Complete notes on

Operating System

Copyrighted By:-

Instagram:- coders.world

Telegram:- codersworld1

Written By:-

Sanskriti Satpute

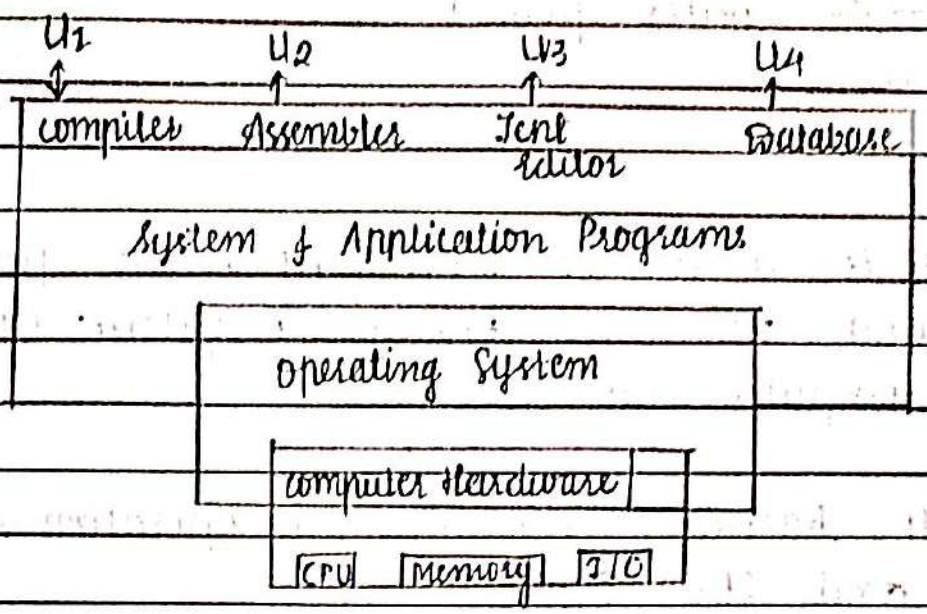
(Software engineer)

@coders.world
Full Stack Developer

@coders.world
Full Stack Developer

DATE : / /

OPERATING SYSTEMS



Structure of computer System.

- Computer systems can be divided into types.

1] Hardware - It provides basic compiling resources. (CPU | memory | IO) 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)

@coders.world
Full Stack Developer

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 ~~operetings 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 & data processing performed at hardware level using interprocess communication & system call.

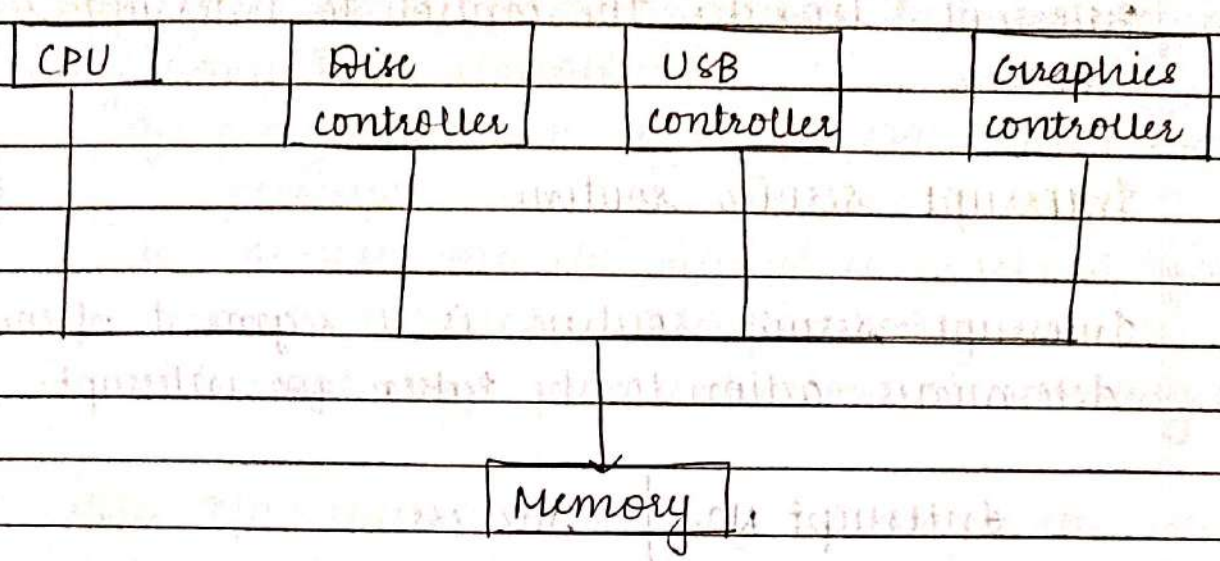
Bootstrap Program.

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

@coders.world
Full Stack Developer

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 incharge 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 finish its operation by causing an interrupt.

@coders.world
Full Stack Developer

DATE: __/__/__ PAGE No.:

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 interrupts; which then polls all devices to figure out the reason for interrupt.

2] Interrupt vector Table

Different interrupt handler will be executed for diff interrupts

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : ____

I/O Structure

1] After I/O starts, control returns to user programs only upon I/O completion

a] wait instruction idols 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.

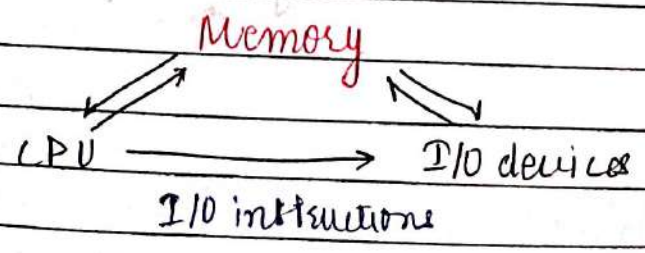
System call - request to O.S to allow user to wait for I/O completion.

Device status Table

a] contains entry of each I/O device indicating its type address & state.

b] OS indexes into I/O devices table to determine device status & to modify table entry to include the interrupt.

Direct Memory Access.



@coders.world
Full Stack Developer

DATE: __/__/__ PAGE No:

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.

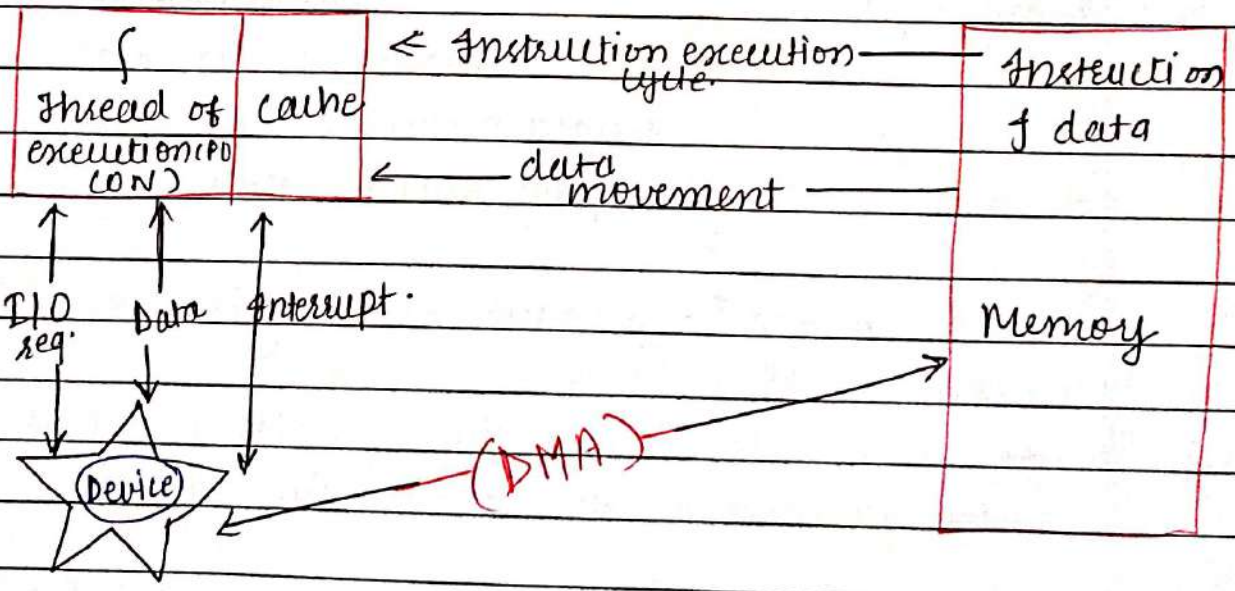
Registers
↑
cache memory
↑
main memory
↑
SSDs ← electronic disc
↑
magnetic disc
↑
optical disc
↑
magnetic tapes
↑

@coders.world
Full Stack Developer

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.



