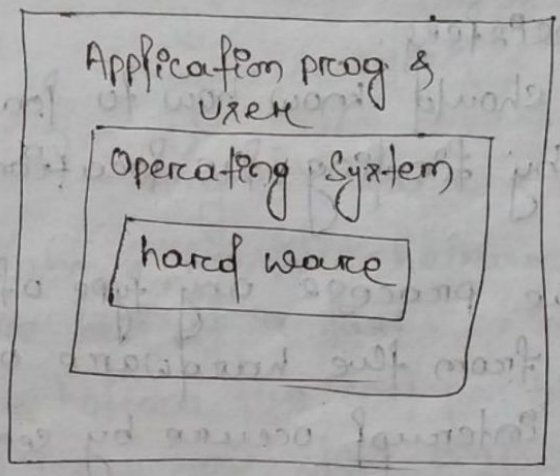
UNIT-1

Operating System:

Operating System is a system software which is used to manage the system hardwares and also it acts as a intermediate between user and system. And it provides a platform to run the application programs



There are two view points of an operating system

(1) User view

(2) System view

User view

From the user point of view the O.S should be designed in a way which is easy to use and gives better performance.

System view

From the system point of view the operating System is designed to manage the hardware and to allocate the resources in a better way.

The OS to control the execution of a program by preventing it from errors.

Computer System organisation:

(1) Computer system operation:

- When a computer starts it has a initial program to run called as bootstrap program it is stored in ROM or EEPROM.
- This program initializes all the memory contents as well as cpu registers.
- This program should know how to load the OS and to execute it by finding the location of the OS in memory.
- In between the process any type of interrupt may occure either from the hardware or from software.
- The hardware interrupt occure by sending a signal to cpu.
- The software interrupt occure by sending a system call or monitor call.
- If a interrupt occurs then the cpu transfers it control to the memory location of the interrupt. After handling the interrupt then it resume the execution of previous process/job.

(2) Memory/Storage structure:

- All the computer programs mainly stored in main memory or RAM.
- The interaction between memory & register done by using 2 instruction.

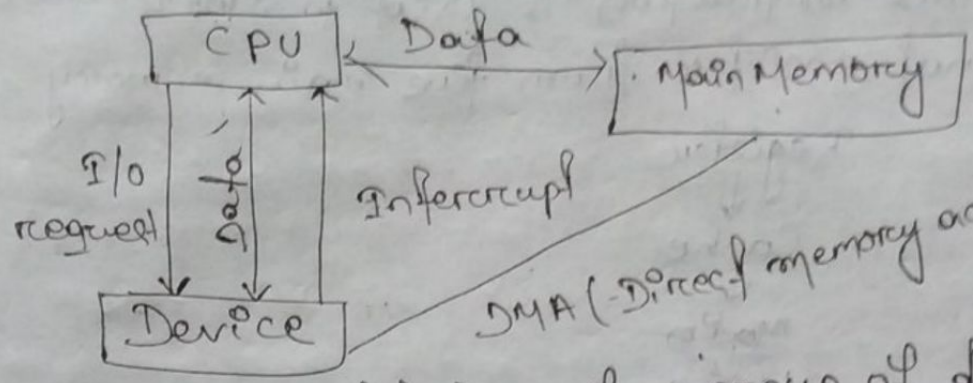
(1) Load - This instruction is used to move a data from main memory to internal register within cpu.

(2) Store - This instruction is used to store the data from internal registers from the main memory.

Register
↓
Cache
↓
Main
Electric
↓
Magnetic disk
↓
Magnetic tape

In the above memory heirarchy the higher levels are expensive but it is fast.
- Towards the bottom the cost per bit generally decreases
- The main memory have the following demerits due to which all the programs and data couldn't be stored in the main memory
(1) The size is small
(2) It is volatile in nature.

(3) I/O Structure:
- In a computer system the CPUs and device controllers connected to each other by using a common bus.
- Each device controller is in the charge of a specific type of device.
- Operating system has a device drive for each device controller.
- The device controller is responsible for transforming on moving data between peripheral devices and its storage.



DMA is used to transfer group of data bulk of data.

Computer System Architecture:

It involves different ways to organize a computer system by using a no. of general purpose processors.

(1) Single Processor System:

This type of system contains one main CPU which controls the execution of different instruction sets.

- This type of system execute or perform multiple processes by using some scheduling algorithms.

(2) Multi Processor System:

This type of system is otherwise called as parallel system. These are interlinked to each other by bus.

These have some advantages over single processor system.

(1) Single throughput:

This means if we increase the no of processor then we can execute more no. of process in less time.

(2) Economy of scale:

A multiprocessor system cost less than multiple single processor system.

(3) Increased reliability:

This means the failure of single processor can't halt the total system.

Levels of OS

- Interface
- Resource allocator
- Manager.

Types of OS

- Batch OS
- Multiprogramming
- Multitasking
- Multiprocessor
- Realtime

Clustered System:

Clustered system are the combination of more than one system coupled together.

These systems are linked via a local area network. These systems operate in a server-client method.

These are of 2 types.

(1) Symmetric

(2) Asymmetric

Symmetric: These type of systems have two servers and both the servers controls all the clients & all also monitors each other.

Asymmetric: In this between two servers one is called as hot stand by mode server that means this server monitors other server.

Operating System Structure:

An OS provides the environment for program execution one of the important aspect is multiprogramming of OS.

MULTIPROGRAMMING:

Multiprogramming is either the CPU or I/O devices are busy for all the time.

- It increases the CPU utilization as it provides one job has to be executed all the time.
- All the jobs of a system is stored in a job pool then the job from the job pool is transmitted to memory by using job scheduling technique because the memory size is smaller as compare to job pool.
- After storing the job in main memory the system choose one job from them for execution by using process scheduling technique/CPU scheduling.

MULTITASKING

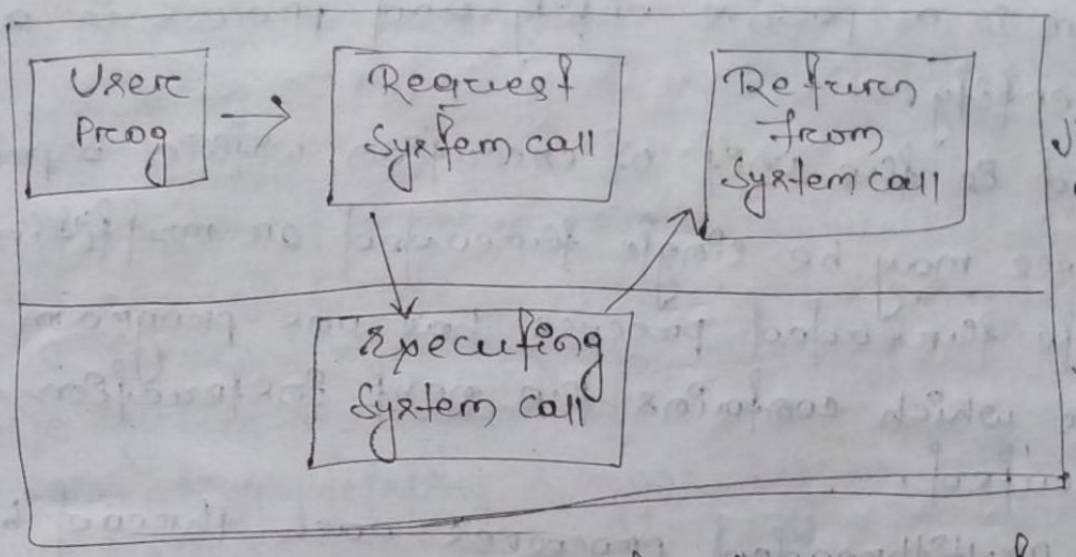
- This type of os is otherwise called as time sharing os.
- In these systems CPU executes multiple jobs by switching among them by sharing the CPU time.

Operating System Operation:

There are 2 types of OS operation.

↳ User Mode

↳ Kernel mode/System mode



The mode bit indicates the correct mode of the operation system i.e 0 for kernel and 1 for user mode.

- User mode is activated when the system executes a user applications.
- When user request service from the operating system then it makes the system call & the mode changes from user mode to system mode.
- The switching between the mode is done by the processor. This dual mode operation helps protecting operating system for outside user.

TIMER:

When a program runs for an infinite time or there is no particular time of program termination to overcome this problem we can use a timer.

Process Management:

Process is a program which is under execution.

- The process is divided into thread to give better performance.

- Program is a passive entity and process is an active entity.
- Thread is the unit of execution within a process.
- A process may be single threaded or multithreaded
- A single threaded process has one program counter which contains the next instruction to be executed.
- In multithreaded processes each thread has multiple program counter and each contains next instruction to be executed on a given thread.

Operating System activities in process management:

- Creating, deleting, suspension and resume the user and system process.
- Process synchronization ~~communication mec~~ deadlock handling, process communication mechanism
- Mechanisms are provided by operating system.

Memory Management:

Activities perform during memory management:

- It keeps track about which parts of memory is currently in use and by whom.
- Allocating and deallocating the memory spaces.
- It decide which process will move into and out from the memory.

Storage Management:

A system uses a logical storage unit and OS access this files by using different storage devices.

(1) FILE SYSTEM MANAGEMENT:

File is a collection of related data which has stored by the user.

- File represent programs and data for both source as well as object code.
- OS is responsible for the following activities during file system management.
  - Creating, deleting & organizing as well as the directories.
  - It also helps during the mapping of files into secondary storage.
  - Backup & recovery from non-volatile storage media on secondary storage devices.

(2) MASS-STORAGE MANAGEMENT:

- It involves the storage unit except the main memory.
- OS performs the following activities.
- Free space management, storage allocation then disk scheduling.
- There are some tertiary storage devices like magnetic disk, CD, DVD. and they uses WORM (Write once read many time) and read. ~~for~~ write format.

(3) CACHING:

- Cache memory is a fast storage system and use to store the data & info in temperary basis.
- In a single processor syetem there is a single cache memory.
- In a multiprocessor environment each of the CPU contains local cache memory.
- As all the CPUs have to execute concorently so in some cases we have to update a value which present in different caches and that is called as cache coherency.

(4) I/O SYSTEM:

On some cases operating system has to hide some features of specific hardwares devices from the user.

- I/o sub system consist of several component.
  (i) Memory management.
  (ii) Device driver interface.
  (iii) Drivers for specific hardware

(5) PROTECTION & SECURITY:

- Protection is a mechanism used to control the access of resources from the processes & user.
- Security is used to protect a system from internal & external attack.
- Most of the ~~user~~ OS maintains user name & user identifiers.

Distributed System:

A distributed system is a collection of separable heterogeneous through a network.

- Network is a collection of systems connected to each other by a communication link to establish an inter communication facilities.

There are different types of networks according to the distance between the system.

LAN: Local Area Network

This type of network connects the computers within a building or room.

MAN: Metropolitan Area Network

This types of network connects the computers from building to another building within a city.

WAN: Wide Area Network

This type of network connects the computers

world wide.

There is a software operating system which help to share files across the network and exchange file data between different processes and different computers.

Special-Purpose System :

(1) Real-Time Embedded system :

- This type of system are used for special purposes like robotics, microwares.
- Embedded systems are those in which the hardware part is controlled by a software & both are present in a single system.
- Real-time systems are those which have used with in a specific time limit to perform a operation.

(2) Multimedia System :

- Multimedia system processes multimedia data like vedio & audio file. DVD players.

(3) Hand-held system :

It includes PDAs (Personal Digital Assistants) like cellphones, palmtops, pocket PCs.

- This type of system have some issues like physical memory, speed processor input output services.
- There is some features like probo portability increase the use of handel systems.

Computing Environment:

Computing Environment are those in which we have to use or implement different types of system.

- Traditional
- Client-Server
- Peer-to-peer
- Web-based

Traditional

In this type of computing env. we use old versions of systems and the web based approach enhance its use.

Ex: UNIX, windows xp, batch operating system, interactive.

Client-Server

It is divided into compute server system & file server systems.

- In this type of system client can send a request to the server & the server gives a respond to the client to complete the action.
- In this type of system the clients can create read write update, delete the file in the server.

Peer-to-peer

In this type of env. there is no client server concept. In this approach each system is connected to each other by a network. If you want to add new peer then we have to follow two ways.

Centralized look up service / Discovery protocal:

The new peer has to contact this service

if it need services from the other peers.

Broad cast:

In this type of service a new peer has to discover what node provides the services by broad casting a request to all the peers then the node which provides the service gives response to the peer.

Web-based

In this type of env. all the systems are connected either wired or wireless.

Eg: Cell phone, PCs.

Open source OS

This type of OS provides the source code which a programmer can easily modify & the best exmp. is Linux OS & other OS like windows are close source OS.

System Structure

OS services

User Interface

User interface have several forms.

(1) Command-line interface (CLI)

This helps to give commands and it includes the methods for entering that command.

(2) Batch interface:

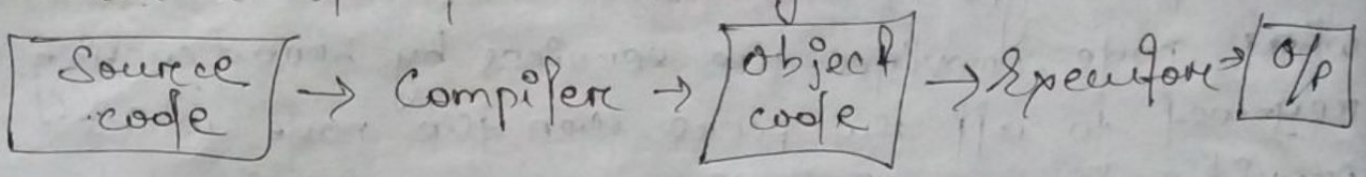
This interface is used to control the commands and its execution.

(3) GUI:

It is a window system which provides I/O operations, entering a text from key board, choose menu etc.

Program execution:

The system should able to execute a program which is stored in memory.



I/O operation:

To execute a program it may need file or I/O devices.
To increase the efficiency and to protect the data from other users the os provide an I/O operation.

File system manipulation:

This includes create, delete, search, update, read, write in a file.
-It also provides permissions management on access restrictions & deny services.

Communication

Communication between the processes either in the same system or in between different systems so that they can synchronize themselves.
This is done by message passing & shared memory.

Error detection & correction:

Errors are of different types.
(1) By cpu or memory
Overstorage in memory or unavailable of memory connection failure.
(2) By I/O devices.
Network failure, lack of paper in a printer.

