheduling.

- pre-emptive approach.
- shortest job response approach.

Ex. 3)

Process Arrival time Burst time

P ₁	10	11	19	81	87	1
----------------	----	----	----	----	----	---

P ₂	1	1	1	1	1	4
----------------	---	---	---	---	---	---

P ₃	2	9				
----------------	---	---	--	--	--	--

P ₄	3	5				
----------------	---	---	--	--	--	--

P ₁	P ₂	P ₄	P ₁	P ₃	
0	1	5	10	17	26

$$W.T = P_i = \text{final exec. time} + \frac{\text{previous burst time}}{\text{arrival time}}$$

$$P_1 = 10 - 1 - 0 = 9$$

$$P_2 = 1 - 1 = 0$$

$$P_3 = 17 - 2 = 15$$

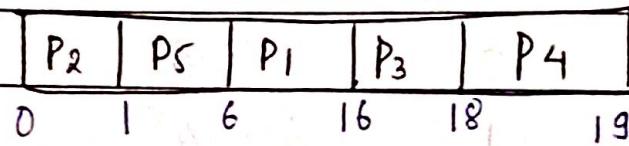
$$P_4 = 5 - 3 = 2$$

Ex. 4) Priority Scheduling.

Process Arrival time

1 → P.T.O

Process	Burst time	Priority
P ₁	10	3
P ₂	1	1
P ₃	2	4
P ₄	1	5
P ₅	5	2



$$A.W.T = \frac{0 + 1 + 6 + 16 + 18}{5}$$

$$= \frac{41}{5} - 8.2$$

→ Priority can't be same for any processes
But Burst and arrival time ~~can~~ can be.

→ Round-Robin Algorithm

→ Pre-emptive

Process	Burst time
P ₁	24
P ₂	3
P ₃	3

Time quantum: 4ms

DATE: _____ / _____ / _____

PAGE No.: _____

	P ₁	P ₂	P ₃	P ₁				
	0	4	7	10	14	18	22	26

$$\begin{aligned} P_1 &= 26 - 20 = 6 \\ P_2 &= 4 \\ P_3 &= 7 \end{aligned} \quad \left. \begin{array}{l} \text{W.T.} \\ \text{3 minutes} \end{array} \right\}$$

$$\text{Avg. wt} \rightarrow 17 - 5.66 \text{ ms.}$$

Multi-level Queue Scheduling.

- Another type of scheduling algo. created for situation in which processes are qualified into diff. groups.

Groups →

◦ Two types of processes are foregr.

i) foreground (interactive processes)

ii) Background (Batch processes)

→ system processes → highest Priority

→ interaction processes

→ interactive editing processes

→ Batch processes

→ student processes

lowest Priority

DATE: 20/11/2023

PAGE NO. 11

→ Process once entered in queue, can't switch the queue.

Multilevel Feedback Queue Scheduling

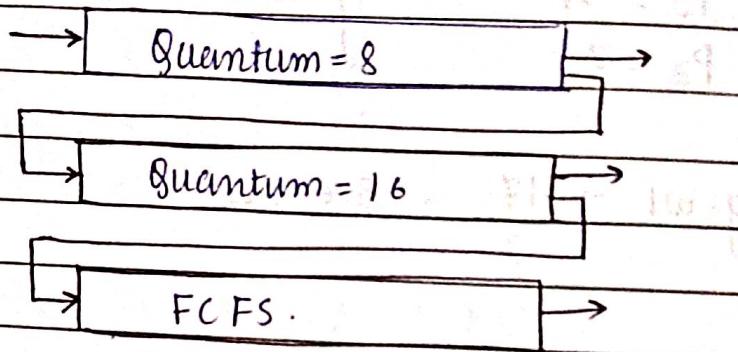


Fig: MFQS

This algo is very similar to ordinary MQS, except job may move from one → another for variety of reasons such as.

i] If characteristics of job change b/w CPU intensive and I/O intensive.

ii] Job has waited for long time can get bumped into higher priority queue. → Mult

iii] Multilevel feed Q is most flexible as it can be tuned for any situation.

iv] It is most complex to implement because of all adjustable parameters.

v] Some of the parameters which define one of these systems include

a) no. of queues

b) scheduling algo. for each queue

DATE : / /

PAGE NO. :

- c) method to upgrade/demote process from one queue to another
- d) method used to determine which a process enters initially

*

Ex-1	Process	Burst time	Priority	Arrival time	
	P ₁	5	1	0	T _Q = 2 ms
	P ₂	7	2	5	T _Q = 2 ms
	P ₃	6	3	10	

① SJF (pre-emptive)