@coders.world
Full Stack Developer

Kernal entry points

→ what are diff event in OS.

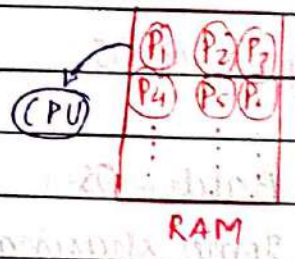
In computing environment event in an action or occurence detected by the program that may be handled by the program.

System call

20/01/22.

Types of operating system.

1] Multiprogrammed OS



→ Non premptive
→ 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 question, will execute the remaining after.

→ not idell

→ But response time is decreased.

@coders.world
Full Stack Developer

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 IOS, Andro

Types of OS.

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

Goals of OS

i) convenience
ii) Efficiency
iii) Both.

→ Functions.

- It is an interference betn User & Hardware
- Alloration of Resources
→ Management of memory, security etc.

@coders.world
Full Stack Developer

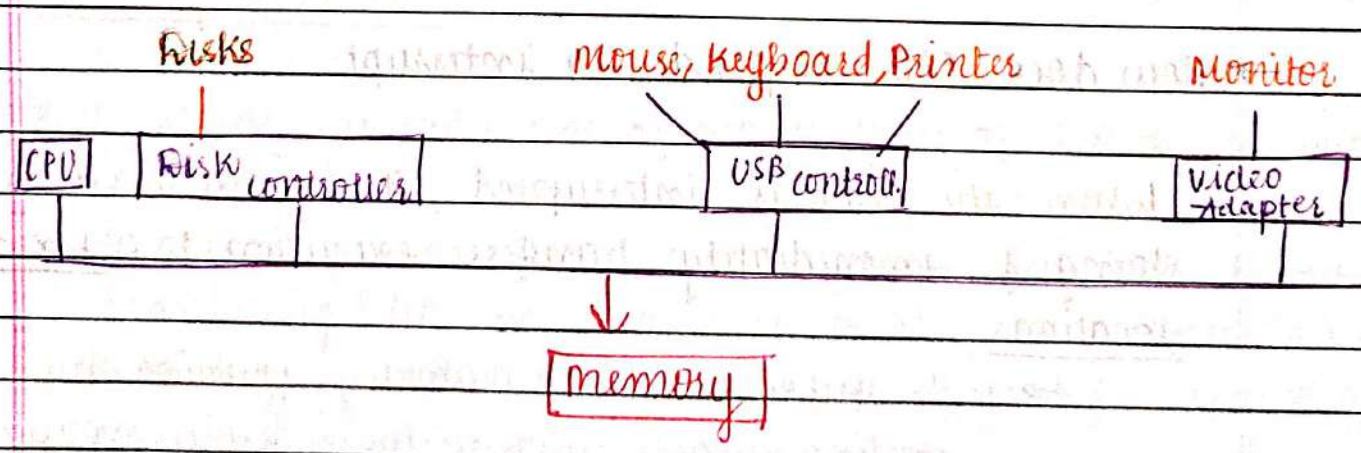
Basic of operating system (computer system operation)

→ 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 devices 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.



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 mem
- It must know how to load the OS

@coders.world
Full Stack Developer

- and start executing that system.
- It must locate and load into memory the OS kernel.
  (heart of OS)

2] Interrupt

- The occurence 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 the way of system bus.

3] System call (monitor call)

- Software may trigger by executing a special operation 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 compl^n, the CPU resumes the interrupted computation

@coders.world
Full Stack Developer

Basic of OS [storage structure]

☆ special case (NVRAM → non volatile)

- Expensive but fast
- smaller size

↑

Registers
↓ ↓
Cache
↓ ↑
Main memory
↓ ↑
Electronic disks
↓ ↑
Magnetic disks
↓ ↑
Optical disks
↓ ↑
Magnetic Tapes.

↓

- cost per bit ↑
- access time ↑
- larger size

VOLATILE → loses its content power is removed.

NON-VOLATILE → Retains its contents even when power is removed.

(I/O Structure)

→ A storage is only one of many types of I/O devices within a computer.

→ A large portion of O.S. code is ~~decid~~ dedicated to ~~mang~~ managing I/O both because of its importance to the realibility 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

@coders.world
Full Stack Developer

DATE : __/__/__

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

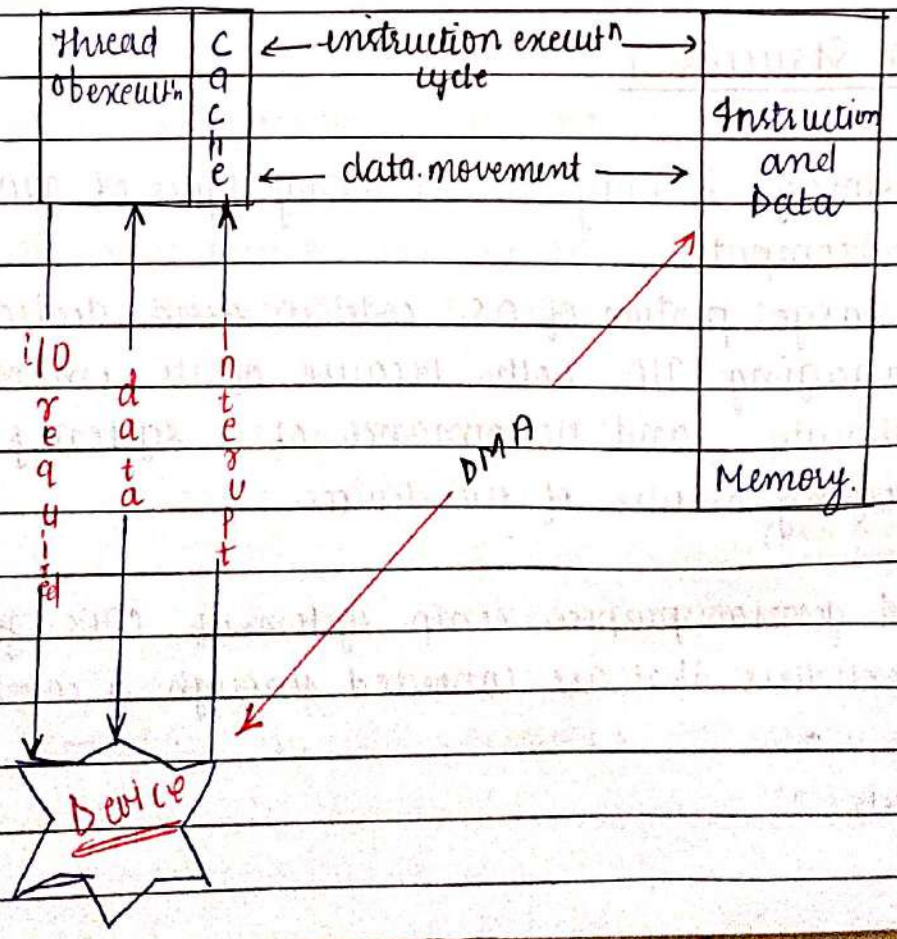
maintains

- 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



@coders.world
Full Stack Developer

DATE : / / PAGE No :

→ To start an I/O operation, the device driver loads the appropriate registers within the device controllers

→ 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 forms 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 transfer 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 accomplish other work.

@coders.world
Full Stack Developer

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
(micro processor chip in keyword)

2) Multi Processor System / Parallel or tightly coupled sys

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

Adv.