(3) By user programs
illigal memory location, arithmatic overflow.
Error correction is done by debugging approach.

Resource allocation:

Different process may require some resources at the same time, this type of condition is managed by

of includes how much and what resources are used used by whom.

Protectional security:

when several process execute concorently we need operating system so that one process could not interfere with the other during its execution & it is done by providing an

User OS Interface

There are two approaches for users to interface with the os there are

- ↳ command line Interface
- ↳ Graphic User Interface.

CLI

- Some of the OS contain command interpreter in the kernel such as UNIX.
- The systems in which multiple command interpreter are used, the interpreters are known as shells.
- There are two ways in which the commands are interpreted.
- Command interpreter itself contain a code to execute the command.

Eg: Create command may cause the interpreter to jump into the section which contains the system calls.

- This approach basically used by UNIX.

Most of the os implement the commands through the system programs.

Eg: In UNIX to remove a file we use rm filename this means first the os has to search for the file rm & execute it by using the parameter the filename.

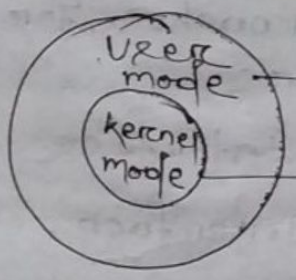
GUI

GUI provides a mouse based windows and menu system as an interface.

System call

System call provides an interface to the service of os.

User ↔ kernel
↓
Context Switching



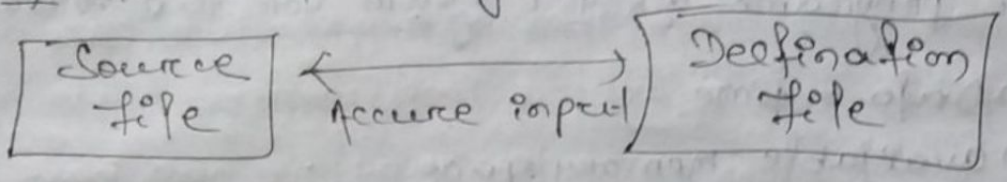
Safe mode

Privileged mode/ System mode

When the system need to access some resources then it changes its mode either from user to kernel or from kernel to user which is called as context switching.

- System call is a program by which a user can request service from the os
- This is written in C or C++

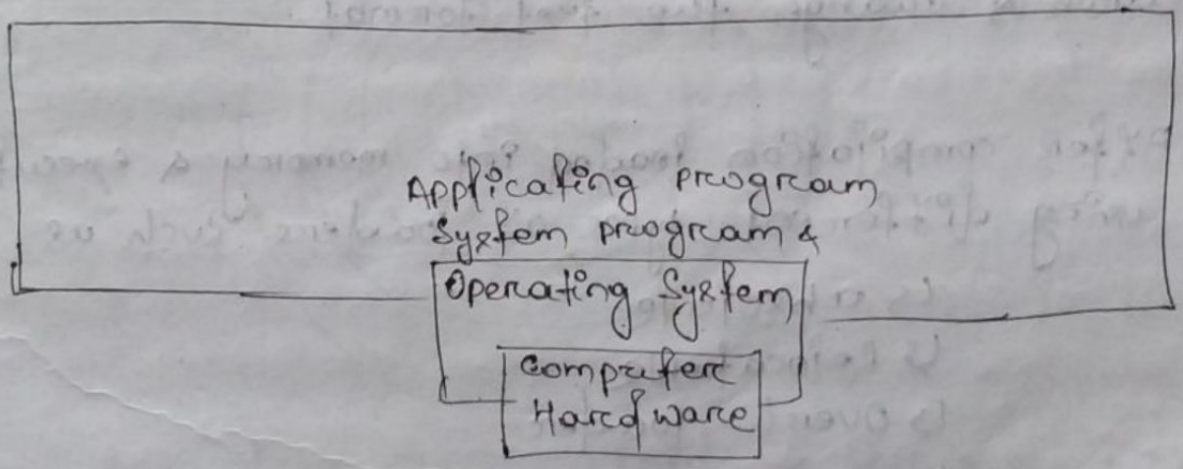
Ex: To read & copy data from 1 file to another.



Types of System call

- Process control
- File management
- Device management
- Information maintainance
- Communicate.

System Program



System program provides an env. for program execution and develop and also it provides an interface to system call.

- System programs are catagorised into following types on the basis of operation.

(1) File Management:

This system program is used to manipulate few files & directories.

- It deals with external parts of files.

(2) Status Information:

These programs ask the system for the following info.

(i) Data & time
(ii) available memory space
(iii) no. of users.
(iv) system performance
(v) debugging
(vi) Store & retrive info.

(3) File modification

These system program deals with the internal parts of file.

- Create & modify the content of the file.
  Ex: text editors
- Search & change the text format.

(4)

After compilation loaded into memory & executed by using different types of loaders such as:

↳ absolute
↳ Relocatable
↳ Overl loader
↳ Linkage editor

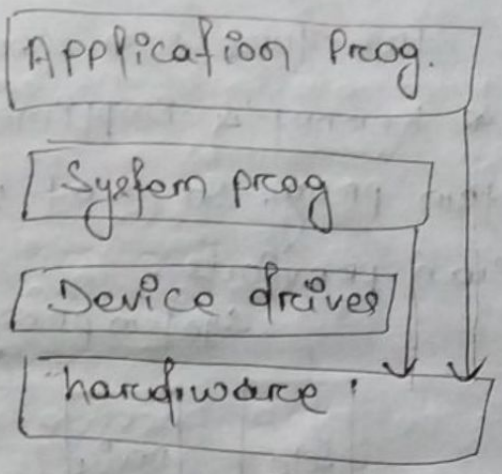
OS structure

(1) Simple structure:

MS DOS is an example of such a system which follows the simple structure of an OS.

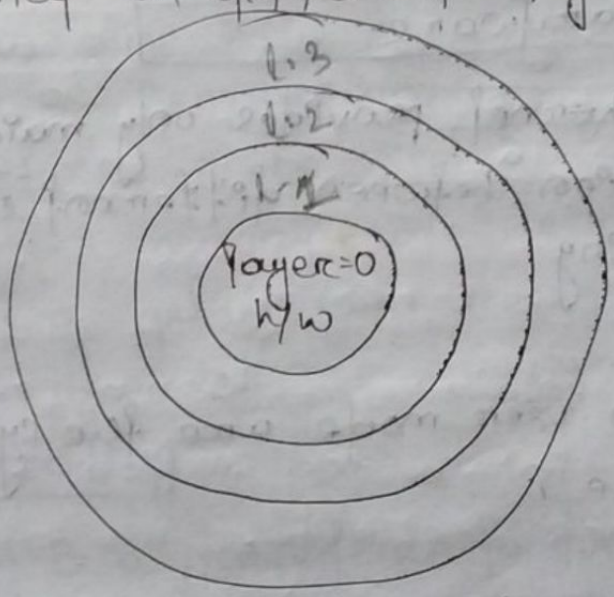
- In this type of structure the application programs are able to acces basic I/O devices.
- The main disadvantage of this structure is if an application prog. has some error during

execution even if affects the hardware directly.



(2) Layered Approach:

In this type of approach all the functionalities are defined in different layer independently.



- Execution, implimentation & debugging is easy.
- If needed you have to debug only that layer.
- User interface doesnot have direct access to the hardware.

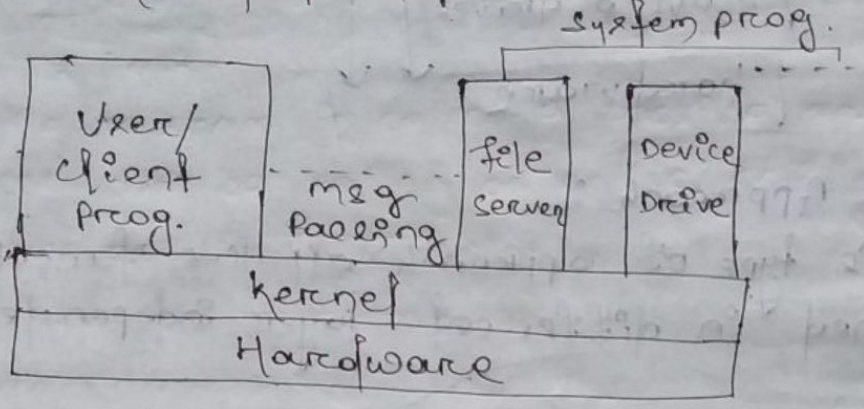
Disadvantages:

Not every efficient i.e it takes more time to execute a user program.

- It is difficult to design the upper & lower layers as a layer can use the layers below it & only it can provide service to the layer above it.

(3) Micro kernels :

By using this structure non-essential part are removed from the kernel & implimented them as user prog. or system prog. So that we can have a smaller kernel than previous.



The microkernel provides only main function & the communication between different components by using msg passing.

Advantages

Most fun' in user mode are the system progs. So it is secure.

Disadvantages

This structure may reduce the performance because the system fun's are increased.

(4) Modules :

- It is the best method for os design which involves oop techniques to create a modular kernel.
- The kernel has a set of core components & links them either during boot time or during the run time.
- This structure uses layered structure concept

as well as the micro kernel structure.
- But it is more flexible than layered structure and more efficient than micro-kernel structure.

OS Generation :

- OS can be generated for one machine at one site or may be designed for a class of machines irrespective of their peripheral configuration.
- It process called as system generation (SYSGEN) which is used to generate an os for each specific system env.

There are some info. which should be determined before the os generation.

1. What CPU to be used ?
2. How much memory space is available ?
3. What a system requires or need from an OS ?
4. What devices are available ?

System booting :

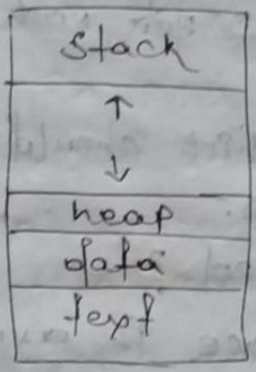
Booting is a procedure to load the kernel to start the computer.
- It is done by bootstrap prog. or bootstrap loader. It helps to locate the kernel loads it into memory and execute it.
- The bootstrap prog loaded in ROM. because it need not to be initialized & can't be affected by computer viruses like RAM.
- All forms of ROM also known as firmware.
- Disadvantage of using ROM is that to change the bootstrap prog. we need to replace the ROM chip by a new one to over come this problem we use EPROM.

UNIT-2

Process Management:

Process is as a program in execution.

- It's main need as a resources to complete the task (i.e. CPU time, memory, file, I/O devices.
- A program is a set of instructions written to complete a specific task.
- If the program is called a passive entity then a process can be called as active entity.



(Process in memory)

Process may contain more than a program code that is stored in the text section & it also contain the current activity.

Stack: stack includes the temperary data like function return values, local variables.

Heap: It includes the memory location which is dynamically allocated during the runtime.

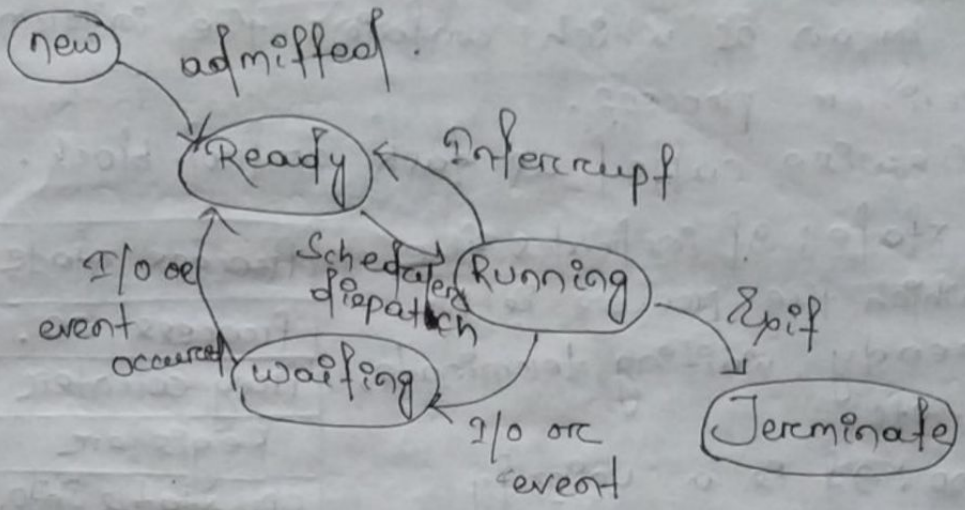
Data: It contains the value of program counter and processor registers.

Program: Set of instructions written for a specific task and which is executed by the system.

- The program and the interfaces are includes to

design o software.

Process state:



New state: A process is in new state when it is created or it goes for execution.

Ready state: After the creating of process it is admitted into the memory in the ready state to be assigned to the processor.

Running state: when the process are going to be executed it is in running state. If any interrupt (Software/Hardware) occure during running state then it goes to ready state.

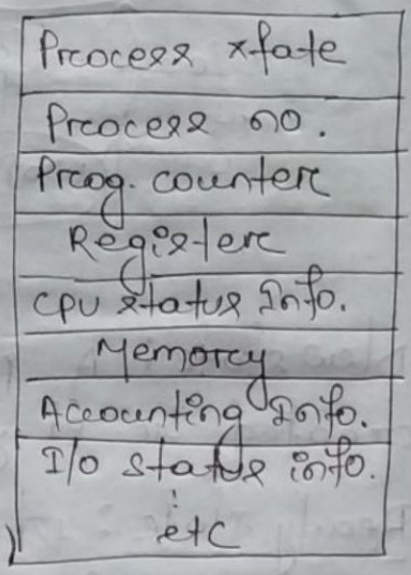
Waiting state: During the executing if a process need some event to complete (i/o devices available) or to get some signals) its execution.

- After the event complition it goes to the ready state.

Terminate state: After the execution the process is terminated.

Process control block:

Process control block is a data structure which is maintained by an OS which contains the information about a specific process.
It can otherwise called as task control block.



(i) Process state: It includes the state in which the process is in.
Ex: New, ready, waiting, terminate, running etc.

(ii) Process no: It is a uniquely identified no. i.e PID no.

(iii) Program counter: It holds the address of next instruction to be executed for the process.

(iv) CPU register: These registers are of different types and different no. according to the system architecture.
Ex. IR, Accumalator, general purpose register, MR, MDR, stack pointer etc.