BP ₃	P ₁	P ₃	P ₂	
0	1	6	11	18

$$W.T. = P_1 = 6 - 1 = 5 \text{ ms}$$

$$P_2 = 11 - 5 = 6$$

$$B_3 P_3 = 11 - 0 = 11 \text{ ms}$$

$$\text{Avg. wt} = 11/3 = 3.66 \text{ ms}$$

$$TAT = P_1 = 6 - 1 = 5$$

$$P_2 = 18 - 5 = 13$$

$$\text{Avg. TAT} = 29/3 = 9.6 \text{ ms.}$$

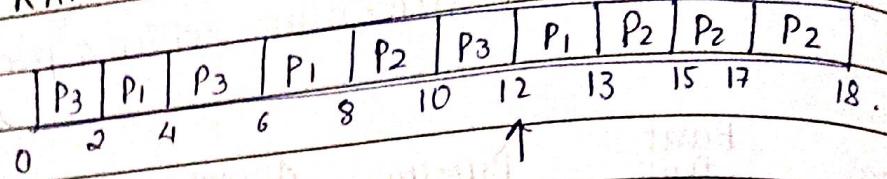
$$P_3 = 11 - 0 = 11 \text{ ms}$$

② Pre-emptive Priority based schedule.

P ₃	P ₁	P ₂	P ₃	
0	1	6	13	18

$$W.T. \rightarrow P_1 = 0 \quad P_2 = 1 \quad P_3 = 13 - 0 = 13 \text{ ms.}$$

③ RR.



$$W.T : P_1 = 12 - 4 - 1 = 7$$

$$P_2 = 17 - 6 - 5 = 6$$

$$P_3 = 10 - 4 - 0 = 6$$

$$\text{Avg. wt} = 6.33 \text{ ms}$$

$$TAT : P_1 = 13 - 1 = 12$$

$$P_2 = 18 - 5 = 13$$

$$P_3 = 12 - 0 = 12$$

$$\text{Avg. wt} : 12.33 \text{ ms}$$

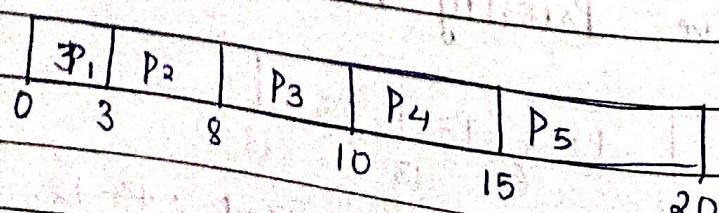
Q) Consider the following set of processes.

Process	Burst time	Arrival time	Priority
---------	------------	--------------	----------

P ₁	3	0	5
P ₂	5	1	3
P ₃	2	2	1
P ₄	5	3	2
P ₅	5	4	4

Calculate avg. waiting time & turn around time

II FCFS.



DATE: _____

PAGE NO.: _____

waiting time.

$$P_1 = 0$$

$$P_2 = (3-1) = 2$$

$$P_3 = (8-2) = 6$$

$$P_4 = (10-3) = 7$$

$$P_5 = (15-4) = 11$$

$$P_{avg} = \frac{0+2+6+7+11}{5} = 6.4$$

$$\text{Average waiting time} = \frac{0+2+6+7+11}{5}$$

$$\text{Turn around time} = \frac{\text{Completion time} - \text{Submission time}}{5}$$

$$P_1 = 3-0 = 3$$

$$P_2 = 10-2 = 8$$

$$P_3 = 8-1 = 7$$

$$P_4 = 10-2 = 8$$

$$P_5 = 15-3 = 12$$

$$\text{Avg. T.A.T} = \frac{(3+7+8+12+16)}{5}$$

5

$$9.2 \text{ ms}$$

DATE: 20/11/2023

PAGE No. 1

2) SJF.

P ₁	P ₃	P ₂	P ₄	P ₅
0	3	5	10	15

Giantt Diagram

waiting time for P₁ = 0

$$P_2 = (5-1) = 4$$

$$P_3 = (3-2) = 1$$

$$P_4 = (10-3) = 7$$

$$P_5 = (15-4) = 11$$

$$\text{Avg.} = \frac{0+4+1+7+11}{5} = 4.6 \text{ ms.}$$

Turn around,

$$P_1 = 3-0 = 3$$

$$P_2 = 10-1 = 9$$

$$P_3 = 5-2 = 3$$

$$P_4 = 15-3 = 12$$

$$P_5 = 20-4 = 16$$

$$\text{Avg. Turnaround time} = \frac{3+3+12+9+16}{5} = 8.6 \text{ ms}$$

$$= \frac{43}{5} - 8.6 \text{ ms}$$

DATE: / /

PAGE No.: 36

3) RR (Round Robin) (Pre-emptive)