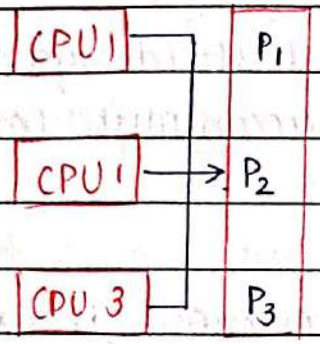
→ increased throughput (Measure of perform. of system
→ economy of scale. (as here sharing of resources)
→ increased reliability.

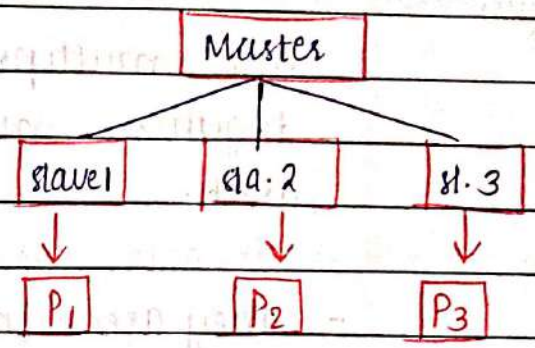
@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : ____

→ Types of multiprocessor system.

- Symmetric
- Asymtric Multiprocessing





→ all the processors are similar to each other

→ commonly used

→ all processors share physical memory.

Master → slave approx.

The master ~~alloca~~ processor schedules & allocate the work to the worker processers

★ → many processes can run simultaneously (NCPUS → N Processes)

★ → one CPU sits idle while other overload

→ Multiprocessing adds CPUs to increase computing power.

→ 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.

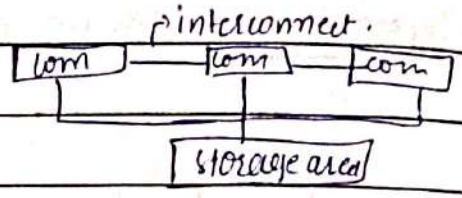
★ All multicore sys → multiprocessors (★not vice versa).

@coders.world
Full Stack Developer

DATE : / / PAGE No :

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

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 → multiple allows host to access the same data on shared storage

Clustering over WAN

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : ____

Operating System Structures

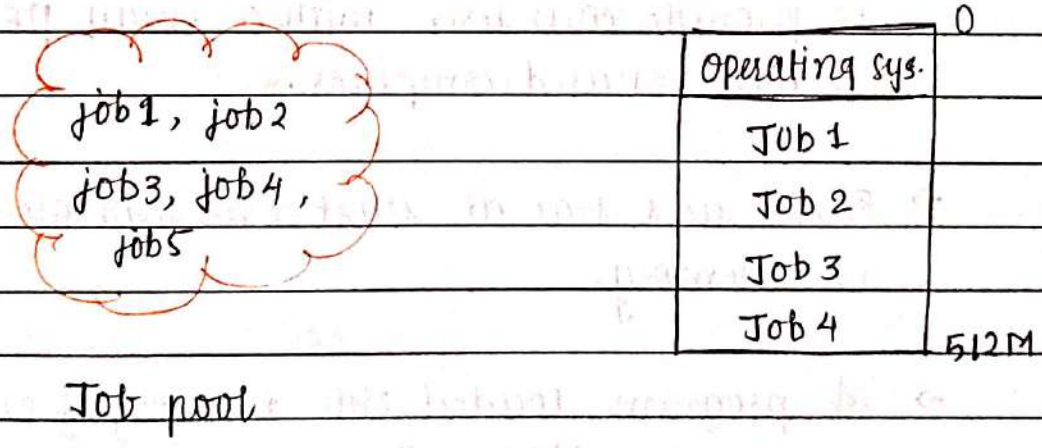
→ (Multiprogramming & Multitasking) → commonalities

→ Operating system vary greatly in their makeup internally

i) Multiprogramming.

→ 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.



→ 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.

@coders.world
Full Stack Developer

DATE : __/__/__

ii) Time Sharing (Multitasking)

→ CPU executes multiple jobs by switching among them (virtual memory)

→ Switches occur so frequently that the users 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 & 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 seperate 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.

@coders.world
Full Stack Developer

DATE : ___/___/___ PAGE No. :

ii) Time Sharing (Multitasking)

→ CPU executes multiple jobs by switching among them (virtual memory)

→ Switches occur so frequently that the users 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 & 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 seperate 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.

@coders.world
Full Stack Developer

DATE : ___/___/___ PAGE No : _____

i) User Interface.

→ interaction betn users & computer
ex- command line, graphic

ii) Program Execution

source code → compiler → objectcode → executor → output

iii) I/O operations
Ex- Keyboards → CPU → monitor

iv) File system manipulation.
create, delete, modify, search, access restriction of file.

v) Communications

vi) error detection

vii) Resource Allocation

viii) Accounting

ix) Protection & Security.

@coders.world
Full Stack Developer

DATE : / / PAGE No :

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 Interference 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 ~~form~~ 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

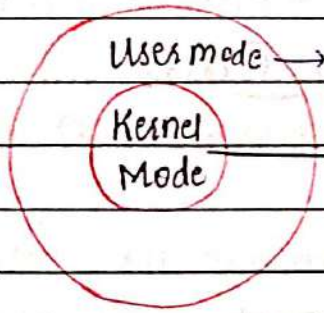
Second app
command interpreter calls the program where code is written to create file.

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : ____

System Calls

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



(safer mode)

Uses mode → (doesn't have access to memory, hardware resources)

Kernel Mode → priveleged mode

- Direct access
- program is executed in kernel mode crashes down, whole system would crash.

- System call is a programming way in which a computer program requests a service from the kernel of the OS

- These calls are general available as routines written in C and C++.

Ex:-

Source file → Destination file

> Acquire input filename
> write prompt to screen
> Access input
> Acquire output filename
> write prompt to screen
> Accept input
> Open Input file.
> If file does'nt (ABORT)

→ create output file
if file exists. ABORT
~~Read from input file~~

@coders.world
Full Stack Developer

loop
Read the input file.
write to output file
until Read fails
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

@coders.world
Full Stack Developer

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 ~~po~~ program developement and execution.

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No :

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

i) File management

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

ii) Status Information

Ask the system for-

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

iii) 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.

@coders.world
Full Stack Developer

iv) Programming - Language Support

- compilers
- assemblers
- Debuggers &
- 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 program provide mechanism for:-
i) Creating virtual connc among processes, users & com. sys.
ii) Allowing users to send messages to one → another comp.
iii) To browse webpages
iv) To send electronic-mail messages
v) To log in remotely or to transfer files from one → another.

@coders.world
Full Stack Developer

DATE: __/__/__ PAGE No:

Operating System Design & Implementation

- Design goals.

1st problem : Defining goals & specifications

- choice of hardware
- Type of system.

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

Requirements

- Uses 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)

@coders.world
Full Stack Developer

* one imp. principle that, the seperation 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 operated on many devices.

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

Structure of OS.

Simple structure

Application program
↓
Resident Sys. Program
↓
Device drivers
↓
ROM BIOS devicedrivers

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No :

Monolithic Structure. → (UNIX)

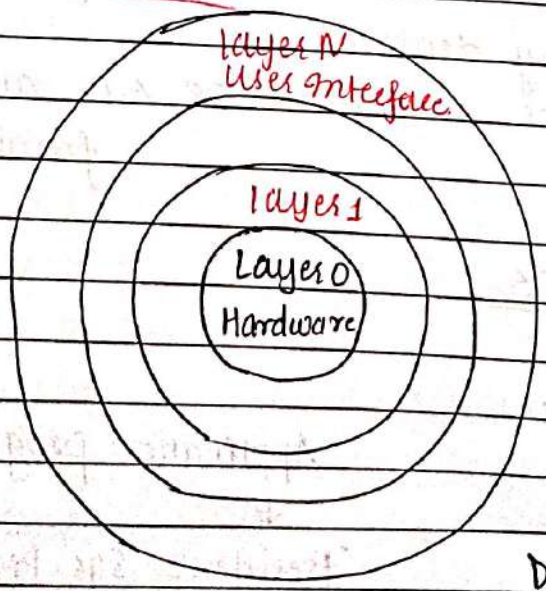
The users
Shells and commands
compilers and interpreters
system libraries

Kernel

System-call interface to the Kernel		
signal, terminal handling	file system	CPU scheduling
character, I/O sys.	swapping block I/O	page replacement
terminal devices	system	demand paging
	disk & tape drivers	virtual memory
Kernel interface to the hardware		
terminal controllers	device controllers	memory controllers
terminals	disk & tapes	physical memory

diff. maintain & implemented.