(v) CPU states Info: This includes the information about the scheduling algorithms, the parameters, a pointer that points the position in the scheduling queue.

(vi) Memory management Info: This includes the value of registers, tables and memory space used by that process.

(vii) Accounting info: This includes the time, process no

(viii) I/O state Info: It stores the no. & list of i/o devices allocated to the process to complete its execution.

Process Scheduling:

The objective of using multiprogramming OS is that to make the CPU busy for all the times which improves the CPU utilization.

- The objective of using time sharing is to switch the CPU between different processes so that the user can interact with all the processes.
- A process scheduler selects a process from all the available processes for execution on the CPU to achieve these objectives.

Scheduling Queue:

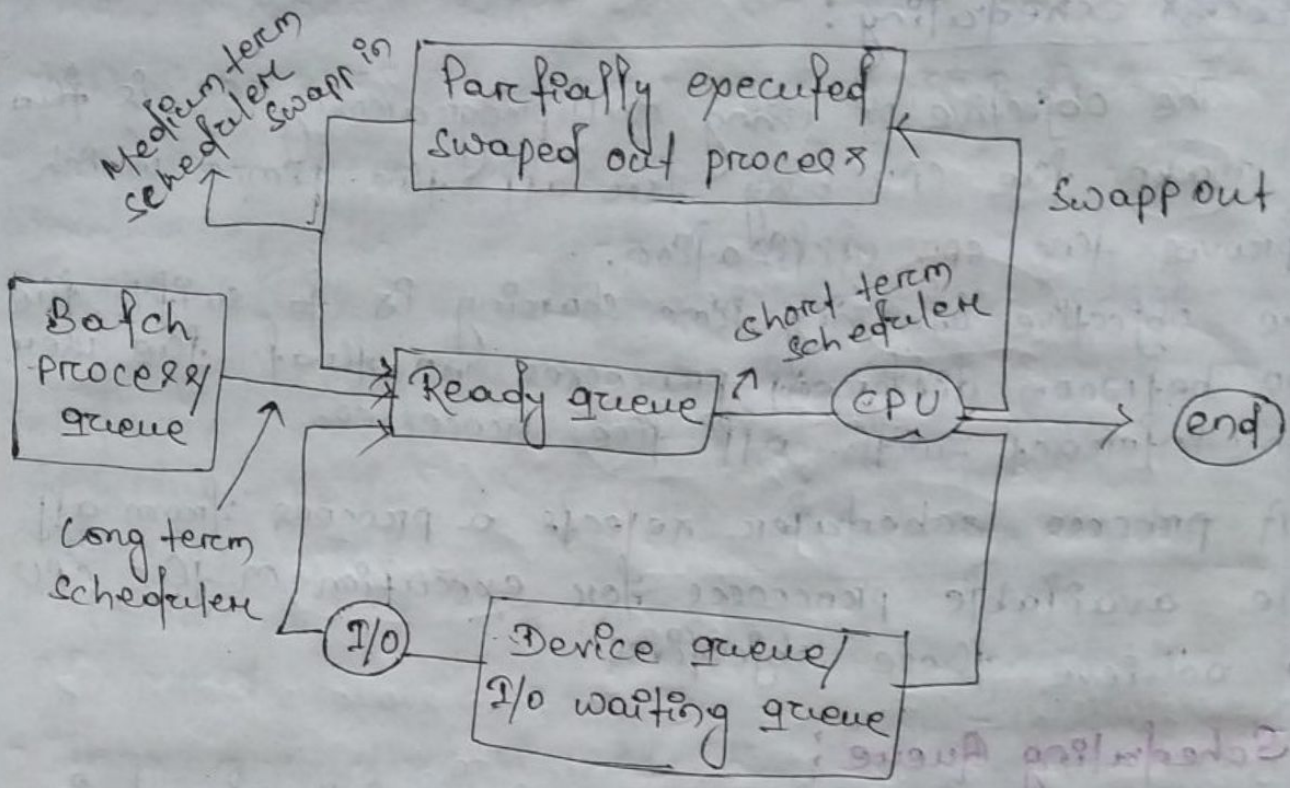
When process enter the system, it is stored in job queue.

The processes which are ready & waiting to execute, stored in ready queue.

- It is mainly in the form of linked list. ~~or data &~~
- The header part points to the first & last PCB in the list.
- During the running state the process may have to wait for a particular I/O device & the list is called as device queue or if some interrupt occurs then it goes to a queue which stores partialy executed process.

Scheduler:

The selection of processes from the queue for execution is carried out by the scheduler.



- Long term scheduler is used to fetch processes from the job pool into the memory for execution. So it is otherwise called as job scheduler.
- Short term scheduler is used to fetch processes which are ready for execution and allocate the CPU to one of them. So it is otherwise called as CPU scheduler.
- The frequency of execution of short term scheduler is more than long term scheduler.
- The long term scheduler has to select the processes according to the two conditions (i) I/O bound processes (those processes which spend more time to perform I/O operation than computation)
  (ii) CPU bound processes (these processes spend more time to perform computation & also produces I/O request in frequently.
- If all processes as CPU bound then the I/O

waiting queue will be empty.
- If all the processes are I/O bound then the ready queue will be always empty.
- Medium term schedular are used to load partially executed processes (if interrupt occurs) into ready queue. This is basically used in time sharing system.

Context Switch:

When some interrupt occurs the system needs to save the current context of the process which are running in the CPU so that it can be resume after the interrupt execution.
- We perform a state save to save the current state of the CPU either in kernel mode or in user mode and a state restore to resume the operation.
- When context switch occurs the kernel saves the context of whole process in its PCB and loads the context of new process from its PCB for the execution.
- The context switching speed varies from machine to machine.

Operations on process:

Process creation:

A process can create several new processes by using a system call (fork() or CreateProcess())
- The creating process is called as parent and the new processes are called as children of that process.

- The children processes can also create a no. of new processes by forming a tree structure.
- Each processes has assigned with a unique process identifier no. (Pid).
- If a process create new processes then there are 2 possibilities for the execution.
  (i) The parent process has to be executed simultaniously with the child process.
  (ii) The parent process has to wait till the execution of some or all the child processes.
- There are also two possibilities in terms of the content of the new process.
  (i) The new process contain same as parent process.
  (ii) The child process may have new program or data loaded into it.

(1) A new processes is created by a system call i.e fork().
(2) The exec() is used after the fork() to start the execution of one of the new processes.
(3) The execlp() is used to get list of processes which are going to be executed.
(4) When a parent process creat more children or it has to wait till the execution of child processes, it has to go to the waiting state by issuing the wait() to move it self into the ready queue.

⑤ When a process completes their execution it has to invoke or issue the exit sys'call().

Process Termination:

- When a process finishes its execution then operating system issues an exit() to delete it.
- At this point all the resources allocated to that processes must be deallocated.
- This situation can be called as normal termination.
- Termination can also be occured in some circumstances there are

(i) if a child process exceeds the use of the resources.

(ii) The task assign to the process is no longer needed.

(iii) If the parent process is terminating then the os does not allow the child process to continue. There is a cascading termination by which the child processes may not be terminated if the parent process exit.

Interprocess Communication:

It is a process by which another process can communicate with another process.

- The process may be either independent (it can't affect or affected by other processes in the system) or cooperating (it can affected or affect other processes in the system).

Independent → Don't share data

cooperating → share data.

There are several reasons for providing an environment to establish cooperation.

(1) Information sharing.

This means no. of processes may need to share same information for execution or no. of users interested in the same information.

(2) Computation speed:

To execute a process in a less amount of time we need to devide it in sub processes and all the processes has to be executed parallely to speedup the computation.

- This can be implimented in an environment which have multiple processing unit (CPU, I/O)

(3) Modularity:

Sometimes we need to devide system function in modules for better throughput.

(4) Convenience:

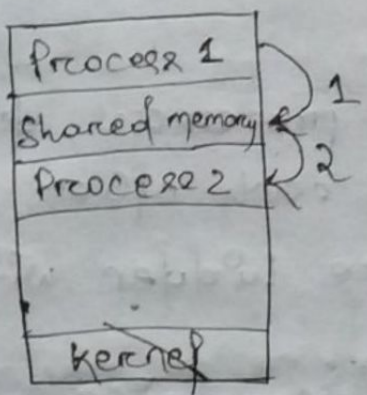
Some users may work on multiple task at the same time.

Ex: printing, browsing, listening music, downloading in parallel.

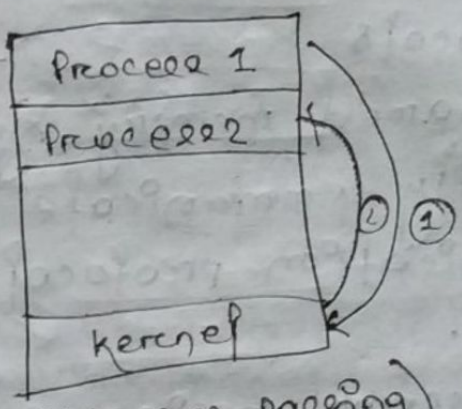
There are 2 models used to perform interprocess communication.

(1) Shared memory

(2) Message passing.



(Shared memory)



(Message passing)

Comparision between shared memory and message passing.

① Interface to communication:
- In shared memory systems communication between processors is implicit and transparent. processors access memory through the shared bus.
- In message passing systems processors must explicitly communicate with each other through messages.

② Complexity of the Architecture
- In shared memory system leverages conventional architecture better since existing processors can be added to the shared bus system easily.
- In message passing system it leads to a simpler multiprocessing architecture overall.

③ Convenience
- In shared memory system serial code runs without modification.
- In message passing system message passing libraries are available for a wide variety of

platforms.

(4) Protocols.

- In shared memory system processors do not explicitly communicate with each other so communication protocols are hidden within the system.
- In message passing system the communication protocols are fully under user control.

(1) Shared memory System:

It uses a memory which is shared by the processes to establish interprocess communication.

- Shared memory will be located in the address ~~base~~ space of the process which create or initialize the communication and if other processes want to share that memory then they have to attach their address space with the shared memory.
- The operations of the shared memory are not in the control of OS.
- More than one process can't write on the shared memory simultaniously but they can read.
- We follow the <u>producer-consumer problem</u> to define a shared memory concept.
- A producer process has to produce the info. which are going to be consume by the consumer process.

or

- Both the producer and consumer has want to

operate simultaneously then they have to use shared memory.

- They are maintaining a buffer of items which will be filled by the producer and emptied by the consumer.
- The process has to be synchronise i.e a consumer can't consume an item that is not yet produced.
- There are 2 types of bnd buffers.

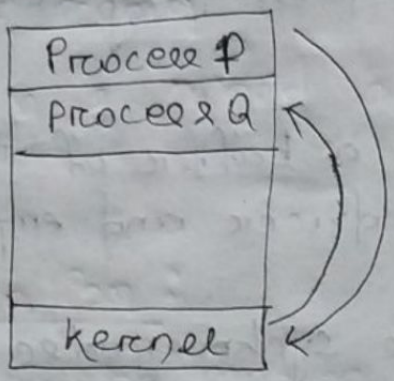
(1) Bounded : The size is limited, that means the producer will wait when the buffer is full and the consumer will wait when the buffer is empty.

(2) Unbounded : The size is unlimited, that means the consumer may have to wait if it need new items but the producer can produce new items always.

(2) Message Passing System.

- Message passing feature allows the processes to communicate with each other without using a shared memory in their address space and

- The message passing operation uses send (message) and receive (message) operation.
- The messages are either fixed size or variable size.
- Fixed size : The implimentation is easy but the programming is difficult.
- Variable size : The implimentation is difficult but the programming is



If process P wants to communicate with process Q or vice versa, then they must send to and receive from each other by using a communication link.

To implement a communication link logically between two processes there are several methods direct and indirect method, synchronous and asynchronous, automatic or explicit buffering.

There are some issue may arrise during these implementation.

(i) Naming
(ii) Synchronization
(iii) Buffering

Naming: This type of issue arrises in direct or indirect communication.

- In direct communication the communicating process know the sender as well as receiver name.
- The communication link has the following features automatically established when 2 process want to communicate.

There is exactly 1 link between two processes.

Symmetry

The sender & receiver process must have a name to each other for communicating.

Ex: send (Q, message) → Sends a msg to Q

Receive (P, message) → Receive from P.

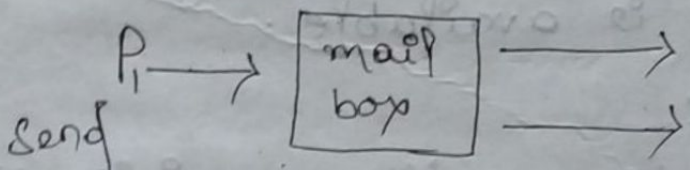
A Symmetry

The sender must have the name of the receive but the receiver does not know or not required to name the sender.

- An indirect communication has the following properties.

A link may be associated with more than one processes are in between each pair of processes there may exist more than one links.

The link is establish between the pair of processes if they shared a mail box.



def P_1 send a message and which process will receive the process will depends on following operation.

* There must be a single communication link
* By allowing one process at a time to receive the process.
* The recipients may be chosen arbitarily by using different algorithm.

Ex: round robin

- The mail box is owned by the os or one by the a process.

Synchronization:

Message passing may be blocking or non blocking also called as synchronous & asynchronous.

Blocking send:

The sending process is blocked until and unless the message is received by the receiver or by the mail box.

The sender sends the msg & resume the operation whether the message is received or not.

Blocking receive:

The receiver block its receiving process until the message is available.

Nonblocking receive:

The receiver continue the receiving process either if is valid process or not.

Buffering

During the communication the communicating processes stage in a temporary queue & the queue may can be implimented by the following ways.

(1) 0 capacity - Maximum length is 0.

There is no message in the waiting list & follows blocking send.

(2) Bounded capacity: The link capacity is finite or limited. If the queue is not filled then a new message can place in it and if it is full then sender block until the space will available.

