Operating System

29 Jan 2024

It is an interface b/w user and the machine

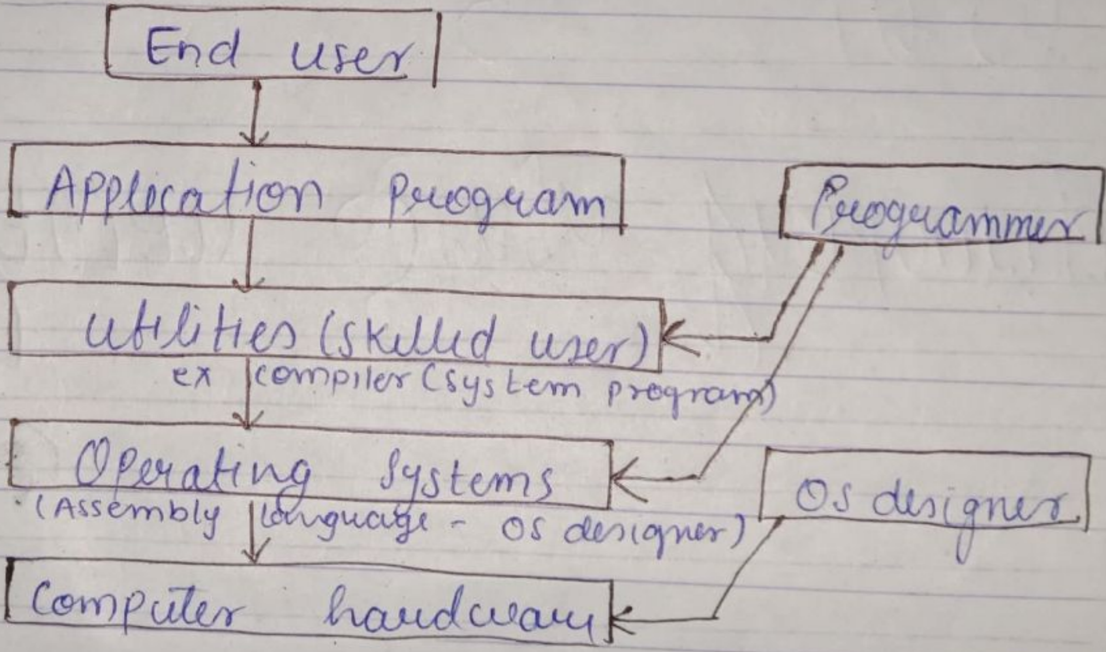
Definition:- It is a program that controls the execution of application program and acts as an interface between the user of the computer and the computer hardware.

⇒ Objectives / functions of OS

→ Easy to learn (provide convinience)

→ Provide efficiency.

⇒ Logical structure / Hierarchical structure of OS



→ utilities :- It is a set of system program provided for controlling the computer hardw-are are referred as utilities and they implement frequent used function that access the program, that manages the files and the input output devices.

The most important system program is OS which hides the details of the hard-ware from the programmer and provides to the programmer.

★ Editors for creating file

→ Unix editor :- VI editor, cat command

→ Windows :- Notepad, Cmd

→ Dos :- COPYCON command, EDIT

⇒ Services of OS :-

→ Program creation :- Such as editors and debuggers to assist a program in creating programs and the services are in the form of that are actually part of OS but are accessible throughout OS.

→ Program execution :- A no. of tasks need to be performed to execute a program instruction and data must be loaded into main memory, IO devices, files must be initialised and other resources must be prepared. OS does this for the user.

→ Controlled access to file :- It is done by OS in case of a system for multiple simultaneous users by providing protection mechanism to control access to the files.

→ System access :- for public or shared system OS controls or protects the resources of data from unauthorised users and it resolves the conflicts.

→ Error detection and response :- Such as internal and external hardware errors ex:- memory error, device failure malfunction software errors like arithmetic overflow, access to forbidden memory etc.

→ Error detection and response :- Such as interna and external hardware errors ex:- memory error, device failure malfunction software errors like arithmetic overflow, access to forbidden memory etc.

→ Accounting of resources and monitors performance such as response time

- Turn around time :- when we submit a program to the computer
- Response time :- time between the submission and the first response of the program

*Note :- Both these times must be minimum.

→ Os is a resource manager where a computer is a system of resources for the movement, storage and processing of the data and for control of the function. The OS is responsible for managing the given resources

Evolution of Operating System

→ Serial processing :- (1940 - 50) here the programs directly interacted with the other computer no OS was there, program was written in ~~machine code~~. assembly language.