P ₁	P ₂	P ₃	P ₄	P ₅	P ₁	P ₂	P ₄	P ₅	P ₂	P ₄	P ₅	
0	2	4	6	8	10	11	13	15	17	18	19	20

$$\text{Waiting time} = P_1 = (10 - 2) = 8$$

$$P_2 = (18 - 4) = 14 \text{ ms}$$

$$P_3 = (4 - 2) = 2 \text{ ms}$$

$$P_4 = (18 - 4 - 3) = 11 \text{ ms}$$

$$P_5 = (19 - 4 - 4) = 11 \text{ ms}$$

$$\text{Avg. wt time} = \frac{12 + 1 + 11 + 11 + 8}{5} = \frac{43}{5} = 8.6 \text{ ms}$$

$$\text{TAT} = P_1 = 2 + 11 - 0 = 11 \text{ ms}$$

$$P_2 = 18 - 1 = 17 \text{ ms}$$

$$P_3 = 6 - 2 = 4 \text{ ms}$$

$$P_4 = 19 - 3 = 16 \text{ ms}$$

$$P_5 = 20 - 4 = 16 \text{ ms}$$

$$\text{Avg. TAT time} = \frac{11 + 17 + 4 + 16 + 16}{5} = \frac{64}{5} = 12.8 \text{ ms}$$

4) Priority Based. (Pre-emptive)

P ₂	P ₄
0	2

P ₁	P ₂	P ₃	P ₂	P ₅	P ₁
0	3	5	10	15	20

Non-preemptive

P ₁	P ₃	P ₄	P ₂	P ₅
0	3	5	10	15

DATE: 22/2/22

PAGE NO:

Unit-3

Process Management & Synchronization

There are 2 types of process

- I/O-operative processes \rightarrow 1 printer shared by all comp.
- independent processes \rightarrow bank acc. in SBI & HDFC. (exams)

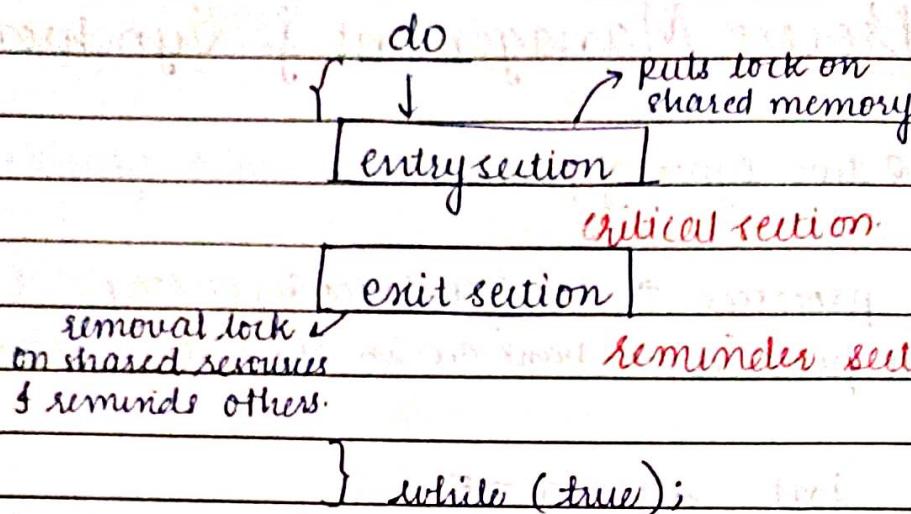
int shared = 5 {

P ₁	P ₂
int x = shared	int y = shared
x++	y--
sleep(1)	sleep(1)
shared = x	shared = y
Terminate.	Terminate

Raise condition

- In OS we have no. of processes & this processes require no. of processes resources.
- Now think of situation. we have 2 processes & this process reading same variable. they are reading & updating the value of variable & finally writing the data in memory.

DATE : 21/11/2023
PAGE NO. : 13



i) Mutual exclusion

If a process is in a critical section, then other processes shouldn't be allowed to enter i.e. there must be mutual exclusion between processes.

ii) Progress

It means that if one process doesn't need to execute into critical section then it shouldn't stop other processes to get into critical section.

iii) Bounded waiting

There must be some limit to no. of times a process can go into critical section i.e. there must be some upper bound.

If no upper bound is there, some process will be allowed to go critical section again and again.

DATE : _____ / _____ / _____

PAGE NO.: _____

& other processes will never get a chance to go into critical section.

24/02/24

Semphor

Semaphore.

wait(s) - decrement

, signal(s) - increment.

wait(s)

{

while ($s == 0$); // There is a 'j' here

$s = s - 1$; → $E \leftarrow E - 1$

} until condition becomes $\rightarrow (s > 0)$ conditions

$[0 < s]$

signal(s)

{

$E \leftarrow E + 1$

$s++;$

} operations below {

} remaining time

Types of semaphore

(1) Binar

(2) timer

(3) binomial

i) Binary semaphore

(4) timer

ii) Counting semaphore

(5) timer

(6) timer

(7) binomial

(8) timer

(9) binomial

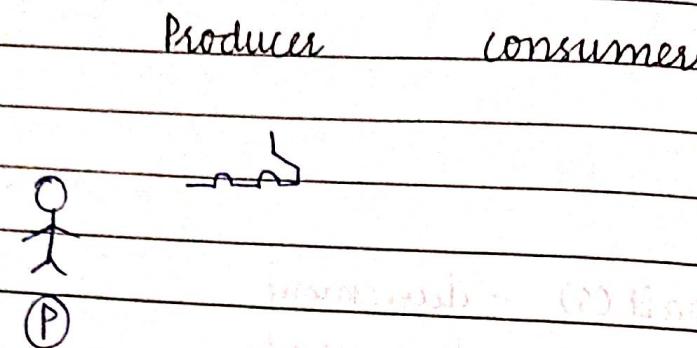
(10) timer

(11) binomial

DATE: / /

PAGE NO. /

Producers - Consumer Problem.