(3) Unbounded: The size is unlimited

Communication in client-Server System

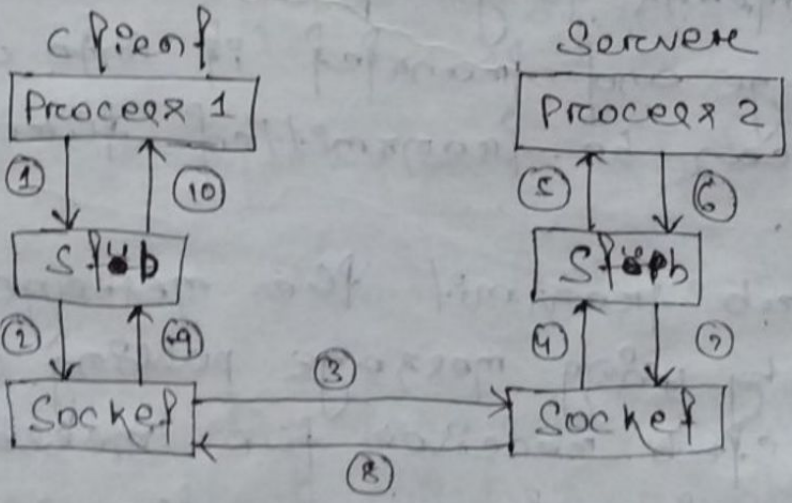
Sockets:

- Sockets are the end point of a communication and it is different for different processes.
- To identify a socket you need IP address & a port no.
- After getting the request from client the server system listen the request to a specified port.

Ex: ftp server listen to the port 21.

- A client process is assigned to a port no. by the host computer which is greater than 1024 because all the ports below 1024 are used for standard services.

Remote Procedure call:



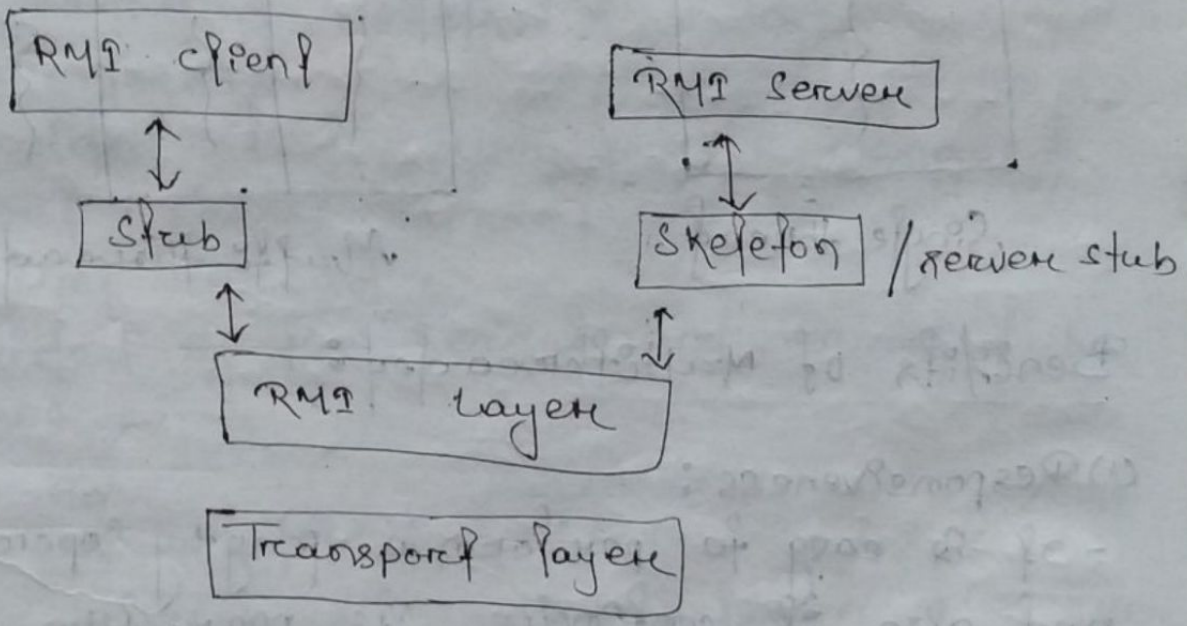
When a process wants to communicate with another process which is present in another system through a system and connected via

network uses a protocol called as RPC (Remote procedure call).
- RPC is a protocol which helps to establish communication between two remote processes.
- It follows message passing concept used in interprocess communication.
- Each procedure contains an identifier and the parameters.
- The RPC system hides the details by placing a stub on client side.
- On client side as well as in the server side when the client invokes remote procedure rpc system calls the stub by passing the parameter through remote procedure.
- The stub locates the code in the server and marshal the parameter.
- Marshaling means to group the parameters into a package and transfer it into a form so that it can be transmitted through the network.
- Then the stub transmit the message through the server by using message passing concept.
- The server stub receives the message from the socket and execute the procedure and returns the value to the client side by using same technique.

Remote Method Invocation:

Parameter	RPC	RMI
Support	Procedural prog.	Object oriented prog.
Parameter	Ordinary data	Objects
Efficiency	Less than RMI	More than RPC
Easy to use	Easy.	Difficult
Overheads	High	less.

Remote Method Invocation:



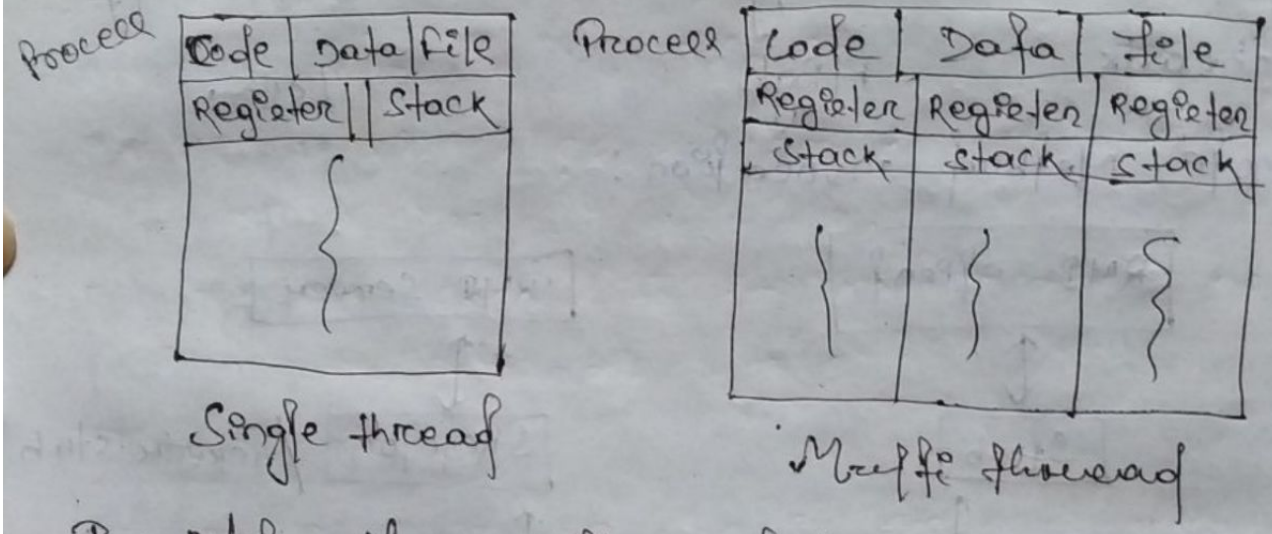
It is a Java feature is similar to RPC.

- It passes the parameter by using remote object.
- When a client called a remote method the stub for the remote method is called.
- Stub passes a parcel containing the method the name and and the skeleton is responsible for receiving the parcel.

- Then the skeleton unmarshaled parameter and invoke the appropriate method in the server.
- The xfub reassemble the return parcels and return it to the client.
- If the objects are the remote object they are pass by the reference.

Multithreaded Programming :

Process thread view process explorer



Single thread

Multi thread

Benifits of Multithreaded :

(1) Responsiveness :
- If is easy to perform a lengthy operation and also it continue its operation if 1 thread will be block.

(2) Resource sharing :
- The resources of the process will be shared between different thread which helps to perform more than one activity simultaneously.
- Ex : code, data, file.

(3) Economy :
- Single memory space and resources can be shared between multiple threads of the same process.
- It takes more cost to create and allocate memory and resources for each threads.

(4) Utilization of multiprocessor System :
- As a single threaded process can only run one cpu where multithreaded can run multiple cpus at a time.

Multithreading Models :
(1) User thread .
- Provided at the user level & managed without kernel support.
(2) Kernel thread .
- Provided by the kernel and managed by the OS.

Types of models :
- To establish the relation between the user thread and kernel thread there are different model.
- One-to-one
  many-to-one
  many-to-many.

Multithreading Models

One-to-one

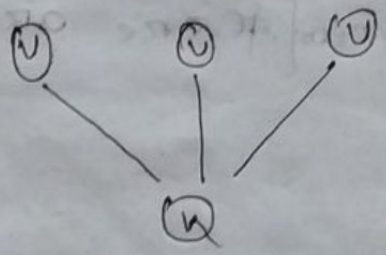
Advantages

- One user thread is assigned to one kernel thread. By using this if one thread makes a blocking system calls, others will not be affected by this.
- Parallel execution on multiprocessor is easy. ~~Eg: Solaris system~~

Disadvantages

- Burden ~~overall~~ on the system performance is high because per user thread you have to create corresponding kernel thread;
  Eg: Solaris system

Many to One



Advantages

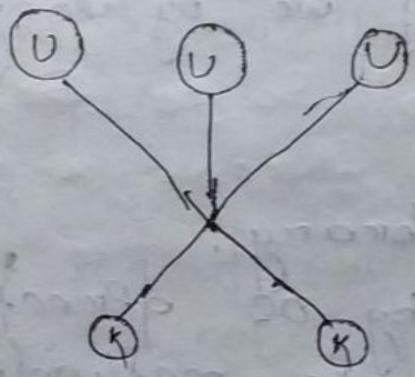
- Many user thread is assigned to one kernel thread.

- Thread management is easy so it is more efficient than others.

Disadvantages

- Parallel execution is difficult.
- If one thread makes a blocking system call then whole process will be block.
  Ex: Windows xp, Linux.

Many-to-Many



Advantages

- Many user thread assigned to equal or smaller no. of kernel threads.
- Multiprocessor utilization is high.

Disadvantages

- Less efficient than one-to-one & many-to-one
- If one user thread is bounded to one kernel thread.

Ex: Some unix, IRIS

Thread Library :

- Thread library provides Application programming Interface (API) for managing

(1) User level library
- All the code, data, methods are exist in user space.
- To invoke this type of function without using a system call or by using local function call.

Ex : POSIX Pthreads

(2) Kernel level library :
- It is managed by OS directly.
- All the code, data, methods exist in kernel space.
- For invoking you need system call.

Ex : Windows, POSIX (Win32)

NOTE: Java thread API.

Threading Issues :

(1) Fork System call

Fork() system call is used to create separate and duplicate process.

(2) Execution system call

exec() system call will replace the entire process of all threads.

(3) Cancellation

Cancellation is the task of terminating a thread before its completion.

- A thread that is to be cancelled is called as target thread.
- This process is done be two scenario.
  (1) Asychronous
  (2) Deferred

(1) Asychronous:
A thread is cancelled instantly without checking any options.

(2) Deferred:
The target thread periodically check whether to terminate or not.

(4) Signal Handeling:
Signal is used when a specific event occurs. For signal handling we follow some specific pattern.

① Signal generation
② Delivering of a signal to a particular process.
③ After the delivary of signal within the return it must be handel.

Sychronous signal:
It delivered to the same process which generates the signal.

signal is generated

- There are 2 types of signal handler.
  (1) Default (kernel)
  (2) User defined (User)

Transmit the signal to the specific thread which needs it.
- Deliver the signal to every thread in the process.
- Deliver the signal to a certain thread
- Assign a specific thread to receive all the signals.

(5) Thread pool :

When a server receives a request it creates a thread to process that request.
- There may be some problems will arrise like the amount of time needed for thread creation.
- If there are multiple no of request at the same point of time and need to be executed by the thread.
- To overcome this problem we use a thread pull.
- Thread pull is a place where a no. of thread are created when the process start execution and place it in the pool.

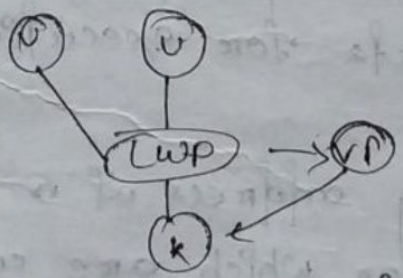
- The request can be provided to a existing thread which is faster than creation a new thread.
- The thread pool have a limitation on how many thread will be present in the pool.
- It is determine by some factors like
  (1) No. of CPUs in the system.
  (2) Amount of physical memory.
  (3) Expected concorent client request.

(6) Thread Specific data :

At some point of time each thread may need its own copy for some specific data.

Ex: Transaction processing system i.e each transaction has a specific identifier.

(7) Scheduler Activation :



So many to many model where is a light weight process (LWP) which appears to be a virtual processor for user thread library and it helps to schedule the user thread to run.

- There is a technique called as scheduler activation which works as follows.

- The kernel provides the available threads to the available virtual processors through a procedure called as upcall and there is a upcall handler for a thread library.

OS Example

Windows xp threads:

Windows xp provide fiber library which supports functionality of one-to-one model.

Components of thread:

- Thread ID
- Register set
- User stack
- kernel stack
- Private data storage: It uses various dynamic storage libraries or dynamic link library (DLL).

There are 3 thread blocks.

(1) E THREAD: It starts for executing thread block.
- It contains starting address of a thread or the set of instructions which are controled by that thread.
- A pointer to the parent process

(2) K THREAD (kernel thread block):
- It contains all the informations like scheduling, synchronization kernel, stack and a pointer to TEB.

- kthread and E-thread operates for kernel thread.

(3) TEB (Thread environment block): thread specific
- It contains thread ID, user stack and data storage and it exist in user space.

Linux thread:

- Linux uses fork system call and another one is clone system call to create a thread.
- There are multiple flag are used to determine how much sharing is possible between parent & child class.

Flags are: into
CLONE-FS - File sharing.
CLONE-VM - Memory space is shared.
CLONE-SIGHAND - Signal handlers shared.
CLONE-FILES - Open file sharing.

UNIT-3

Preemptive Scheduling:

~~Ready to~~
- Runing to ready. (Non-preemptive)
- Running to waiting (Preemptive)
- Waiting to ready. (Non-preemptive)
- When a process terminates. (Preemptive)

Dispatcher

- Dispatcher is a component which gives control of CPU to the processes which are selected by the short term scheduler.
- The time taken for a dispatcher to stop one process and start another process for running is known as dispatcher latency

Scheduling Algorithms:

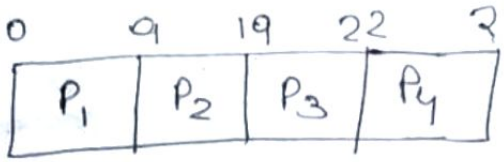
(1) FCFS (First come - First Serve)

- FCFS stands for first come first serve algorithm.
- It is used in queue and same FIFO concept.
- It is a non-preemptive type of algorithm.
- Ex: (1)

Process	CPU Burst (m.s)	Waiting time
P_1	9	0
P_2	10	9
P_3	3	19
P_4	5	22

The result of the algorithm will be shown in Gantt chart.