⇒ Drawbacks :-

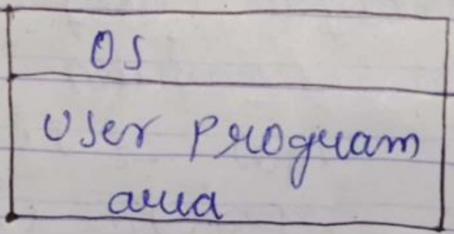
- Scheduling :- most installation used a sign up time to resolve machine time the user signed

Scanned by CamScanner

a block of time if it completed earlier then the rest of the time it had was wasted and if it might run in problem then it is of short of Time prior to resolving its problem

- we have to write in assembly language (need to learn the codes of assembly language)
- Set up time :- single program called a job could involve loading the translator another function which took lot of time in seting up the program to run

→ Simple batch system :- Early machine ~~code~~ were very expensive and therefore it was important to maximize the machine ~~code~~ use the wasted time cost by scheduling the set up time was unacceptable.



Main memory

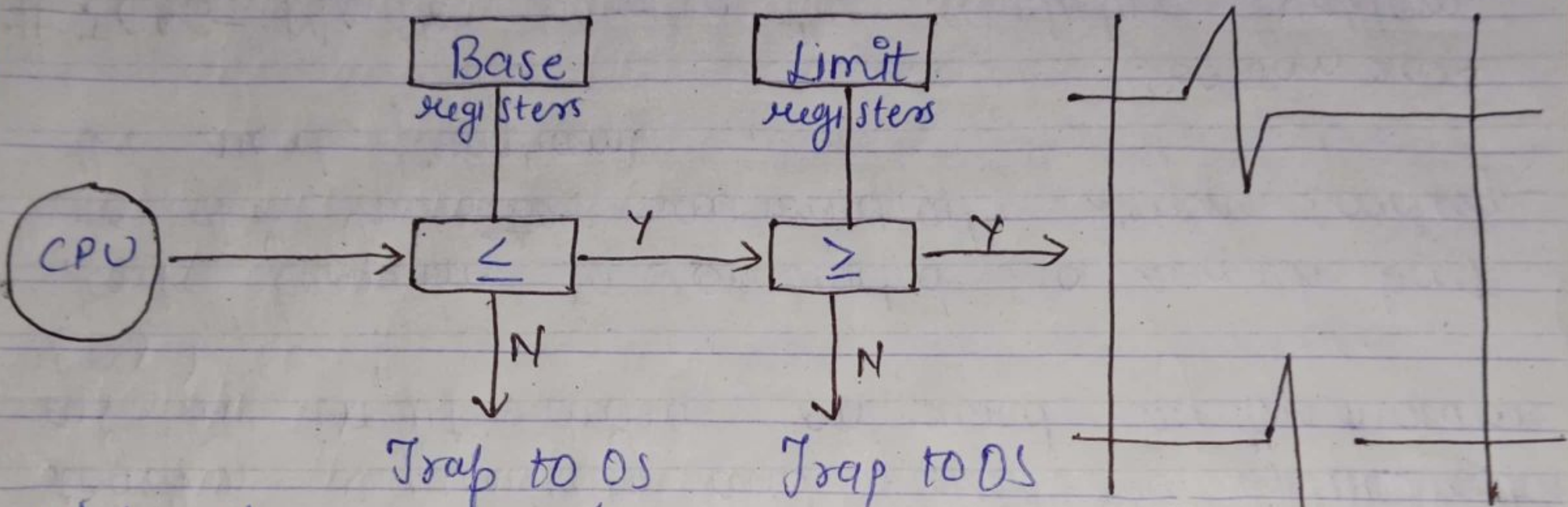
- To overcome this problem the batch Os is developed, the first Os is developed by general motor for IBb 701 machines
- The central idea behind the simple batch processing scheme was the use of a piece of software known as Monitor with this type of Os the user no longer had direct access to the machines
- The user submitted the jobs on cards and tapes to the operator which batched the jobs together sequentially and placed the entire batch on an input device on used by the monitors (OS).
- Each program is constructed to branch back to the monitor automatically begins loading the next program.

30 Jan 2024

JCl :- Monitor handles the scheduling problem of job by using a special type of programming language JCl which provides instruction to the monitor such as END, LOAD, RUN etc.