- multiple types of semaphore variables.
 - semaphore (s) → mutual exclusion,
 $s \rightarrow 0+1$ ← initially
 - semaphore (e) → empty space in buffer
 $e = n$
 - semaphore (f) =
space filled by producer
 $f \geq 0$

void producer ()

```
    {  
        while (T)  
        {  
            producer (e)  
            wait (e)  
            wait (s)  
            append ()  
            signal (s)  
            signal (f)  
        }  
    }
```

void consumer ()

```
    {  
        while (T)  
        {  
            wait (F)  
            wait (s)  
            appendtake ()  
            signal (s)  
            signal (e)  
            use ()  
        }  
    }
```

DATE: / /

PAGE No. /

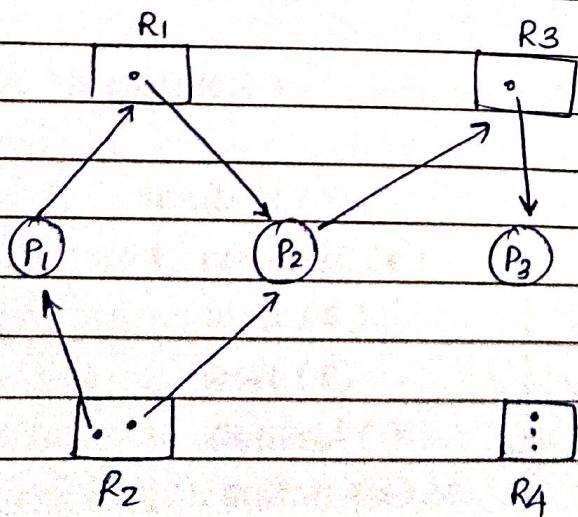
DATE: / /

Unit - 4 Deadlocks

- Deadlock is a situation where a set of processes are blocked because each process is allocated holding a resource & waiting for another processes resource acquired by some other process.
- The execution of two or more processes is blocked because each processes holds some resource while for another resource holds some other process.

Necessary conditions

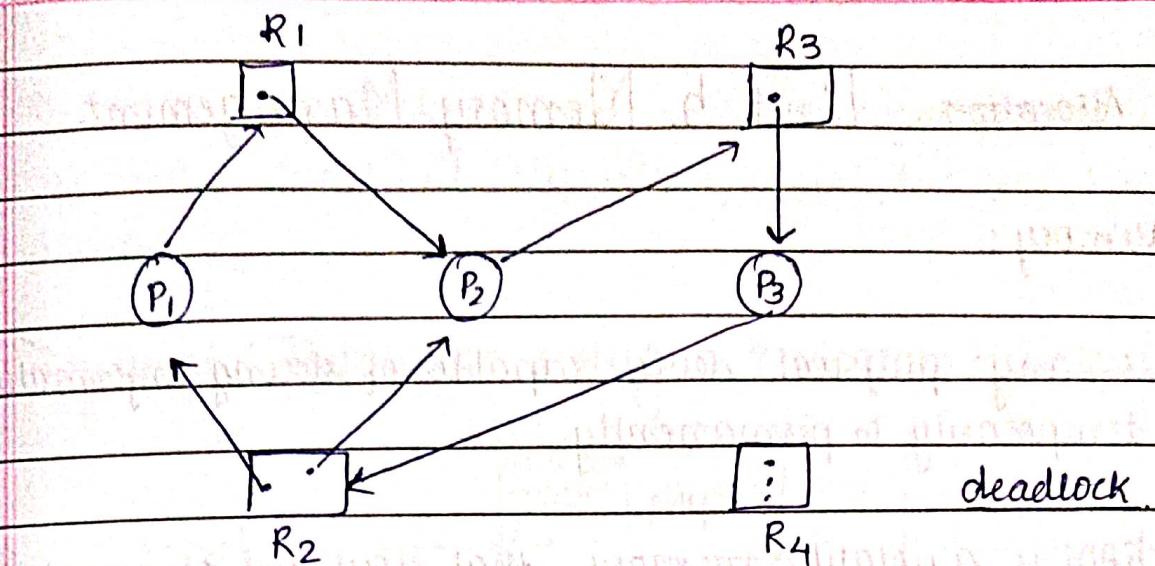
- Mutual exclusion
- No preemption
- Hold & wait
- Circular wait.