Layered structure.



Adv

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

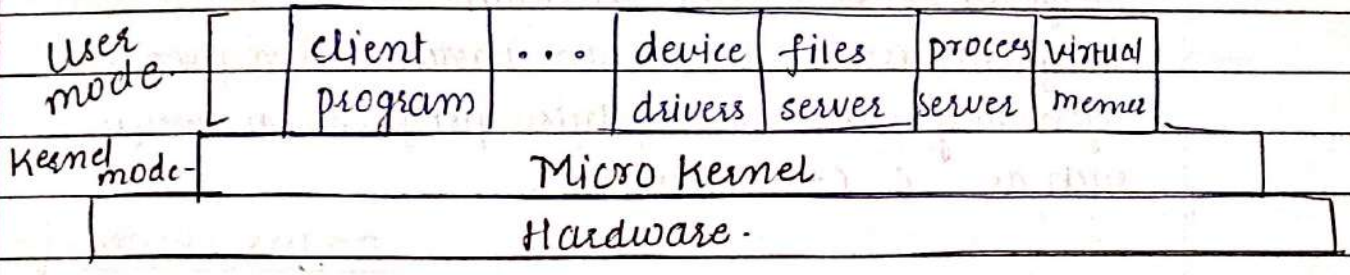
Disadv

Deciding the position of layer as layer can use layer below it

@coders.world
Full Stack Developer

→ not every efficient
→ as it has go layer by layer one by one.

Microkernels



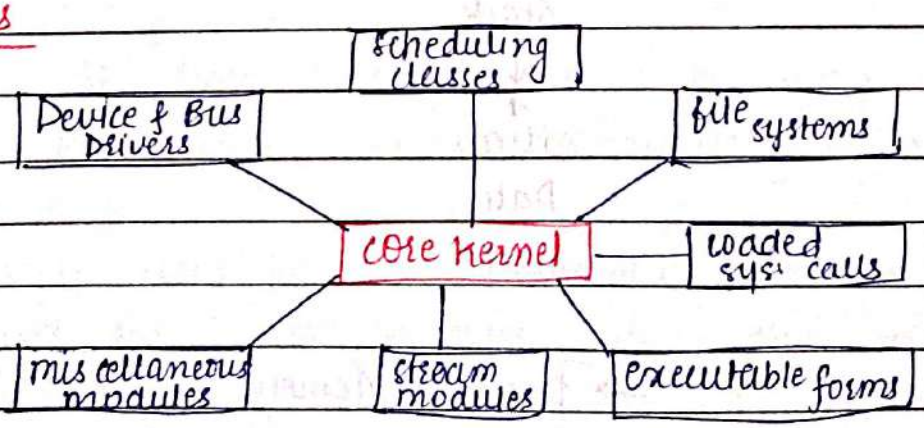
Adv

- To Remove non-essential component from system. (only core imp) 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 → By far The Best.
→ loaded directly, no system overhead.

[threads, multireading model, goals of rescheduling & dith, cont

@coders.world
Full Stack Developer

DATE: 27/01/22

Unit - 2 Process & It's Scheduling

2.1] Process

- Program ~~is a~~ under execution.
- Program is consider as an instance of a program
- generally users writes their program in specific lan 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 activi active entity when it loads into memory.
- Several process may be associated with same progra eg. opening up several instances of

Stack
↓
↑
Heap
Data
Text

→ Process in Memory

@coders.world
Full Stack Developer

Four sections are -

i) Text

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

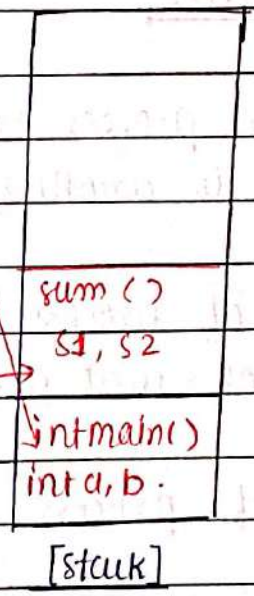
ii) Stack section.

- Stack memory hold temporary data. such as fun', 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 & would go to main memory.

- 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.

@coders.world
Full Stack Developer

DATE : __/__/__

PAGE No.

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 etc.

2.2) Process States

- During the process execution, it changes it's 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 it's current status

- Each process may be in one of the foll^n state.

@coders.world
Full Stack Developer

DATE : ___/___/___ PAGE No : _____

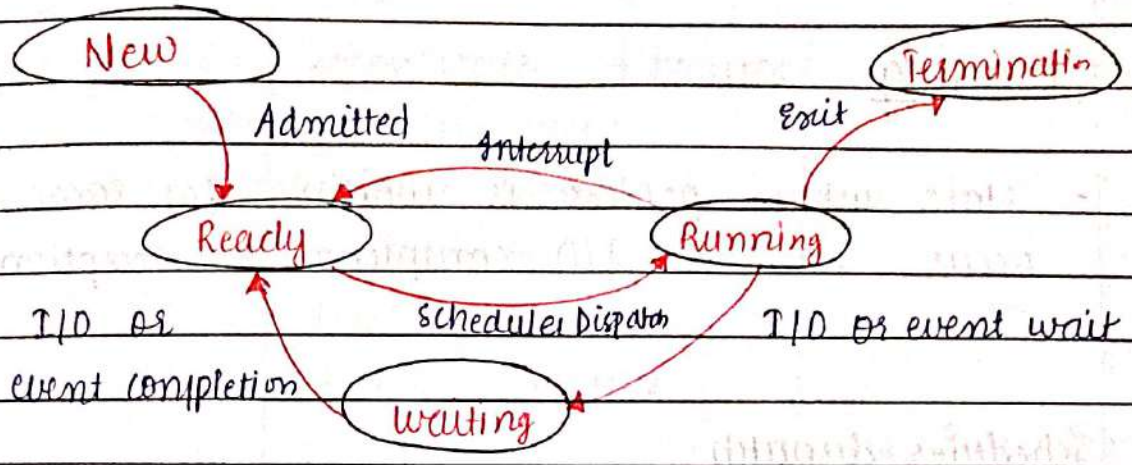


Fig: Process state

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 ~~exter~~ execution of process is carried out. From running state process may go to one of following state.

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

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No :

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.

@coders.world
Full Stack Developer

DATE : / / PAGE No :

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

Fig: Process Control Block

i) Process State

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

ii) Program counter

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

iii) CPU register.

The registers vary in number and type of depending on computer architecture.
It includes accumulators, index registers, stack pointers and general purpose registers.

iv) Process Number

It represents identifer of the process

@coders.world
Full Stack Developer

DATE : / / PAGE No :

v] Memory management information.

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

vi] Pointer

It include pointer to next PCB that is pointer to 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] 3. Process Scheduling

- Process scheduling is the technique adopted by OS which governs the order in which process need to be executed.
- 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.

@coders.world
Full Stack Developer

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. It consists of all processes in the system.

2] 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.