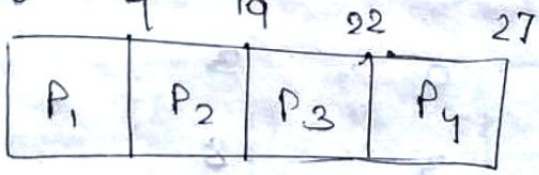
Arrival time = 0



Avg. waiting time = $\frac{0+9+19+22}{4}$ = 12.5 ms

(2).

Process	CPU Burst (ms)	Arrival time
P_1	9	0
P_2	10	3
P_3	3	5
P_4	5	9



Waiting time

$P_1 = 0$

$P_2 = 9-3 = 6$

$P_3 = 19-5 = 14$

1. $P_4 = 22-9 = 13$

Avg = 3.23 ms

Advantage

of is simple and easy to implement.

Disadvantage

Convoy effect - All the processes has to wait for 1 big process to be executed.

~~through~~ Turnaround time

$P_1 = 9$

$P_2 = 16$

$P_3 = 17$

$P_4 = 16$

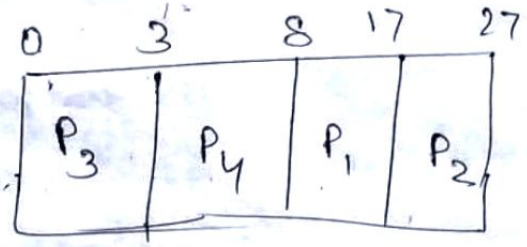
(2) SJF (Shortest job first)

→ Non-Preemptive

Turn around time = completion time − arrival time.

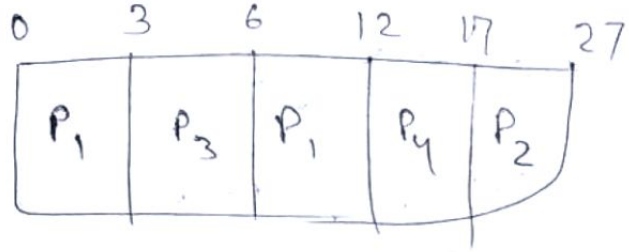
Waiting time = Turn around time − Burst time
or execution start time − arrival time.

Process	CPU Burst	Waiting time	Turn around time	Arrive time
P_1	9	8	17	0
P_2	10	17	27	3
P_3	3	0	3	5
P_4	5	3	8	9



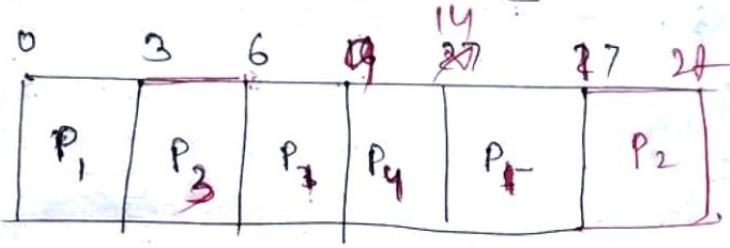
→ Preemptive

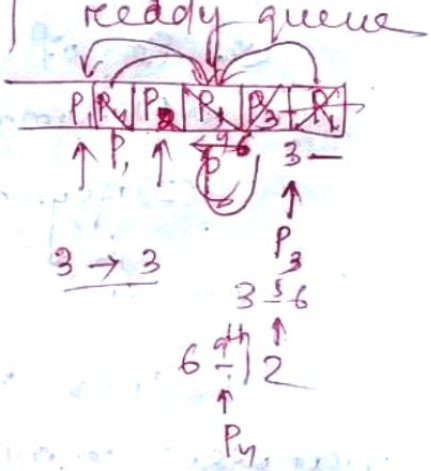
Process	CPU burst	Arrival time	waiting time	Turn around time
P_1	9	0	12-9 = 3	12-0 = 12
P_2	10	5	22-10 = 12	27-5 = 22
P_3	3	3	3-3 = 0	6-3 = 3
P_4	5	9	8-5 = 3	17-9 = 8



(3) Priority Scheduling

Process	CPU Burst	Priority	Arrival time
P_1	9	3	0
P_2	10	4	5
P_3	3	0	3
P_4	5	2	9



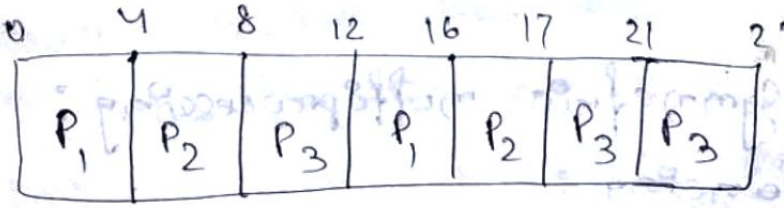


(4) Round Robin Scheduling :

Arrival time = 0

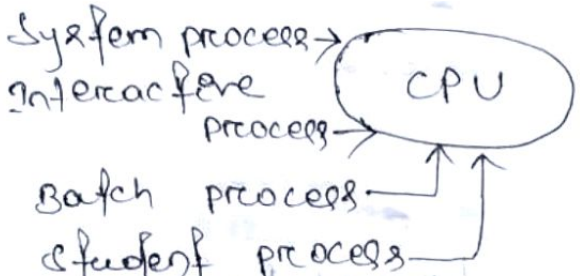
Time quantum = 4 ms

Process	Burst	Turnaround time	Waiting time
P_1	8	16 - 0 = 16	16 - 8 = 8
P_2	5	17 - 0 = 17	17 - 5 = 12
P_3	9	22 - 0 = 22	22 - 9 = 13



Average time = $\frac{12 + 8 + 13}{3} = \frac{33}{3} = 11$

(5) Multilevel Queue scheduling



System processes :- FCFS
Interactive processes :- SJF
Batch processes :- Round Robin
Student processes :- Priority.

Multiple Processor Scheduling:

It is of 2 types.

(1) Symmetric
(2) Asymmetric

Symmetric

All cpus have the same priority and the processes may be in a common queue or each processor have its own private queue.

Asymmetric

It uses client-server strategy or master slave strategy i.e one processor handel all the activities (master) and other processors work in user mode.

Issues of Symmetric multiprocessing:

(1) Load balancing:

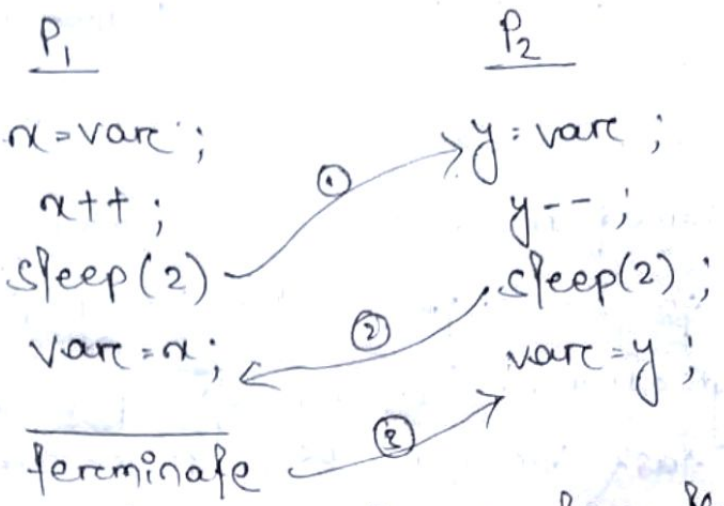
The work load will be evenly distributed

among all the processors in a symmetric multi processing system.
- There are 2 approaches of load balancing i.e
  (i) Push migration: A process is moving from overloaded to an idle or less busy processor.
  (ii) Pull migration: when an idle processor pulls a waiting task from a busy processor.

(2) Thread scheduling:
It is needed in every multithreading ~~contension scope~~ models.
- How it will be scheduled decided by contension scope. for many to many and many to one process contension scope.
- 1 to 1 one to one if is called system contension scope.

(3) P Thread scheduling:
P Thread identify contension scope value
- PTHREAD-SCOPE-PROCESS schedules the thread using PCS.
- PTHREAD-SCOPE-SYSTEM uses SCS.

Synchronization:
There are two type of processes i.e co-operative and independent.
- Co-operative processes are that who share something between them i.e variable, code, file, etc.
- The processes are not shared between them in independent.



After 1st process termination the value of shared variable i.e var=5 but after the completion of P_2 process the value is 3.

- The last updated value of variable is incorrect as both the process and allowed to manipulate the variable concorcently.
- This type of condition is called as race condition.
- So to deal with this problem we need to allow one process at a time manipulate shared data we use process synchronization or co-ordination.

Critical Section Problem:

Each process have some shared segment or location called as critical section, in this location a process perform all shared operation.

- Solution to critical section problem must satisfied the following characters.
  - Mutual exclusive
  - progress
  - bounded wait
  - No assumptions regarding the H/W

Peterson's Solution :

```
#define N=2;
#define false=0;
#define true=1;
#define int interest[N]=false;
entry(int process)
 {
  int other;
  interest[process]=true;
   other=1-process;
   turn=true;
 while (interest[other]==false && turn[process]
                                  ==true)
 {
   CS
 }
exit()
 {
  interest[process]=false;
 }
```

Semaphore :

- Common variable
- Multiple mutually exclusive co-operative process.
- Satisfy the synchronisation.

It is of 2 types.

(i) Countable (∞ to $-\infty$)

(ii) binary (0,1)

```
wait (semaphore s)
 {
  s.value = s.value - 1;
```

```
if (S.value < 0)
{
send the process to block list and sleep();
}
}
```

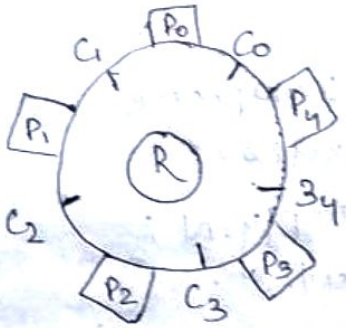
Reader-Writer problem by using Semaphore:

```
typedef int semaphore;
semaphore mutex = 1
semaphore db = 1
int rc = 0 ; // Readers counter
void reader (void)
{
  while (TRUE){
  down (mutex)
  rc = rc + 1 ;
 if (rc == 1) down (db)
  up (mutex) ;
  read_data_base();
  down (mutex);
  rc = rc - 1 ;
 if (rc == 0) up (db) ;
  up (mutex);
  use data read()
 }
}
void writer (void)
{
  while (TRUE) {
  think_up_data();
  down (db)
 write_data_base();
} } up (db)
```

Producer-Consumer Problem:

```
# DEFINE N=5
Typedef int semaphore
semaphore mutex = 1;
semaphore empty = N;
semaphore full = 0;
void producer (void)
{
 int item;
while (true) {
down (empty);
down (mutex);
insert_item (item);
up (mutex);
up (full);
}
}
 void consumer (void)
 {
  int item;
 while (true) {
 down (full);
 down (mutex);
 item = remove_item ();
 up (mutex);
 up (empty);
 consume_item (item);
 }
 }
```

Dining Philosophers Problem:

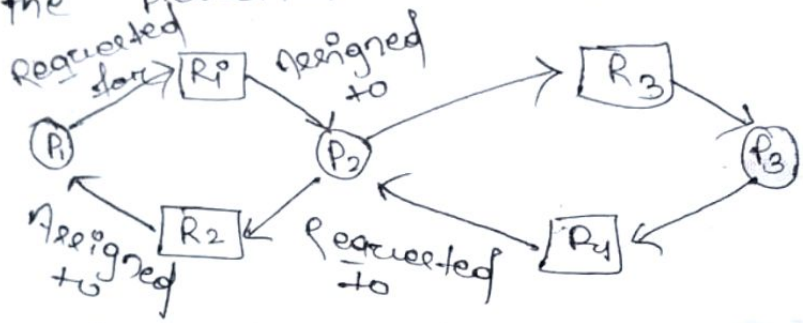


Atomic Transaction :

UNIT-4

Dead Lock

Dead lock is a situation where one process is requesting for a resource so it will be in the waiting state and the requested resource is held by another waiting process. that means both the processes are in the waiting state.



(Resource Allocation Graph)

- This graph consist of two types of nodes
  - Process : represented by circle
  - resource : represented by square.
- There are 2 edges
  - Requesting edge : From the process to resource.
  - Assignment edge : From the resource to process.

Conditions for dead lock :

(1) Mutual exclusion
(2) Hold and wait
(3) no. preemption
(4) Circular wait.

(1) Mutual exclusion : One process can access a non shareable resource at a time

(2) Atleast one resource be in the non sharable

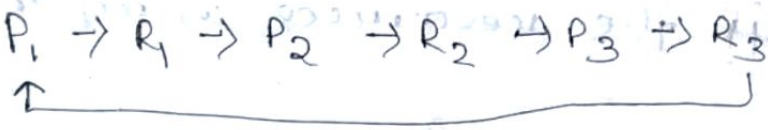
mode.

(2) Hold and wait: In this condition one process hold one resource and waiting for other additional resource to complete their execution which is held by other process.

(3) No. Preemption:

The resources are not release in middle of execution.

(4) Circular wait:



Deadlock Prevention:

Mutual Exclusion:

Rules:

- Convert all non-sharable resources to sharable resource but this condition is not applicable for some resources like printers.

Hold and wait:

Rules:

- A process can request a resource if it has none.
- A process has to request and be allocated all its resources before execution.

No Preemption:

Rule:

- A process can request a resource in two conditions, if they are available then we allocate them to the process.

of the resource is not available and allocated to a process which is in waiting state for other resources, if this condition occurs then preempt the resource from the waiting process and allocate it to the requesting process.

Circular wait:

Rule:

~~Give the no.~~

- Numbring all the resources in an increasing order.
- A process can request a resource of type R_j which is and released all the resources of type R_i such that $F(R_i) > F(R_j)$.