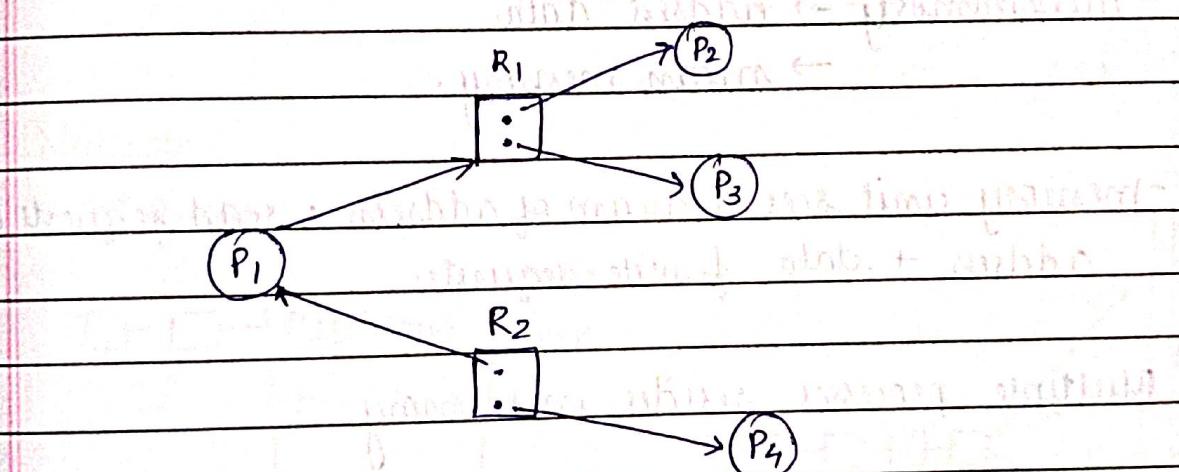
no deadlock.

DATE: _____

PAGE NO.: _____



Resource Allocation Graph with cycle but no deadlock.



Allocation Unit - 5. Memory Management

Memory:

- is any physical device capable of storing information temporarily or permanently.

- RAM is a volatile memory, that stores info on an integrated circuit used by the OS.

- cache memory → address, data
→ main memory.

- memory unit sees a stream of address + read requests
address + data + write requests.

Multiple processes resides in memory

- ensure correct operation.

1) Protect OS

2) protect user processes.

Base and limit Registers

2560000	Operating systems
300040	process
120940	process
8800000	process

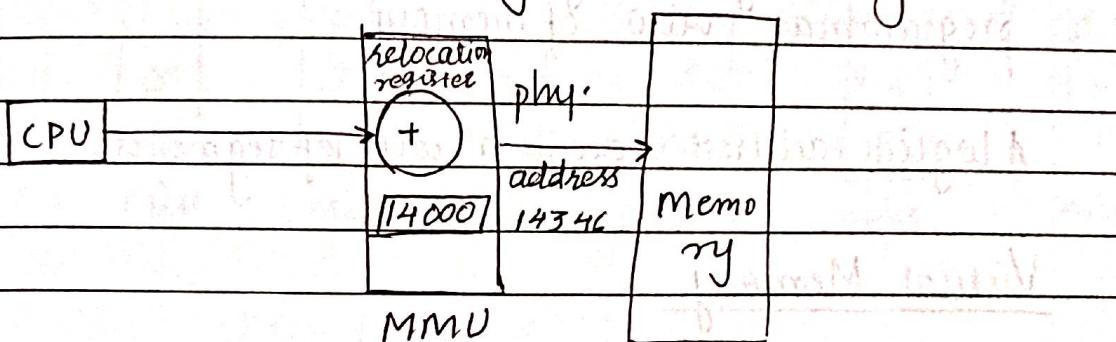
300040 (base)
120900 (limit)