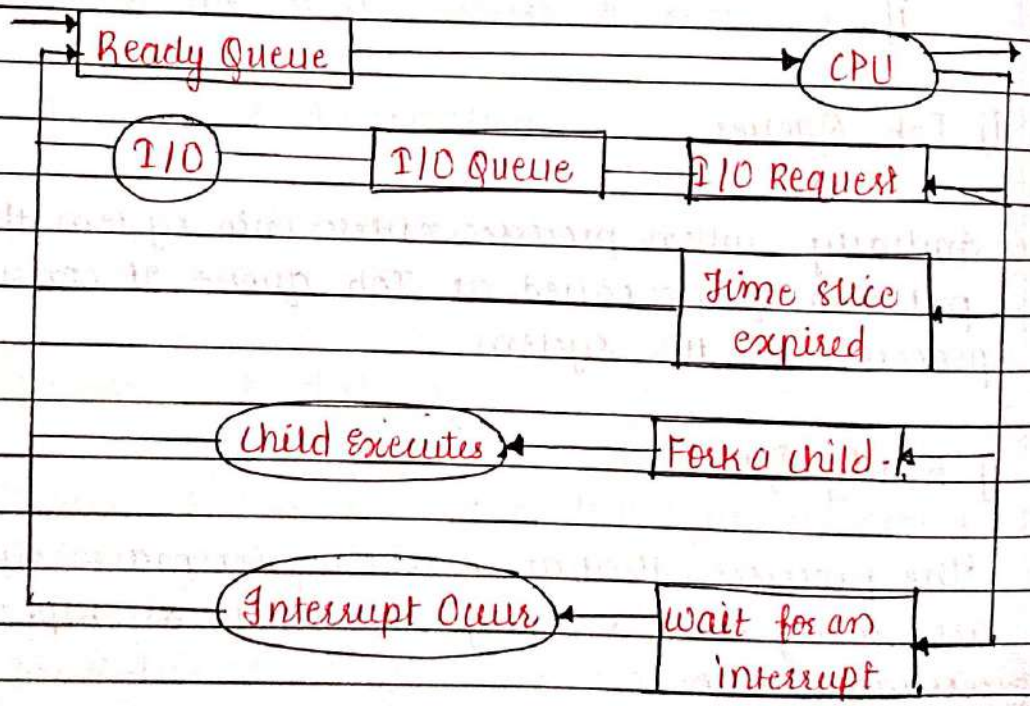
@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No :

iii) Device queue.

The list of processes waiting for a particular I/O devices 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.

@coders.world
Full Stack Developer

Schedulers.

- Schedular is an entity responsible for arranging the processes in proper order and then submitting it to CPU for execution.

- Schedular arranges the processes into an approximate seq.

- As process migrates among various scheduling queues throughout it is life time, this task of migration is carried out by scheduler.

- Scheduler in an intermediate level of scheduling.

- There are diff types of scheduler

i] Long Term Schedulers

- a.k a Job scheduler.
- responsible for selecting the process from Job queue to ready queue
- Simply the scheduler pick the process from secondary memory and load it into primary memory.
- works as a gate keeper
- Also called as admission schedular.
- 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 schedular executes less frequently.

@coders.world
Full Stack Developer

ii) Short Term Scheduler or CPU 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 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) Meduim 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

@coders.world
Full Stack Developer

ii) Shortest Job First Scheduling.

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

Ex. 3)

Process	Arrival time	Burst time
P_1	0	8
P_2	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_1 = final exe. time − Previous time − 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

↳ P.T.O

@coders.world
Full Stack Developer

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

P_2	P_5	P_1	P_3	P_4

0 1 6 16 18 19

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

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

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

→ Round - Robin Algorithm.

→ Pre-emptive

Process	Burst time
P_1	24
P_2	3
P_3	3.

Time quantum : 4ms

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : _____

P1	P2	P3	P1	P1	P1	P1	P1

0 4 7 10 14 18 22 26 30

$P_1 = 26 - 20 = 6$
$P_2 = 4$
$P_3 = 7$
} W.T.

Avg. wt → $\frac{17}{3}$ = 5.66 ms.

Multi-level Queue Scheduling.

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

~~Groups~~ →

- Two types of processes are ~~foregro~~.
  1) foreground (interactive processes)
  2) Background (Batch processes)

→ [system processing es] → highest Priority processing es

→ [Interaction processing es]

→ [Interactive editing processing es]

→ [Batch processing es]

→ [~~Student~~ student Process.] → lowest Priority.

@coders.world
Full Stack Developer

→ Process once entered an queue, can't switch the queue

Multilevel Feedback Queue Scheduling

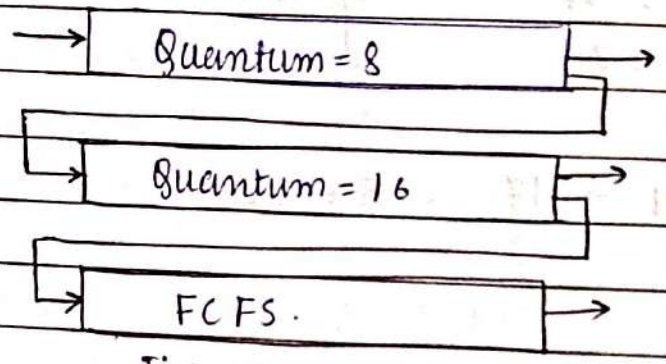


Fig: MFQS

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

i) If characterstics of job change betn CPU intensive and I/O intensive.

ii) Job has waited for long time can get bumped into higher priority queue. ~~Multi~~

iii) Multilevel feed Q is most flexibles 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

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : ____

c) methode to upgrade / ~~degrade~~ demote process from one que → anoth.
d) method used to determine which a process enters initially

Ex-1

Process	Burst time	Priority	Arrival time
P_1	5	1	1
P_2	7	2	5
P_3	6	3	0

Timequant - 2ms.

① SJF (pre-empitive)

P_3	P_1	P_3	P_2

0 1 6 11 18

W.T = P_1 = 1 - 1 = 0
P_2 = 11 - 5 = 6
P_3 = ~~11 - 0 = 11~~ 6 - 1 - 0 = 5.

Avg. wt = 11/3 = 3.66 ms

TAT = P_1 = 6 - 1 = 5
P_2 = 18 - 5 = 13
P_3 = 11 - 0 = 11

Avg. TAT = 29/3 = 9.6 ms.

② Pre-emptive Priority based schedule.

P_3	P_1	P_2	P_3

0 1 6 13 18

W.T → P_1 = 0 P_2 = 1 P_3 = 13 - 0 = ~~13~~ 12.

@coders.world
Full Stack Developer

③ RR.

P3	P1	P3	P1	P2	P3	P1	P2	P2	P2

0 2 4 6 8 10 12 13 15 17 18.

W.T : $P_1 = 12-4-1 = 7$
$P_2 = 17-6-5 = 6$
$P_3 = 10-4-0 = 6.$

Avg. wt = 6.33 ms.

TAT : $P_1 = 13-1 = 12$
$P_2 : 18-5 = 13$
$P_3 = 12-0 = 12$

Avg. wt : 12.33 ms.

2] 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 & ~~a~~ turn around time.

1] FCFS.

P1	P2	P3	P4	P5

0 3 8 10 15 20

@coders.world
Full Stack Developer

DATE : / / PAGE No :

waiting time.

$P_1 = 0$
$P_2 = (3-1) = 2$
$P_3 = (8-2) = 6$
$P_4 = (10-3) = 7$
$P_5 = (15-4) = 11$

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

$= \underline{5.2\,ms}$

Turn around time = completion (time) - submission time.

$P_1 = 3-0 = 3$
$P_3 = 10-2 = 8$
$P_2 = 8-1 = 7$
$P_3 = 10-2 = 8$
$P_4 = 15-3 = 12$

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

$= \underline{9.2\,ms}$

@coders.world
Full Stack Developer

DATE: / /

PAGE No.:

2] SJF.

P_1	P_3	P_2	P_4	P_5

0 3 5 10 15 20

Gantt Diagram

waiting time for $P_1 = 0$

$P_2 = (5-1) = 4$

$P_3 = (3-2) = 1$

$P_4 = (10-3) = 7$

$P_5 = (15-4) = 11$

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

Turn around,

$P_1 = 3-0 = 3$

$P_2 = 10-1 = 9$

$P_3 = 5-2 = 3$

$P_4 = 15-3 = 12$

$P_5 = 20-4 = 16$.