OR

A process can initially request any no. of resources of type R_j and after that it can request for resources type R_i if

$F(R_i < F(R_j)$

Deadlock Avoidance:

Deadlock avoidance can happen if we check the state of the system before allocating the ~~processes~~ or resources or deadlock avoidance algorithms checks before allocating the resources about the state of the system.

There are basically 2 states.

(1) Unsafe (Deadlock might occure)

(2) Safe (No deadlock will happen)

Banker's Algorithm

Available : It is a vector of length.

Available [i] = k means there are k no. of instances of resource.

Need : It is a nxm matrix which defines the no. of resources needed for a process.

need [i, j] = max[i, j] − Allocate [i, j]

Need [i, j] = k means process Pi needs k no. of instances of resource Rj to complete its resource

Max : It is a nxm matrix which defines maximum demand of each process.

Max [i, j] = k means process Pi can demand maximum of k no. of instances of resource Rj.

Allocation : It is currently allocated resource to each process.

Allocation [i, j] = k means process Pi allocated with k no. of instances of resource type Rj.

Safety Algorithm

This algorithm is used to find out whether the system is in safe state or not.
There are two vectors work and finish of length m and n respectively.

① work = available
finish [i] = false, i = 1, 2 ... n.

② find an [i] such that finish [i] = true;
need [i] <= work;

if no such [i] exist goto step 4.

③ work = work + allocation [i];
finish [i] = true;
go to step 2.

④ if finish [i] = true, ∀ i.

Resource Request Algorithm :

This algorithm is used when a process request for a resource.
~~if request is a vector~~
if request i [j] = k that means a process P_i can request k no. of instances of resources of type R_j.

Steps

① if request i <= need i, then go to step 2 otherwise generate an error condition as its request exceeds the need.

② if request i <= available, then go to step 3
otherwise P_i must wait.

③ After allocating the resources to P_i the state of

the following are

Available = Available - request

allocation = Allocation + request

need = need - request.

if resource allocation ~~step~~ is safe is safe that means P_i is allocated with all the resources otherwise the system is in

Process	Allocation			MAX			Available			need		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	0	1	0	7	5	3	10	5	7	7	4	3
P_2	2	0	0	3	2	2				1	2	2
P_3	3	0	2	9	0	2				6	0	0
P_4	2	1	1	2	2	2				0	1	1
P_5	0	0	2	4	3	3				4	3	1

Available = Available + allocation.

	Allocation			MAX			Available			need		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	0	1	1	1	1	3	0	2	1	1	0 ✓ (4)
P_1	2	0	0	6	2	3	4	3	3	1	2	3 ✓ (5)
P_2	1	3	1	4	3	1	5	3	4	3	0	0 ✓ (1)
P_3	1	0	1	1	0	1	5	3	5	1	0	0 ✓ (2)
P_4	0	0	1	3	2	1	5	3	6	3	2	0 ✓ (3)
	4	3	4				7	3	6			

A = 7
B = 3
C = 6
resource

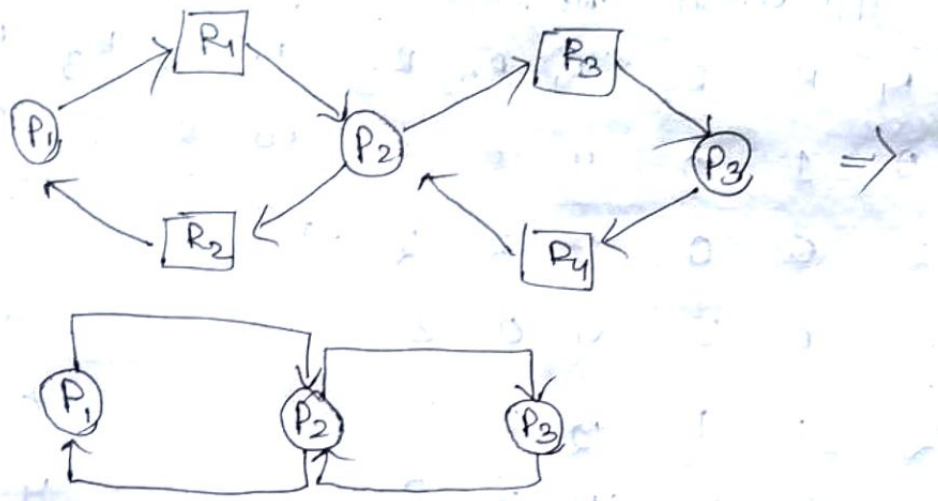
Deadlock detection:

The deadlock detection algorithm uses two methods

(1) Wait-for-graph for single instance resource type.

(2) Detection algorithm for multiple instance resource type.

Single instance of resource type:



(Wait-for-graph)

An edge between P_1 to P_2 means there is a resource shared between two processes and there are two edges from P_1 to R_q & R_q to P_2 for a common resource R_q.

If there is a cycle in wait-for-graph then the system is in deadlock state.

In the above example there are two cycle

Multiple instance of resource type

The detection algorith is approximately same as Banker's algorithm.

Data structures

Available :

Allocation :

Request [i, j] = k :

Detection Algorithm

① Work = Available
Finish [i] = false, allocation ≠ 0
otherwise, finish [i] = true

② Find i ;
finish [i] == ~~false~~ false ;
Request [i] ≤ Available ;

③ Available = Available + Allocation [i] ;
finish [i] = true ;

④ if finish [i] = false
then the system is in ~~deadlock~~ unsafe state and the process p_i is dead locked.

Recovery from Deadlock :

① Process termination :
- Abort all the deadlock processes.
- Abort one by one process until the deadlock is terminated.

② Resource preemption :
- Select a victim.

- Roll back
- Starvation.

Memory Management:

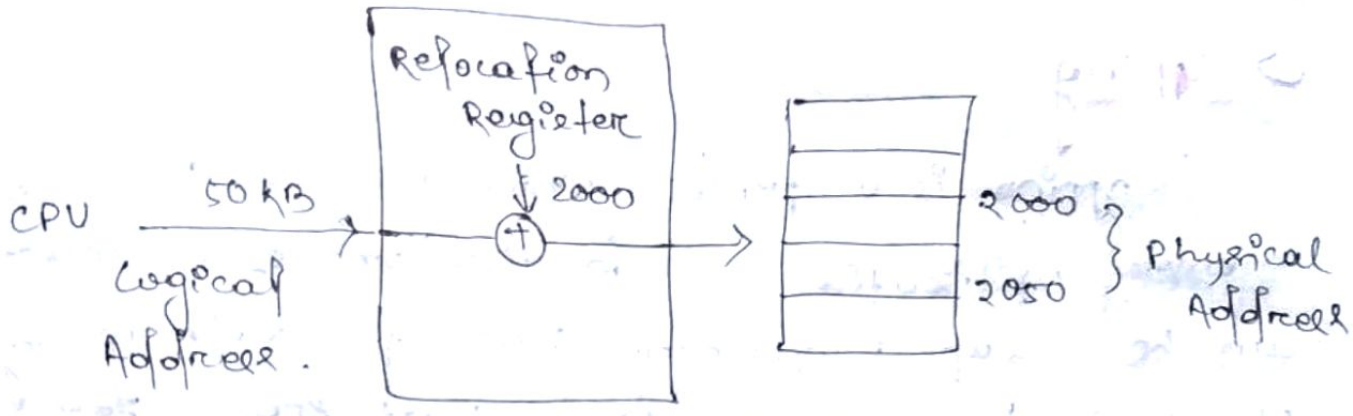
Memory Management is a functionality of a operating system which performs many functions like assigning the memory space to the processes, keep track of allocation and deallocation, load and store operation and memory utilization.

Logical vs Physical ~~memory~~ address space.

- Logical addresses are generated by use and the physical addresses generated by memory management unit.
- Let one program has the size of 50KB and it is loaded in main memory in between 2000-2050. i.e called as logical address. (Actual loaded address)
- All the logical addresses are generated by a program called as logical address space and the corresponding ~~logical addresses~~ physical address of the logical address is called as physical address.

Physical address = Relocation register address + logical address.

Relocation register (Base register)



At the run time the logical addresses are mapped into physical addresses by memory management unit.

Dynamic Linking and Loading:

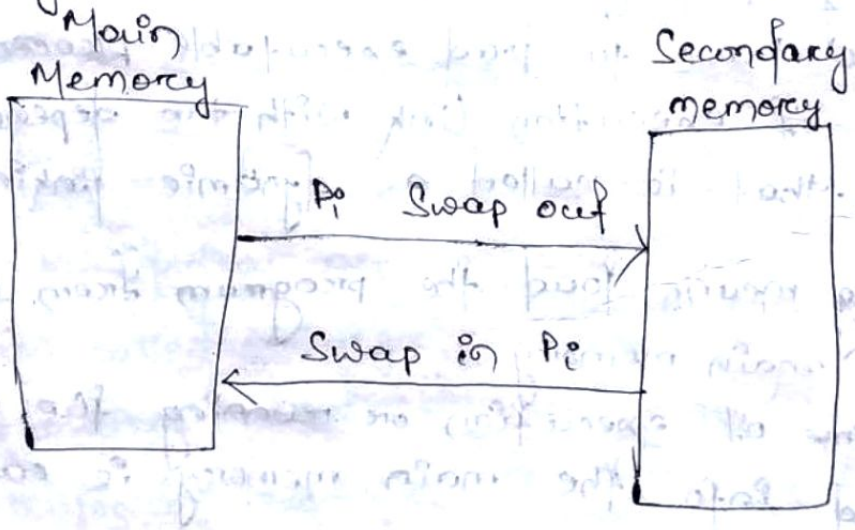
If there are two processes P and Q where P is dependent on Q by loading both the processes into the memory it takes both time & space.
- So if you have to load executable process P at the time of execution link with the dependent process Q, that is called as dynamic linking.

Loading means load the program from secondary memory to main memory.
At the time of execution or running the routines are loaded into the main memory is called as dynamic loading.
If the loading is perform at the compile time is called as static loading.
OR Dynamic loading means a routine is not loaded until it is called.

Swapping

Swapping is a process which is used to increase memory utilization and in this process a process can be swap temporarily from the main memory or disk to make available that space for other processes.

- Though it takes more time but it increases the performance of a multiprogramming system.
- It is also called as a technique for memory compaction.
- To move a process from main memory to secondary is called swap out and secondary to main memory is called swap in.



Swapping required a backing store which has a fast storage disk.

Contiguous Memory Allocation:

In this process the available memory will be allocated to the incoming processes in a continuous manner.
This is done by different method.

(1) Fixed size / static partitioning:

- Single partition
- Multiple partition.
  - * Fixed equal multiple partition.
  - * Fixed variable multiple partition

no. of partition are same but the size of the partition may same or may not be

(2) Variable size

Paging

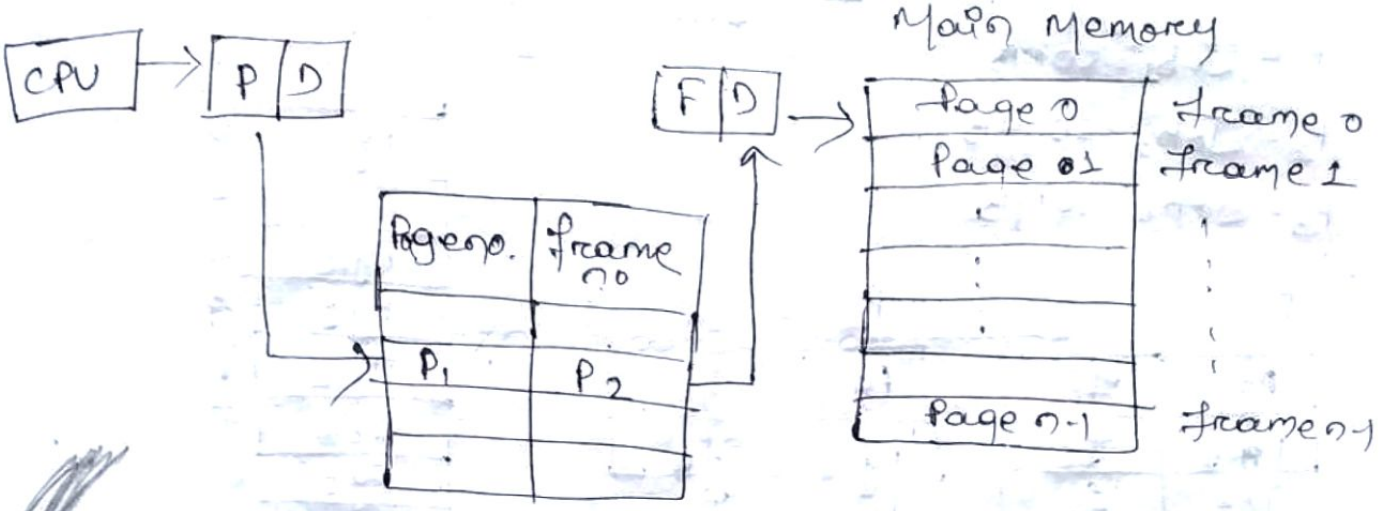
It is a memory management technique in which a memory divided into fixed page.
Paging is used for non contiguous memory allocation method.
- Paging is used for faster access to data.
- when a program needs a page then if it available in the main memory as the OS copies a certain no. of pages from your storage device to main memory so that paging allows the physical address space of a process to be non contiguous.

The basic idea of paging is a physical memory (main memory) is divided into fixed sized block

called as frames, the logical address space is divided into fixed sized blocks called as the pages.

- But the page size and frame size should be equal.
- The size of the frame or a page is depending on the OS.
- The OS maintains a data structure which is a page table, i.e used for mapping.
- The page table specifies some useful information that tells which frames are allocated or available as well as how many total frames are there and so on.
- The general page table consisting of two field, such as (1) Page no.
  (2) Frame no.
- Each OS has its own methods for storing page tables as well as most allocate a page table for each process.
- Paging is to deal with external fragmentation problem, this is allowed the logical address space of a process to be non contiguous which makes the process to be allowcated physical memory.
- Every address is generated by the CPU but it is divided into 2 parts.
  (1) Page no.

(2) Page Offset / Displacement



If the process size is 10 byte and the main memory size is 20 byte where the page size is 2 byte. If the CPU wants 7 byte ~~of~~ the of the process which is stored in 17 byte in the main memory what will be the logical address and physical address.