DATE: _____

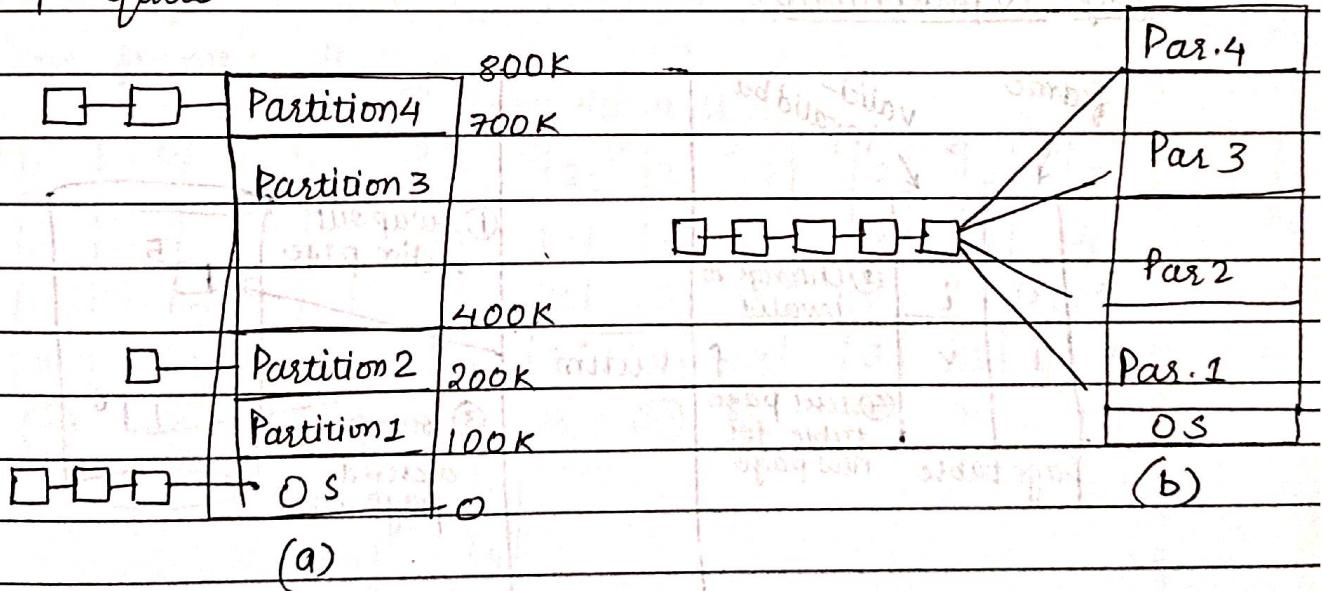
PAGE NO.: _____

The base register holds the smallest legal physical memory address; the limit register specifies the fix size of the range.

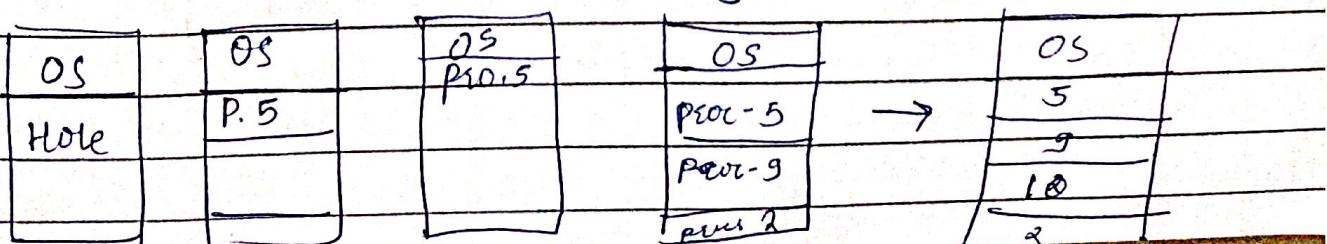
Dynamic Relocation using a relocation register



Multiple input queue



Hole: block of available memory; holes of various sizes are scattered throughout memory.



DATE: 20/9/2023

PAGE NO.

→ Fragmentation.

→ Segmentation.

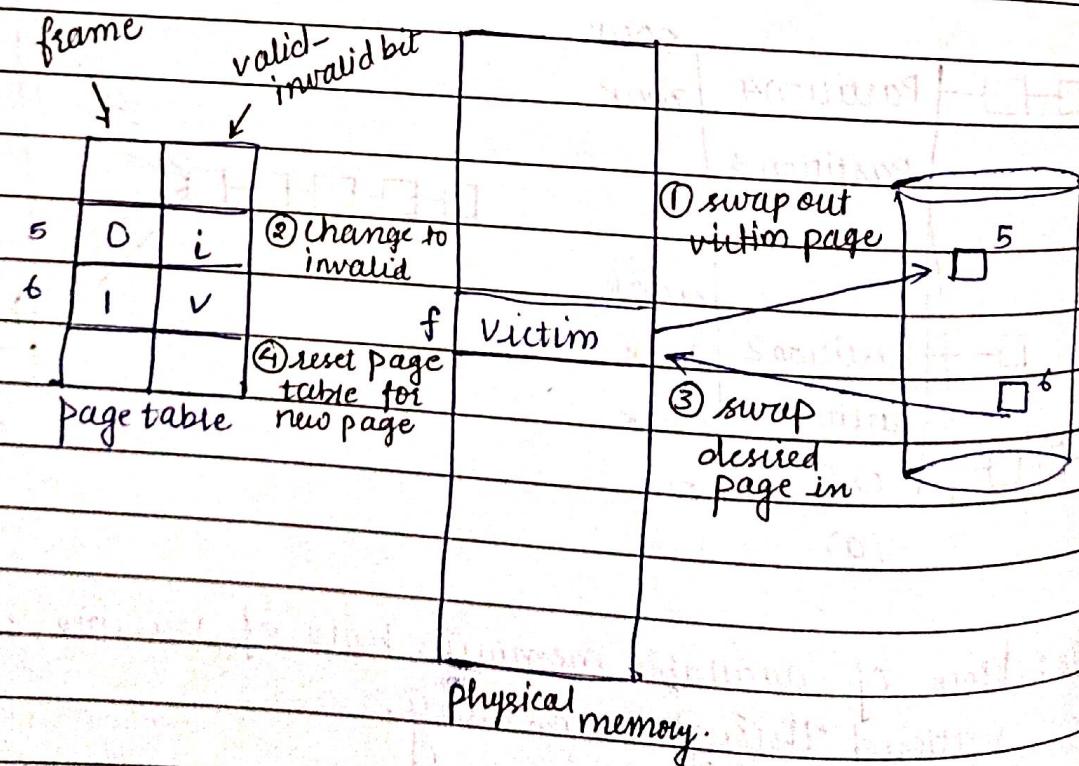
is a memory-management scheme that supports programmer view of memory.