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

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

@coders world
Full Stack Developer

DATE : __/__/__ PAGE No. :

3] RR (slide) (Pre-emp)

P1	P2	P3	P4	P5	P1	P2	P4	P5	P2	P4	P5

0 2 4 6 8 10 11 13 15 17 18 19 20

waiting time $P_1 = (10-2) = 8$

~~$P_2 = (18-4)$~~ $P_2 = (17-4-1) = 12ms$

$P_3 = (4-3) = 2ms$

$P_4 = (18-4-3) = 11ms$

$P_5 = (19-4-4) = 11ms.$

Avg. wt time $= \frac{12+1+11+11+8}{5} = \frac{43}{5} = 8.8\ ms$

TAT = $P_1 = 11-0 = 11ms$

$P_2 = 18-1 = 17ms$

$P_3 = 6-2 = 4ms$

$P_4 = 19-3 = 16ms$

$P_5 = 20-4 = 16ms$

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

4] Priority Based. (pre-emptive)

~~P3~~	~~P4~~

0 2 7

P1	P2	P3	P2	P5	P1

0 1 2 4 13 18 20

Non-preemptive

P1	P3	P4	P2	P5

0 3 5 10 15 20

@coders.world
Full Stack Developer

DATE : 22 / 2 / 22 PAGE No :

Unit-3

Process Management & Synchronization

There are 2 two types of process

- co-operative processes → 1 printer shared by all comp.
- independent processes → 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

Race condition

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

@coders.world
Full Stack Developer

DATE: / / PAGE No:

5M | 7M

do

puts lock on shared memory

[entry section]

critical section

[exit section]

removal lock on shared resources & reminds others.

reminder section

} while (true);

i] Mutual exclusion

If a process in a critical section, then other processes shouldn't be allowed to enter. i.e there must be m.e betn processes.

ii] Progress

It means that if one process doesn't needs 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, same process will be allowed to go critical section again & again

@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : _____

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

24/02/24

~~Simphoi~~

Semaphore.

wait (S) - decreement
signal (S) - increement.

```
wait (S)
{
    while (S==0); // There is a ';' here
        S--;
}

signal (S)
{
    S++;
}
```

Types of semaphore

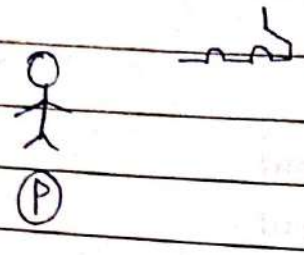
i] Binary semaphore
ii] Counting semaphore

@coders.world
Full Stack Developer

DATE : / /

Produces- Consumer Problem.

Producer consumer



- multipe types of semaphore variables.
  - semaphore (S) → mutual exclusion, $\boxed{S \to 0/1}$ ← initially
  - semaphore (E) → empty space in buffer $\boxed{E = n}$
  - semaphore (F) = space filled by producer $\boxed{F \Rightarrow 0}$

```
void producer ()
{
    while (T)
    { producer (E)
      wait (E)
      wait (S)
      append ()
      signal (S)
      signal (F)
    }
}
```

```
void consumer()
{
    while (T)
    { wait (F)
      wait (S)
      ~~appenc~~ take ()
      signal (S)
      signal (E)
      use ()
    }
}
```

@coders.world
Full Stack Developer

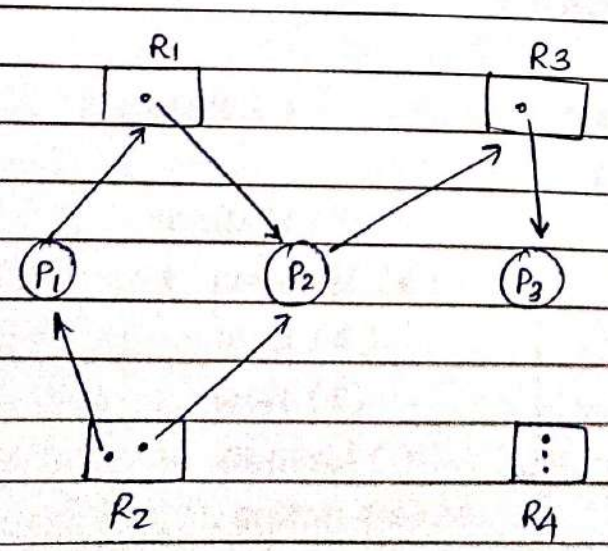
Unit-4 Deadlocks

- 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 & waits for another resource holds some other process.

Necessary conditions

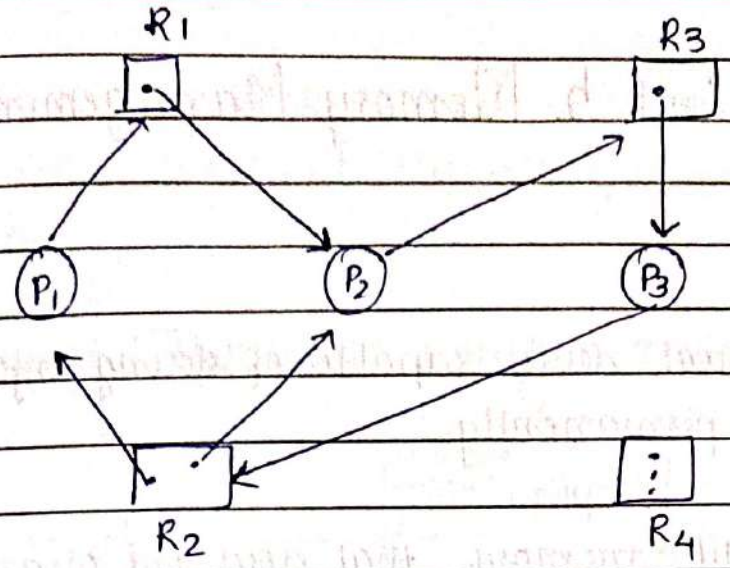
→ Mutual Exclusion
→ No preemption
→ Hold & wait
→ circular wait.



no deadlock.

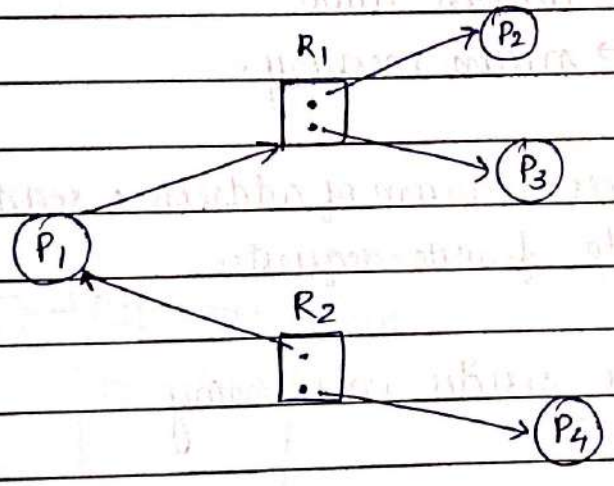
@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : ____



deadlock.

Resource Allocation Graph with cycle but no deadlock.



@coders.world
Full Stack Developer

~~Allocation~~ Unit-5. Memory Management

Memory:

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

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

- cache memory → address, data
  → main memory.

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

Multiple processes resides in memory

- ensure correct operation.

1) Protect OS 2) protect users processes.