Shared Page

P_1 = Job_1 = 50 MB

P_2 = Job_2 = 50 MB

P_3 = Job_3 = 50 MB

Text editore = 100 MB

Main Memory

OS	
Pg_{11}	0
TE_1	1
Pg_{12}	2
Pg_{21}	3
TE_2	4
TE_3	5
Pg_{22}	6
TE_4	7
P_{31}	8
P_{32}	9

P_1

Pg_{11}	f_0
Pg_{12}	f_2
TE_1	f_1
TE_2	f_4
TE_3	f_8
TE_4	f_7

P_2

Pg_{21}	f_3
Pg_{22}	f_6
TE_1	f_1
TE_2	f_4
TE_3	f_5
TE_4	f_7

P_3

P_{31}	f_8
P_{32}	f_9
TE_1	f_1
TE_2	f_4
TE_3	f_5
TE_4	f_7

Shared pages are TE_1, TE_2, TE_3, TE_4

2^{10} - 1 KB

2^{20} - 1 MB

2^{30} - 1 GB

2^{40} - 1 TB

Structure of Page table:

(1) Multilevel paging or hirechical paging:

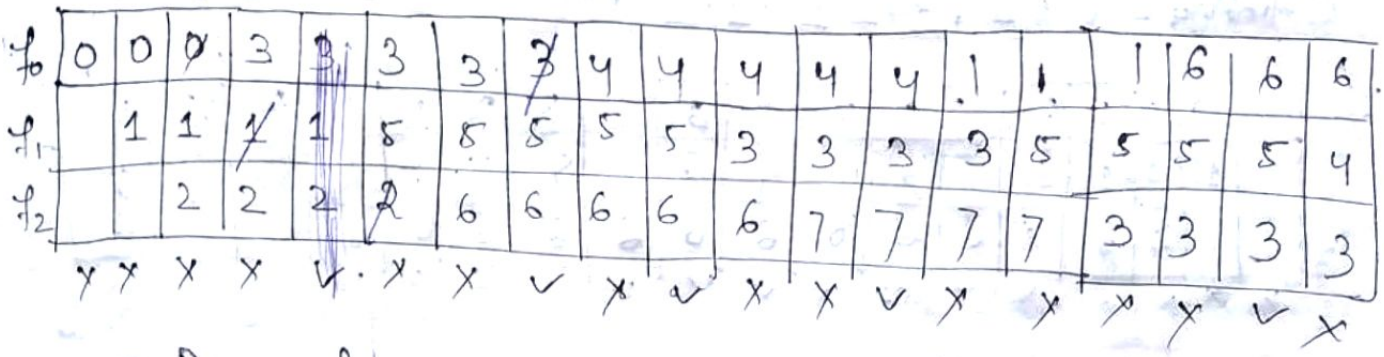
(2) Hash page table.

(3) Inverted Page table.

Page Replacement Algorithm:

FIFO

String :- 0, 1, 2, 3, 5, 6, 3, 4, 6, 3, 7, 3, 1, 5, 3, 6, 3, 4

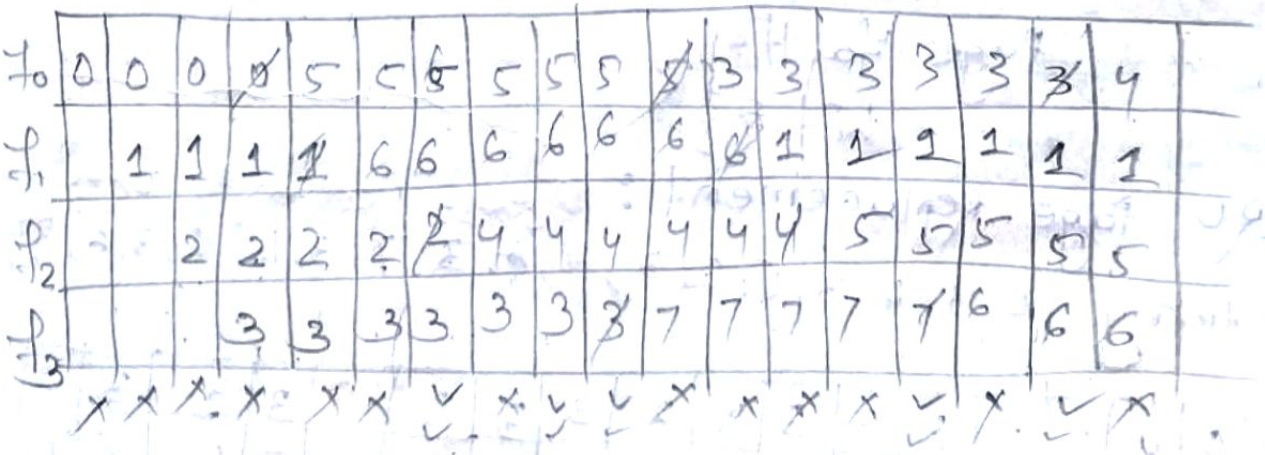


f_0	0	0	0	3	3	3	3	3	4	4	4	4	4	1	1	1	6	6	6
f_1		1	1	1	1	5	5	5	5	5	3	3	3	3	5	5	5	5	4
f_2			2	2	2	2	6	6	6	6	6	7	7	7	7	3	3	3	3
	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗

Page fault = 14

hit = 5

Belady's Anamoly:



f_0	0	0	0	0	5	5	5	5	5	5	5	3	3	3	3	3	3	4	
f_1		1	1	1	1	6	6	6	6	6	6	1	1	1	1	1	1		
f_2			2	2	2	2	4	4	4	4	4	4	5	5	5	5	5		
f_3				3	3	3	3	3	3	7	7	7	7	7	6	6	6		
	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗	

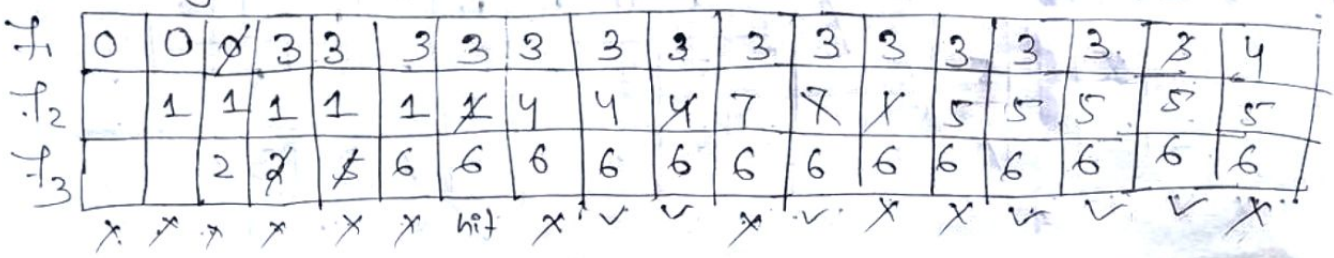
Page fault - 13

hit - 5

Optimal Page Replacement Algorithm:

Replace the page which is not going to be used for a long period of time.

String - 0, 1, 2, 3, 5, 6, 3, 4, 6, 3, 7, 3, 1, 5, 3, 6, 3, 4



Page fault = 11

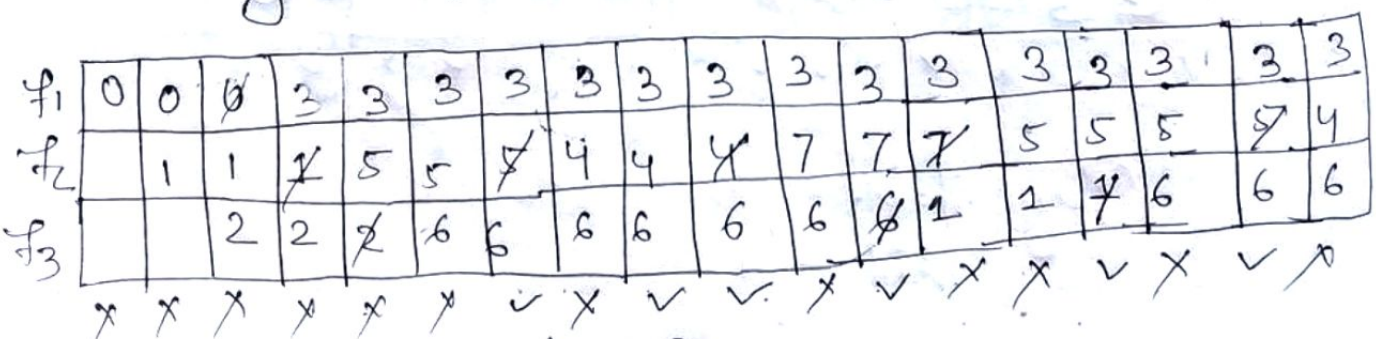
hit = 7

Page fault = 10

hit = 8

LRU Page Replacement: Least recently used

String: 0, 1, 2, 3, 5, 6, 3, 4, 6, 3, 7, 3, 1, 5, 3, 6, 3, 4



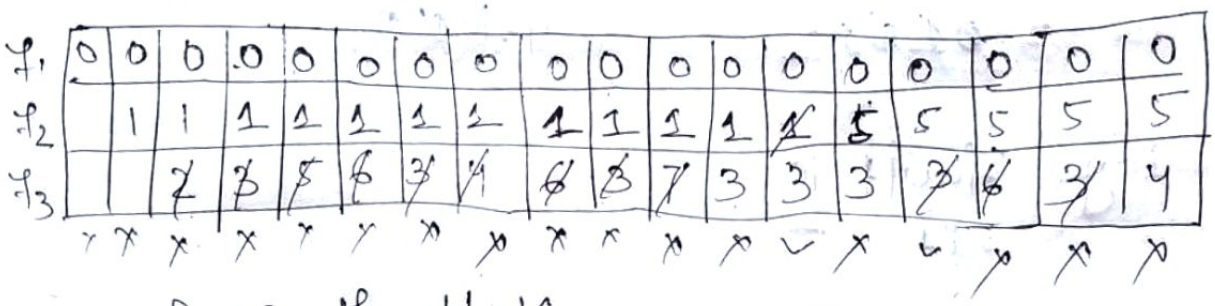
Page fault = 12

hit = 6

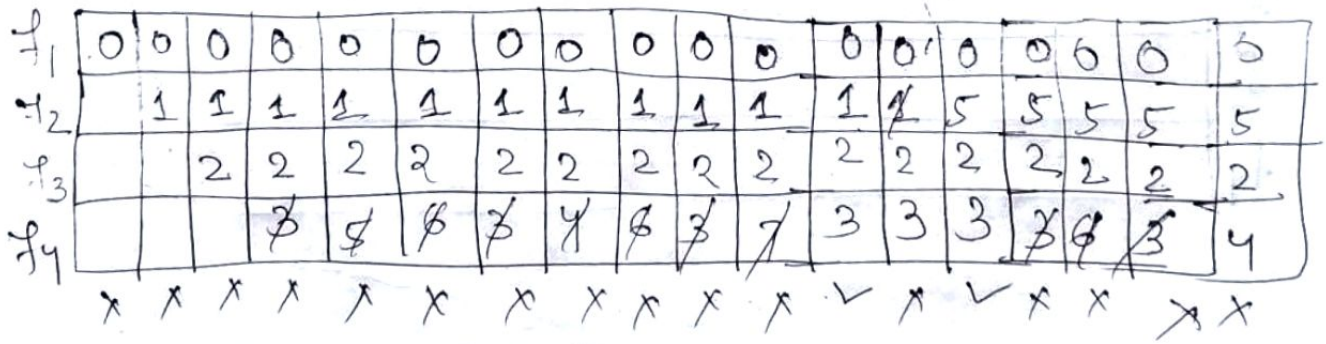
Page fault =
hit =

MRU Page Replacement : Most recently used.

String = 0, 1, 2, 3, 5, 6, 3, 4, 6, 3, 7, 3, 1, 5, 3, 6, 3, 4



Page fault = 16
hit = 2



Page fault = 16
hit = 2

Least Frequently used (LFU)

Reference string :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2

f_1	7	7	7	2	2	2	7	4	4	3	3	3	3	3	3
f_2		0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_3			1	1	1	3	3	3	2	2	2	2	2	1	2
	x	x	x	x	hit	x	hit	x	x	x	hit	hit	hit	x	x

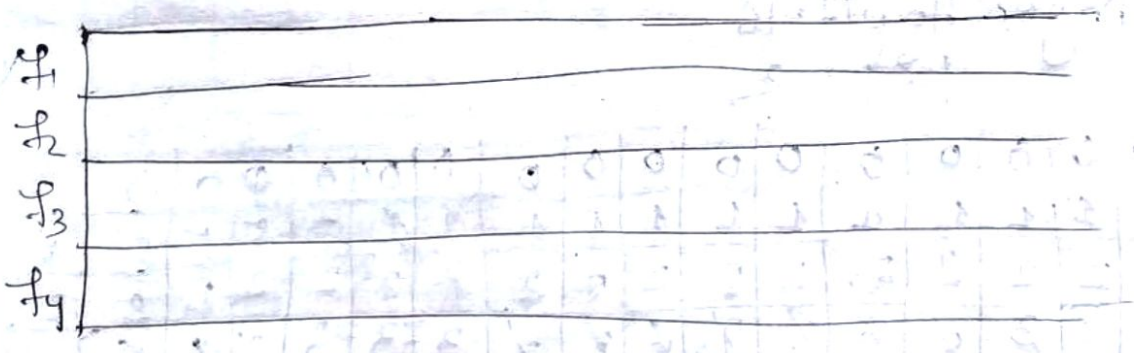
Frequencies

7 = 0 1 0 , 0 = 0 1 2 3 4 , 1 = 0 1 0 , 2 = 0 1 0 1

3 = 0 1 0 1 2 , 4 = 0 1 0

Page fault = 10

hit = 5



Most Frequently used (MFU)

Reference string :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2

f_1	7	7	7	2	2	2	2	2	2
f_2		0	0	0	0	3	3	3	3
f_3			1	1	1	1	0	4	4
	x	x	x	x	hit	x	x	x	hit