A logical address space is a collection of segments.

Virtual Memory

→ TLB - Translation Lookaside Buffer.

Page Replacement



DATE : _____

PAGE NO : _____

Page Replacement Algorithms.

① FIFO

Page Ref: 1 3 0 3 5 6 5

Page frame: 5.

1		3	0	3	0	5	0	6	0	3	∅	3
		3	3	3	3		3	6	6		6	
1	1	1	1	1	1	5	5	5	5	5	5	

miss miss miss Hit miss miss miss miss

Page faults: 6

Hits 1.

Page ref: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3

Page frame: 4.

② Least Recently Used.

7		0	1	2	2	0	2	3	2	0	2	4	2	2	3	2	0	2	3	2
		1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	7	7	3	3	3	3	3	3	3	3	3	3	3	3	3	3

M M M M H M M H H H H H H

2 2 3 2

4 4

0 0

F → 6

3 3

H → 8

H H

DATE : / /

PAGE NO. :

DATE :

③ Optimal Page Replacement Algorithm

Page ref: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3

Page frames: 4

7	0	1	2	2	0	2	3	2	0	2	4	2	2	2
		1	1		1	1	1	1	1	1	4	1	4	4
	0	0	0		0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	7	3	3	3	3	3	3	3	3	3

M

M

M

M

H

M

H

M

H

4

4

m

OPR

4

3

2

3

2

3

2

4

4

4

4

4

0

0

0

0

0

3

3

3

3

3

H

H

H

H

H

P-f

fail

Page fault = 6

Page hit = 8

* This is efficient but not implementable.

Q. P.R - 4, 7, 6, 1, 7, 6, 1, 2, 7, 2

P.f : 4.

① FIFO

4		7	6	1	7	1	6	1	1	1	2	1	1	1
			6	6	6	6	6	6	6	6	6	6	6	6
4	4	7	4	7	7	7	7	7	7	7	7	7	7	7

m

m

m

m

H

H

H

m

H

Hit : 5

Miss - 5.

DATE: / /

PAGE No. :

LRU

09109

4	7	6	1	1	7	1	6	1	1	1	0	1	7	1	2	1
		6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2

m m m m H H H H H H m H H

OPR

4	7	6	1				6	1	1	1	2	1	7	1	2	1
		6	6		6		6	6	6	6	6	6	6	6	6	6
	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
4	4	4	4	4	4	4	4	4	4	4	4	4	4	2	2	2

m m m m H H H H H H m H H

P-f : 3

FIFO - Pf = 6

frame

LRU - Pf = 6

OPR - Pf = 5

① FIFO.

4	7	6	6	1	6	7	6	6	6	1	6	2	6	7	2	7
		7	7	7	7	7	7	7	7	7	7	7	7	7	2	2
4	4	4	4	1	1	1	1	1	1	1	1	1	1	1	1	1

m m m m H H H H H H m H H

miss = 6

hit = 4.

② LRU

4		7	6	6	1	6	7	6	6	1	6	2	6	7	7	2	7
4		7	7	7	7	7	7	7	7	7	7	7	2	2	1	1	1
4		4	4	4	1		1	1	1	1	1	m	m	m	m	H	
m	m	m	m	m		H	H	H	H	H	H	m	m	m	m	H	

Page fault = 6

Hit = 4

③ OPR.

4		7	6	6	1	6	7	6	6	1	6	2	7	2	2	2
4		7	7	7	7	7	7	7	7	7	7	7	H	7	7	7
4		4	4	4	1		1	1	1	1	1	m	m	m	m	1
m	m	m	m	m		H	H	H	H	H	H	m	m	m	m	H