⇒ Hardware features desirable for memory protection are as follows:-

→ Memory protection:-



While the user program is executing it must not alter the memory area containing the monitor. If such an attempt is made the processor hardware should detect the error and transfer control to the monitor. The monitor would then abort the job print out an error message and will load in the next job

→ Timer:- It is used to prevent a single job from monopolizing the system. The timer not alter the memory area containing the monitor. If such an attempt is made the processor hardware should detect the error and transfer control to the monitor. The monitor would then about the job print out an error message and will load in the next job

→ Timer :- It is used to prevent a single job from monopolizing the system. The timer

operating system set at the begining of each job if it expires an interrupt occur an controls returns back to the monitor

→ Privilaged instruction :- Instructions that can be executed only by the Monitor if the processor encounters such an instructions while executing an user program an error interrupt occurs Among the privilaged instruction are I/O instructions so that the monitor retains control of all I/O devices if a user program wishes to perform I/O it must request that the monitor performs the operation for it. If a privilages instruction is encountered by the processor while it is executing a user program the processor hardware considered it as an error and transfores the control to the monitor

→ Interrupt :- Earlier computers didn't have this facility, this features gives OS more flexibility in relinquishing the control to regain control from user program, with a batch OS machine time alternates between execution of user program and the execution time of the monitor

NOTE :- There are two sacrifices

① Some main memories are given over to the monitor (OS)

② Some machine time are consumed by the monitor

→ Both are overheads but simple batch OS used of the computers

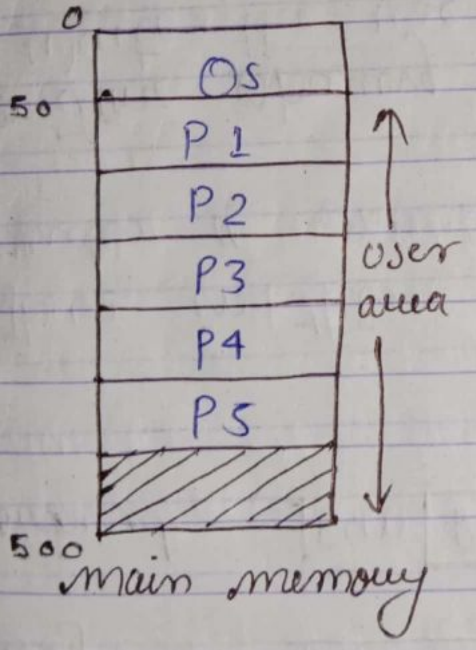
→ Multi program batch OS :-

- Even the automatic job sequencing provided by a simple batch OS the processor is often idle. As the io devices are slow as compared to the processor.
- Solution to ~~that~~ this problem can be that there must be enough memory to hold the OS (permanently residing monitor) and more than one user program now when one job needs to wait for io, the processor can switch to the other job which likely is not waiting for io this process is known as

there mus
OS (permanently residing mo
than one user program now when one
job needs to wait for io, the processor can
switch to the other job which likely is not
waiting for io this process is known as

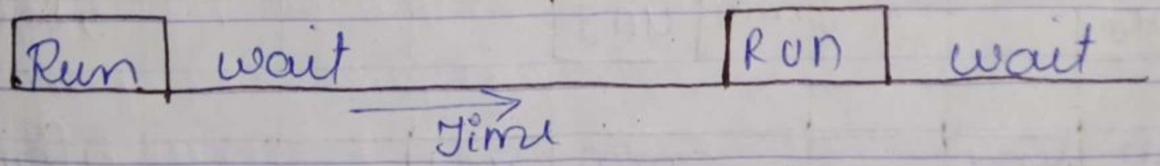
multiprogramming
NOTE:- Keeping multiple programs in the user
area is multipleprogramming
* Two types of program
- I/O bounds
- CPU bounds

NOTE:- There mix should be such that CPU
idle time is minimum possible which
is reduced for the efficiency