Frequencies

7 = 0 1 0

0 = 0 1 2 0 1

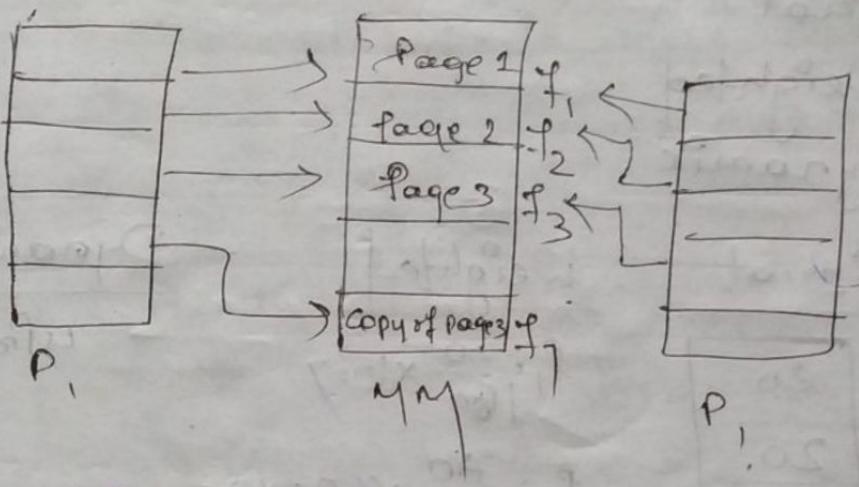
1 = 0 1 0

2 = 0, 1, 2

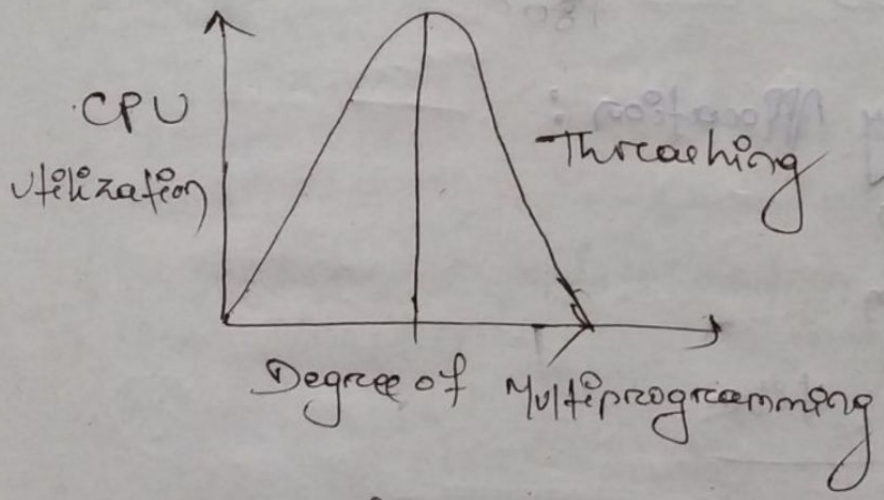
3 = 0 1

4 = 0 1 2

Copy-on-Write



Thrashing



Locality Model

Working-set-Model

D - no. of demand of page
m - Available of frame

D > m : Thrashing
D <= m : Not thrashing.

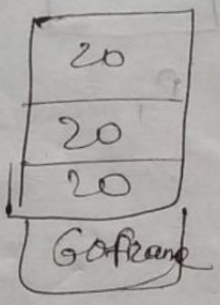
Page Fault Frequency :-

Frame allocation

- Equal
- Weighted
- Dynamic

Equal

P_1 - 20pg
P_2 = 60pg.
P_3 = 100pg.



Weighted

$P_1 = \frac{20}{180} \times 60 = 7$

$P_2 = \frac{60}{180} \times 60 = 20$

$P_3 = \frac{100}{180} \times 60 = 33$

Dynamic

↳ priority

Kernel Memory Allocation :

(1) Buddy System
(2) Slab allocation

- Memory mapped file

File Management

File is the related document collection. It is stored in secondary memory

Types of File

- text → .txt
- Source → .src, .java
- Object → .exe.

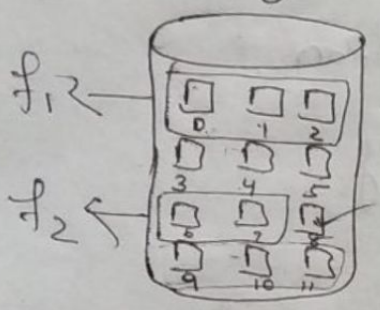
Attributes

- Name
- Location
- Size
- Type
- Date : create, update, modified
- Protection (Authen., non authenticated.
- Repositioning / move
- Truncating

Access Method

- Sequential
  - read next
  - Write next
  - Rewind
- Direct
  - Read n
  - Write n
  - jump to n
  - Reset
- Indexed
  - ↳ Pointer

Contiguous file allocation:



disk

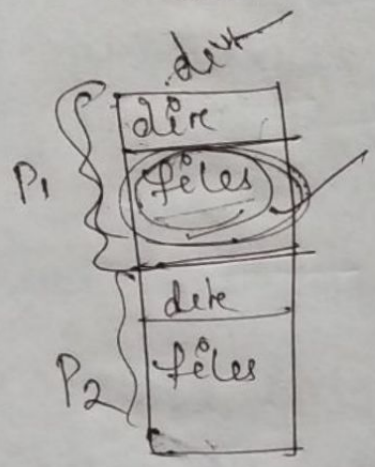
filename	Start block	length
f_1	0	3
f_3	6	2

Directory → a table.

blocks.

random, sequential.

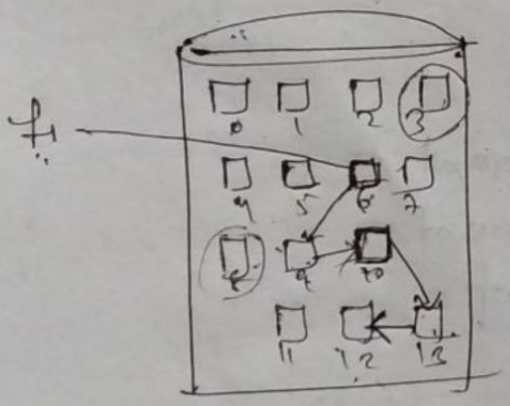
6+1 = 7



Dis - Frag.

8
7+1

Linked file



Linked list

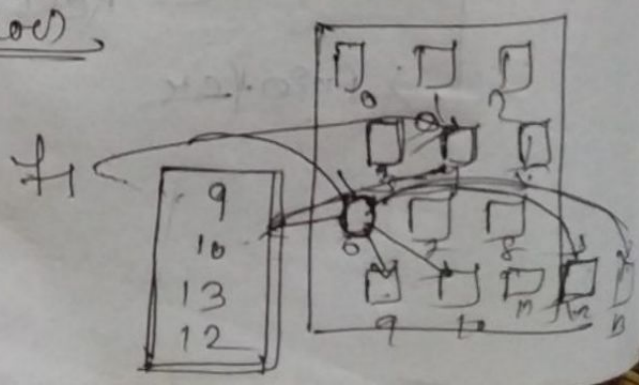
file	Start	end
f_1	6	13

dis random access ✗

f_2

Indexed file allocn

file	Index block
f_1	4

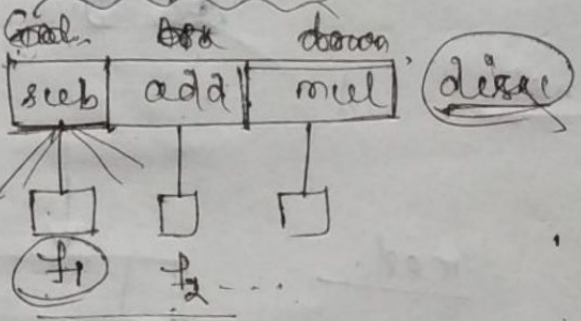


Directory structure

- Single level
- Two level
- Tree structure
- Acyclic graph structure. (2)

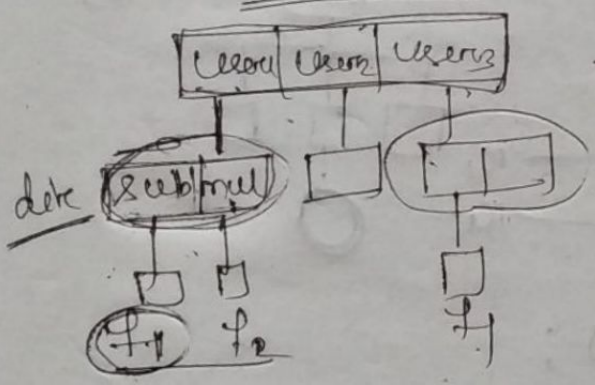
SL

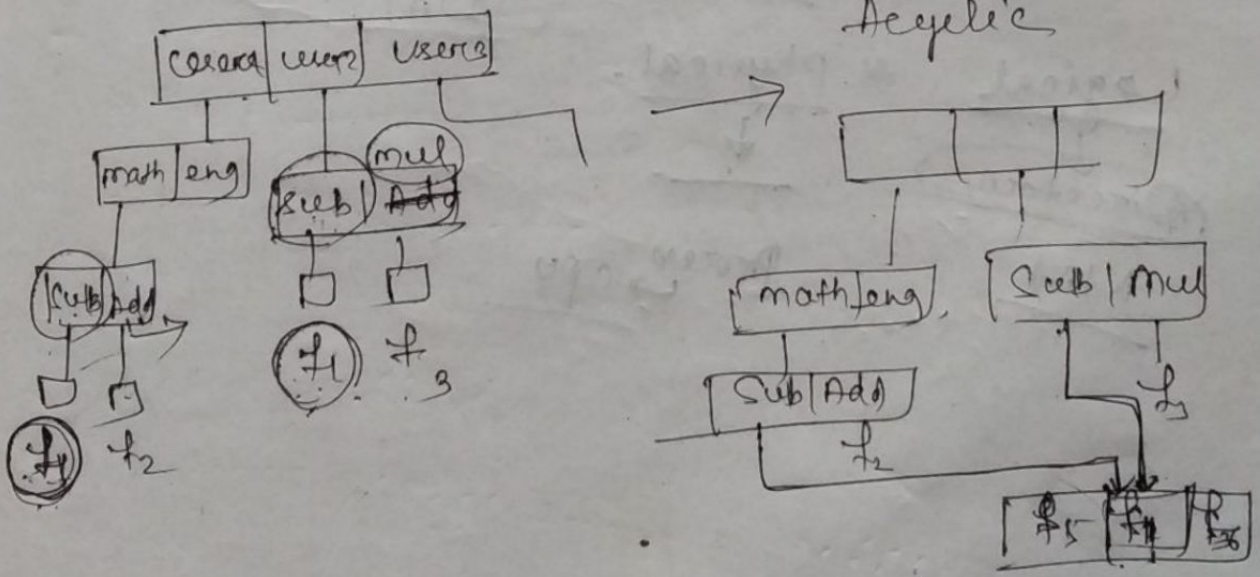
Gallery



1000

User level

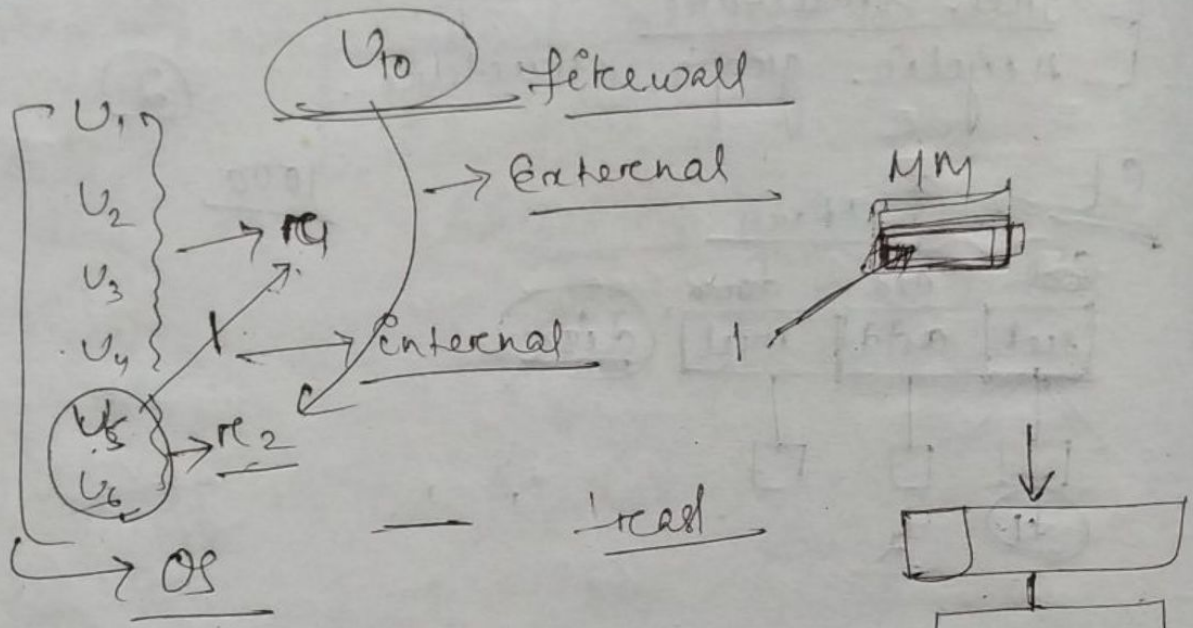


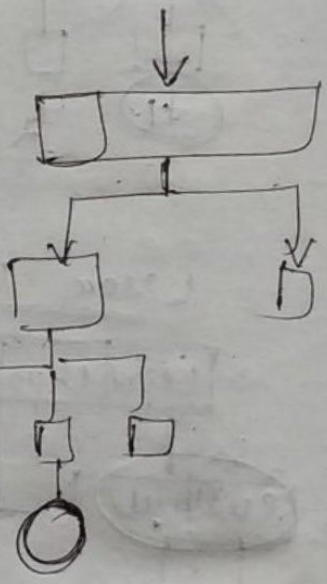


Protection & Security

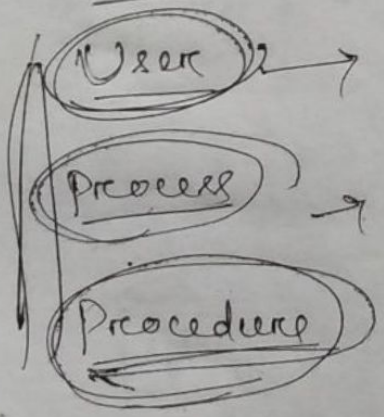
↓ Internal

↓ External.





Domain



$U_1 \rightarrow r_1$

$U_2 \rightarrow r_3$

$U_3 \rightarrow$

$P_1 \rightarrow U_1 \rightarrow r_1$

$U_2 \rightarrow r_2$

obj. fcn. ( ) ;

↗ ↘ UA

Logical & physical.

↓

Directory

MM

Process → CPU