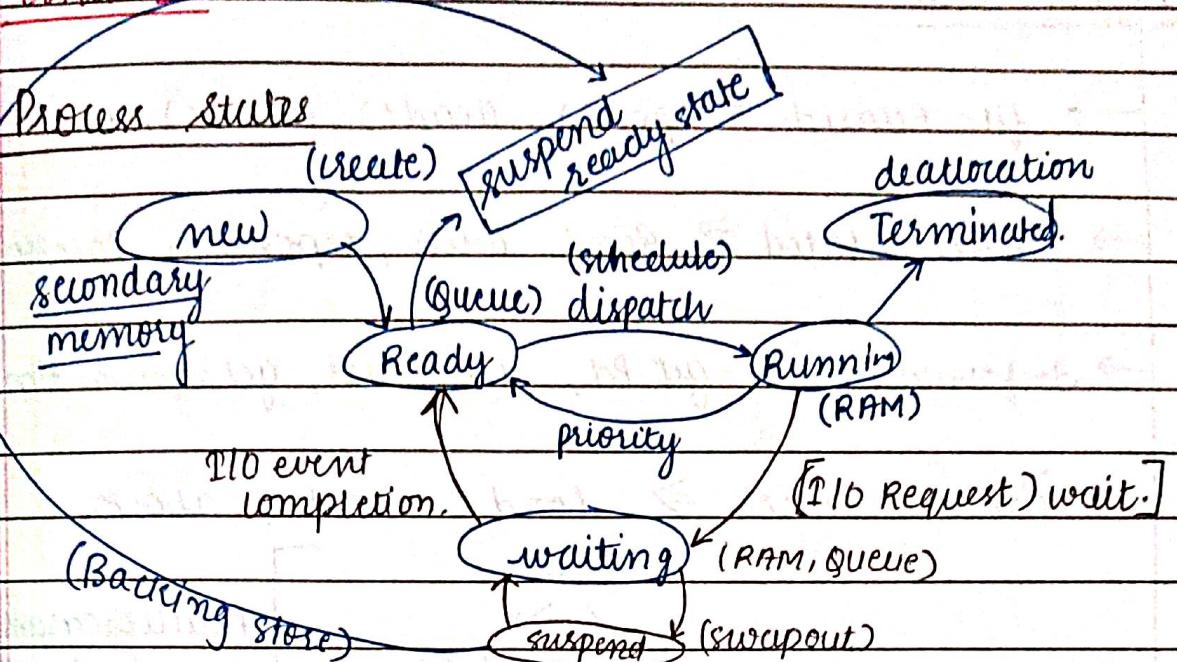
Page fault : 5

Hit = 5.

DATE: _____ / _____ / _____

PAGE NO.: _____

Unit - II



long term scheduler - maximum process brought in ready queue.

Short term scheduler - (ready state to running state)

Medium term scheduler - If when the waiting state is full, with I/O request of processes, then, the coming process is sent to (suspend state) to swapout. This is done by medium term scheduler.

- same with ready state.

DATE:

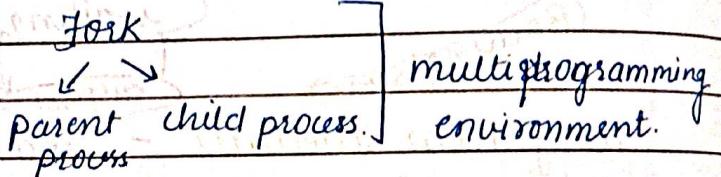
PAGE NO.:

DATE:

System Calls

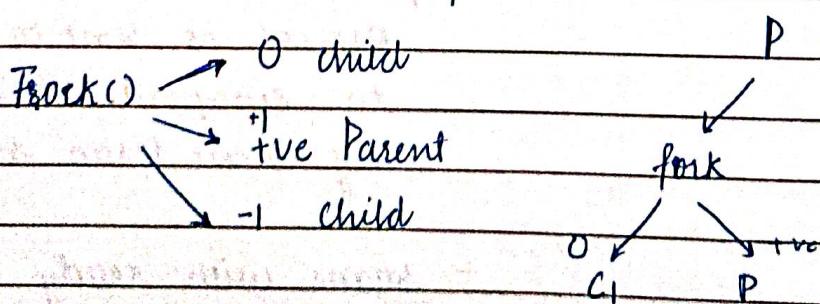
- file Related ⇒ open(), Read(), write(), close(), creat()
- device related ⇒ Read, write, reposition, ioctl, flock
- Information ⇒ get Pd, attributes, get system time/date
- Process control ⇒ load, execute, abort.
- Inter process communication - pipe(), create() / delete connection, shmemget().

wait, signal, allocate etc.



Fork()

- creates a child process.



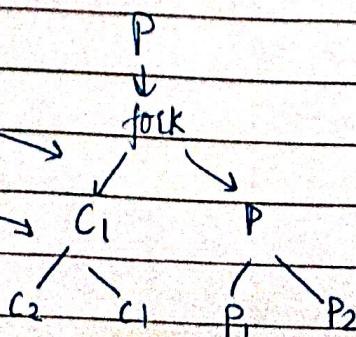
main() {

 fork();

 fork();

 printf("Hello");

}



DATE: ___ / ___ / ___

PAGE No.: ___

no. of parent execution $2^n \leftarrow$ (no. of time fork written).
child process $2^n - 1$