Base and limit Registers

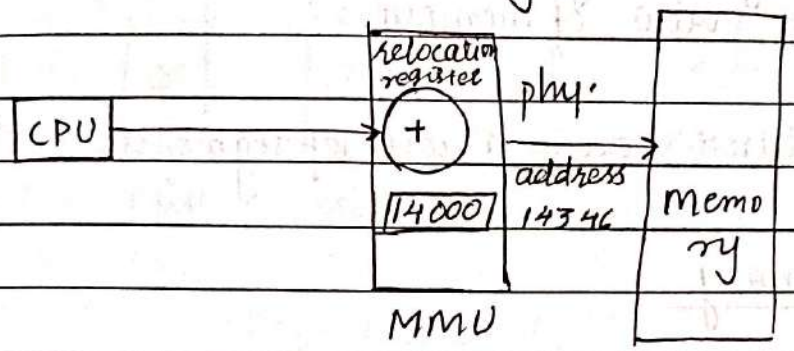
0		
2560000	Operating systems	
300040	process	300040 (base)
420940	process	120900 (limit)
880000	process	

@coders.world
Full Stack Developer

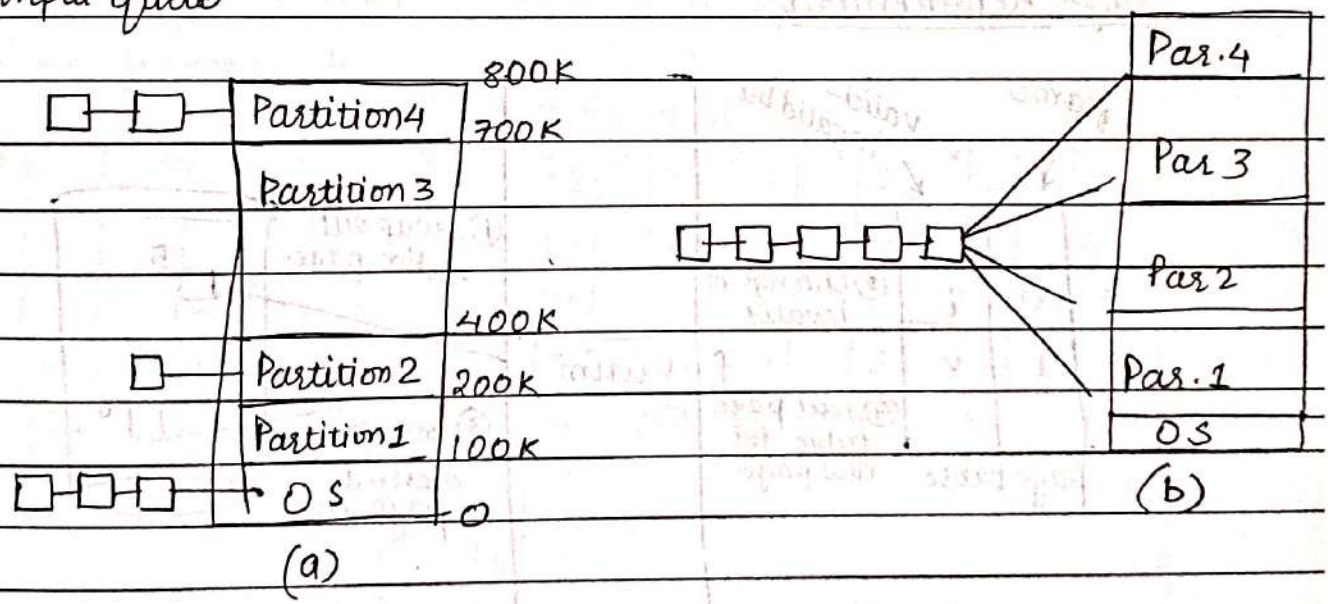
DATE : __/__/__ PAGE No : ____

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

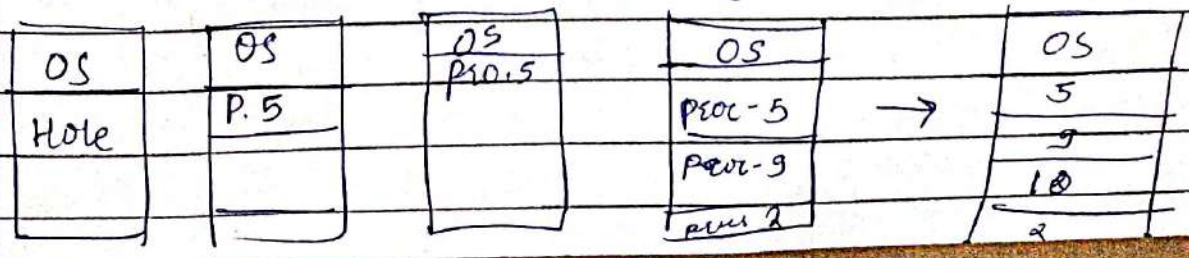
Dynamic Relocation using a relocation register



Multiple input queue



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



@coders.world
Full Stack Developer

→ Fragmentation.

→ Segmentation.

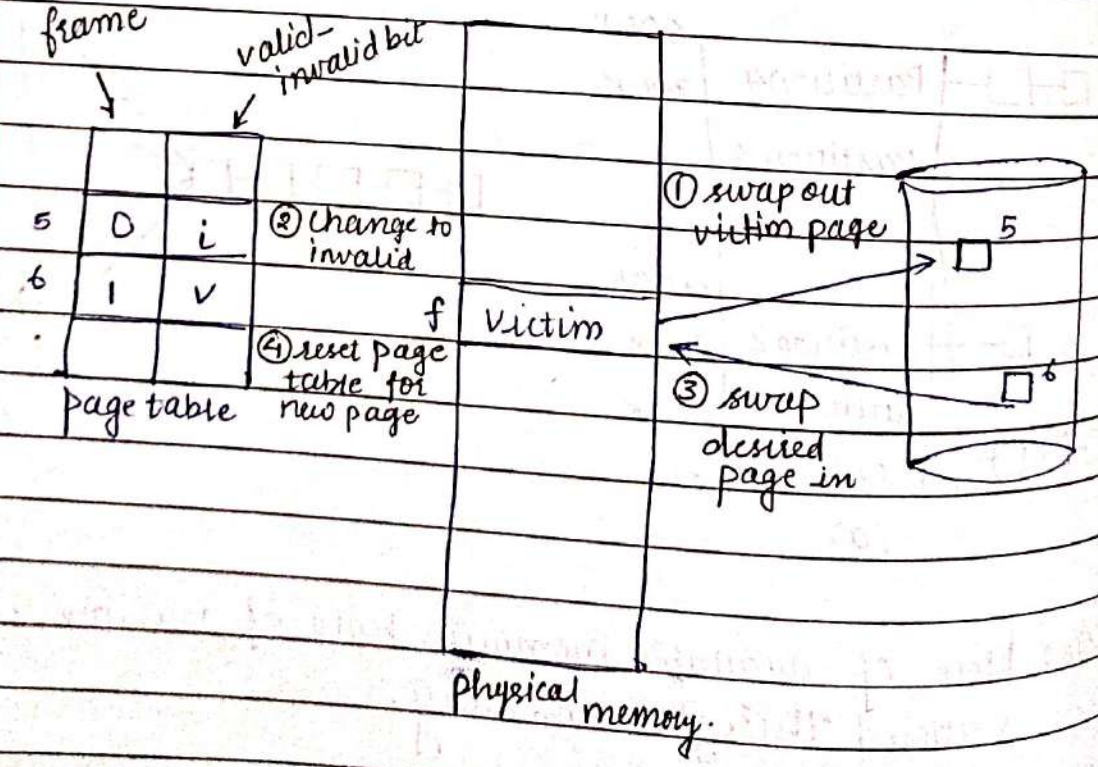
is a memory-management scheme that supports programmar view of memory.

A logical address space is a coll^n of segments.

Vistual Memory

→ TLB - Translation Lookaside Buffer.

Page Replacement



@coders.world
Full Stack Developer

DATE : / / PAGE No :

Page Replacement Algorithms.

① FIFO

Page Ref : 1 3 0 3 5 6 3

Page frame : 5.

1	3	0	3	5	6	3
		0	0	0	0	~~0~~ 3
	3	3	3	3	~~3~~ 6	6
1	1	1	1	~~1~~ 5	5	5
miss	miss	miss	Hit	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 Recent Used.

7	0	1	2	0	3	0	4	2	3	0	3	2	3
			2	2	2	2	2	2	2	2	2	2	2
		1	1	1	1	1	~~1~~ 4	4	4	4	4	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	H	H	H		

2	3
2	2
4	4
0	0
3	3
H	H

F → 6

H → 8

@coders.world
Full Stack Developer

DATE : / / PAGE No :

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	0	3	0	4	2	3	0	3	2	3
				2	2	2	2	2	2	2	2	2	2	2
			1	1	1	1	1	1	4	4	4	4	4	4
		0	0	0	0	0	0	0	0	0	0	0	0	0
	7	7	7	7	7	3	3	3	3	3	3	3	3	3
	M	M	M	M	H	M	H	M	H	H	H	H	H	H

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	6	1	2	7	2
				1	1	1	1	1	1	
			6	6	6	6	6	6	6	
		7	7	7	7	7	7	7	7	
	4	4	4	4	4	4	4	2	2	
	M	M	M	M	H	H	H	M	H	

Hit : 5 Miss - 5.