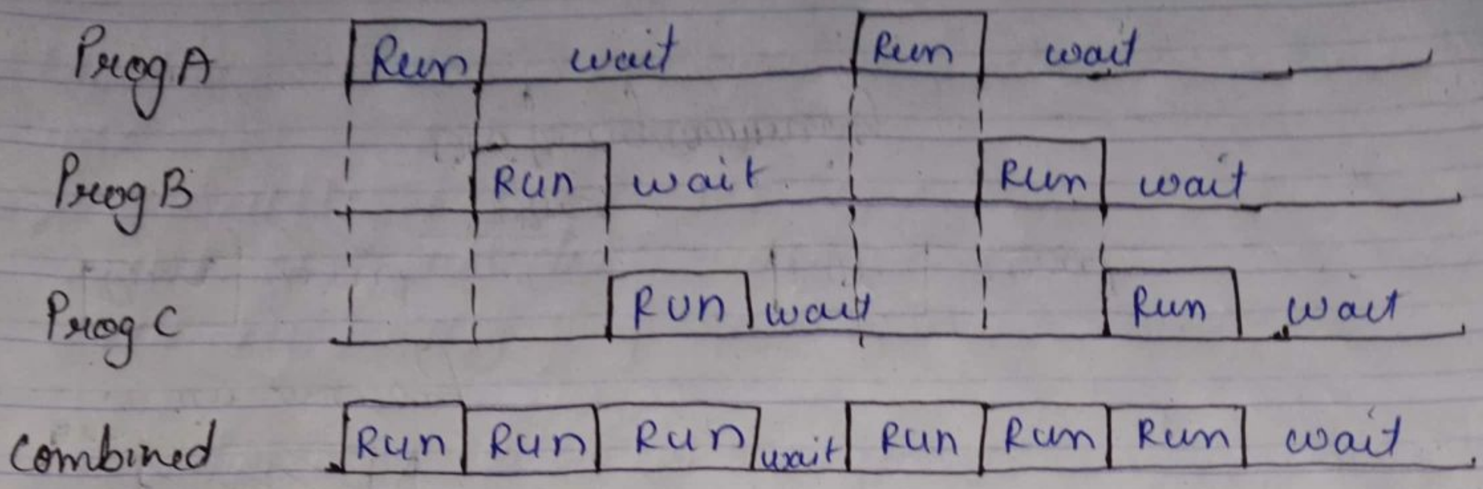


main memory

$$\begin{tikzcd}
 & \text{Types of program} \arrow[dl] \arrow[dr] & \\
\text{Io Bound} & & \text{CPU Bound}
\end{tikzcd}$$



Uniprogramming



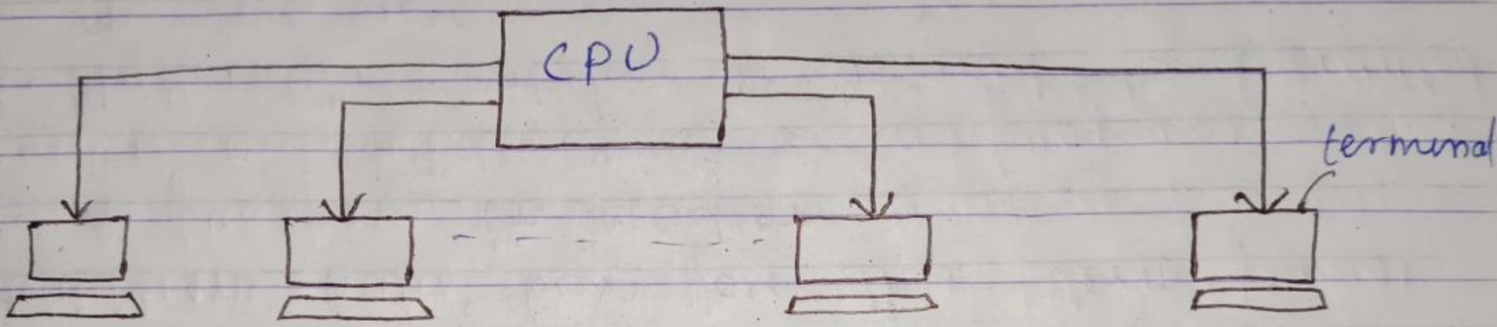
Multi programming

* Thus in this jobs are run concurrently under a multiprogramming Os [coexisting]

→ Time sharing System (or Multi Tasking/interactive

- Write the use of multiprogramming, batch processing can be quite efficent
- However for many jobs it is desirable to provide of mode with which the user interacts directly with the computer (eg for transaction etc)
- To overcome this, just as multiprogramming allows the processor to handle multiple jobs at a time (there can be starvation in

jobs if there is very long CPU bound program for other programs but when the processor's time is shared among multiple users it is called Time sharing system



Time sharing system

- If there are n users actively requesting service at one time each user will see on the average only 1/n of the effective computer speed, not counting OS overhead.
- Difference, although both use multiprogramming

	Batch Multiprogramming	Time sharing / Multi Tasking
Principle objective	Maximum processor use	Minimum response time
Sources of instr. to OS	Jobs control language instr. provided with the job	Commands entered at the terminal.