@coders.world
Full Stack Developer

DATE : / / PAGE No :

LRU

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

OPR

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

Miss = 6

Hit = 4.

@coders.world
Full Stack Developer

② LRU

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

Page fault = 6

Hit = 4

③ OPR.

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

Page fault : 5

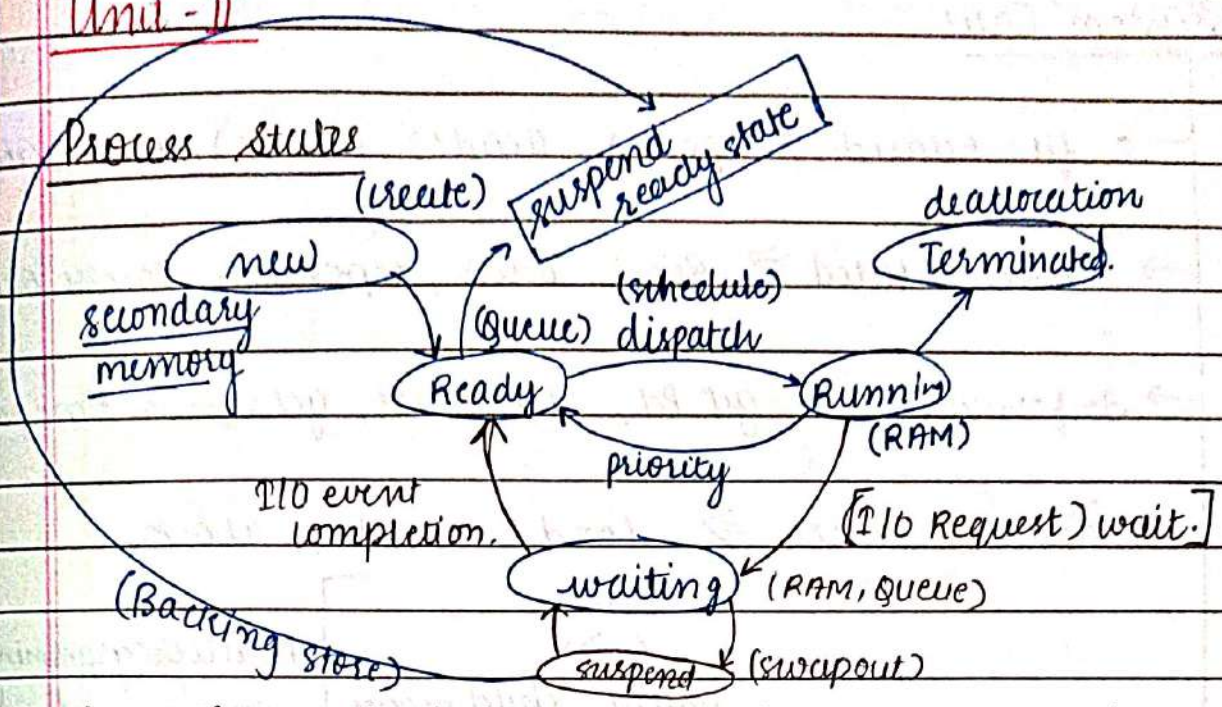
Hit = 5.

@coders.world
Full Stack Developer

DATE : / / PAGE No :

Unit - II

Process States



long term scheduler - maximum process bought 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 medi term scheduler.

- same with ready state.

@coders.world
Full Stack Developer

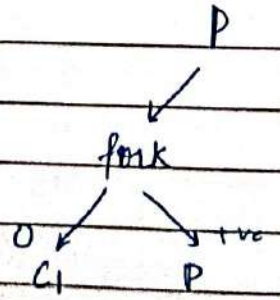
System Calls

- → file Related ⇒ open(), Read(), write(), close(), createfile
- → device related ⇒ Read, write, reposition, Pointer, Ioctl
- → Information ⇒ get Pid, attributes, get system time/date
- → Process control ⇒ load, execute, abort,
  - fork → parent process, child process.] multiprogramming environment.
  - wait, signal, allocate etc.
- → Inter process communication - pipe(), create() / delete connection, shemget().

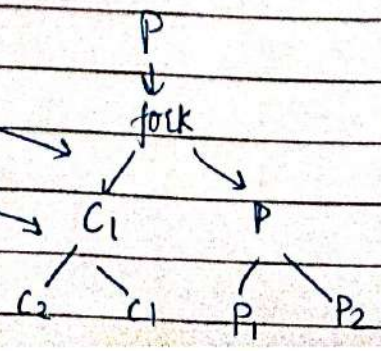
Fork()

- creates a child process.

Fork() → 0 child
→ +1 +ve Parent
→ -1 child

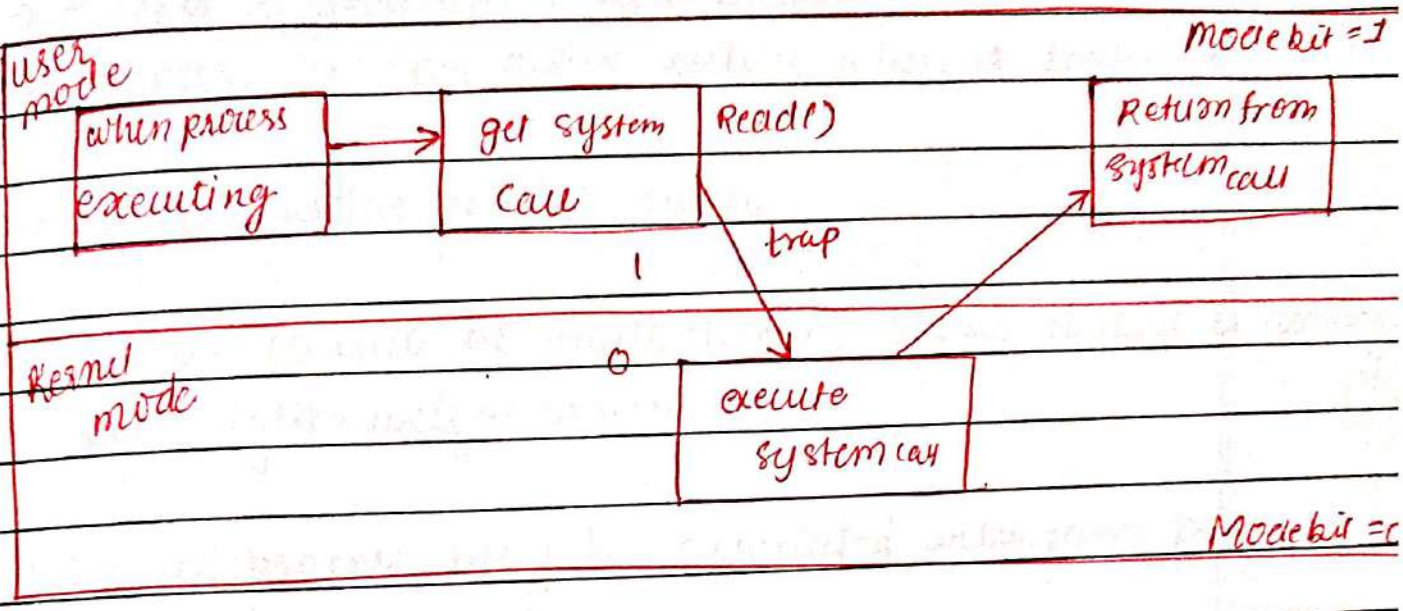



```
main() {
    fork();
    fork();
    printf("hello");
}
```



@coders.world
Full Stack Developer

DATE : __/__/__ PAGE No : ____

no. of ~~parents~~ execution 2^n ← (no. of time fork written).

child process $2^n - 1$