Problems with Time sharing & multiprogramming are:-

→ If multiple jobs are in memory then they must be protected from interferring with each other, by modyfying each other's data

→ Contention for resources such as printer, storage devices etc. occurs which has to be handled.

★ Process:- The concept of process is fundamental to the structure of OS. It was first used by designers of Multi

- It is more general term than job Many definitions have been for this term process
  1. A program in execution
  2. The animated spirit of a program
  3. That entity that can be assigned to & execute on processor.

★ 3 Major lines of computer system development Created problem in timming & synchronization that contributed to the development of the concept of the process.

- multiprogramming batch operation
- Time sharing
- Real-Time transaction system

* Ist :- Multiprogramming was designed to keep the

★ 3 Major lines of computer system development Created problem in timming & synchronization that contributed to the development of the concept of the process.

- multiprogramming batch operation
- Time sharing
- Real-Time transaction system

* Ist :- Multiprogramming was designed to keep the processor & I/O devices, including storage devices simultaneously busy to achieve maximum efficiency.

* 2nd was general purpose time sharing. Here the key design objective is that the system be responsive to the needs of the individual users & yet for cost reasons, be able to simultaneously support many users.

* 3rd line of development has been real-Time transaction processing systems. (ex:- airline reservation system)

★ With several jobs in progress at any one time there were four main causes of errors:

Scanned by CamScanner

→ Improper Synchronization :- eg a program initiates an I/O read & must wait until the data are available in a buffer before proceding. In such cases, a signal from some other routine ~~[illegible]~~ is required. Improper design of the ~~scrolling~~ signalling mechanism can result in signals being lost/duplicate signals being recieved

→ failed mutual exclusion :- It is often the case that more than one user or program will attempt to make use of a shared resources at the same time.

There must be some sort of mutual exclusion mechanism that permits only one routine at a time to perform a transaction against a portion of data. The implementation of such mutual exclusion is difficult to verify as being correct under all possible squences of events

→ Non determinate program operation : When program share memory & their execution is interleaved by the processor, they may interfare with each other by over writing common memory areas in unpredictable ways

→ Deadlocks :- It is possible for 2 or more program to be hung up waiting for each other

5 Jan 2024

Parallel system or (Multiprocessor system or tightly caupled system)
It is a computing architecture where multiple processor or central processing units (CPUs) work together simultaneously to execute tasks These processor are closely interconnected and share resources such as memory and I/O devices. This architecture allows for concurrent processing of multiple tasks, which can lead to increased performance and throughput compared to a single processor system

⇒ Symmetric multiprocessing model :- In which each process runs are identical copy of the OS

⇒ Assymetric multiprocessing :- in which each processor is assigned a specific task. A master processor controls the system & other processor either look to the master for or have predifined tasks (master - slave).

* Reasons for including such system is the advantage of increased throughput :

# Distributed systems (or loosely couple system)

→ This is the recent trend in computer system that is to distribute computation among several processors

→ In this the processor do not share memory instead each processor has its own local memory.

→ The processors communicate with one another through various communication lines, such as high speed buses or telephone lines

→ [The processor in the system may vary in size & function & such as commenication lines such as high - speed buses or telephone lines] ⇒ wrong

→ The processors in the system may vary in size & function (such as small microprocessors workstation, mini computers & large general-purpose computers) & they are referred to by a no. of different names such as sites, nodes, computer etc.

⇒ Reason for building the distributed system:

→ Resource sharing :- It provides mechanism for sharing files at remote sites, processing information in a distributed database, printing files at remote sites, using remote specified h/w devices.

→ Computational speedup :- by dividing computations into subcomputations that can seen concurrently This movement of jobs is called load sharing

→ Reliability :- If one site fails, another can be us

→ Communication :- eg - electronic mail etc.

Real time system :- Another form of a special-purpose OS, is the real-time system.

* It is used when there are rigid time requirement on the operation of a processor or the flow of data, thus is often used as a control device in a dedicated application
* A real time OS has well designed, fixed time constraints processing must be done with in the defined constraints or the system will fail

$$\begin{tikzcd}
 & \text{Type of real-time system} \arrow[dl] \arrow[dr] & \\
\text{Hard} & & \text{Soft}
\end{tikzcd}$$

Hard

These system guarantee that critical tasks complete on time

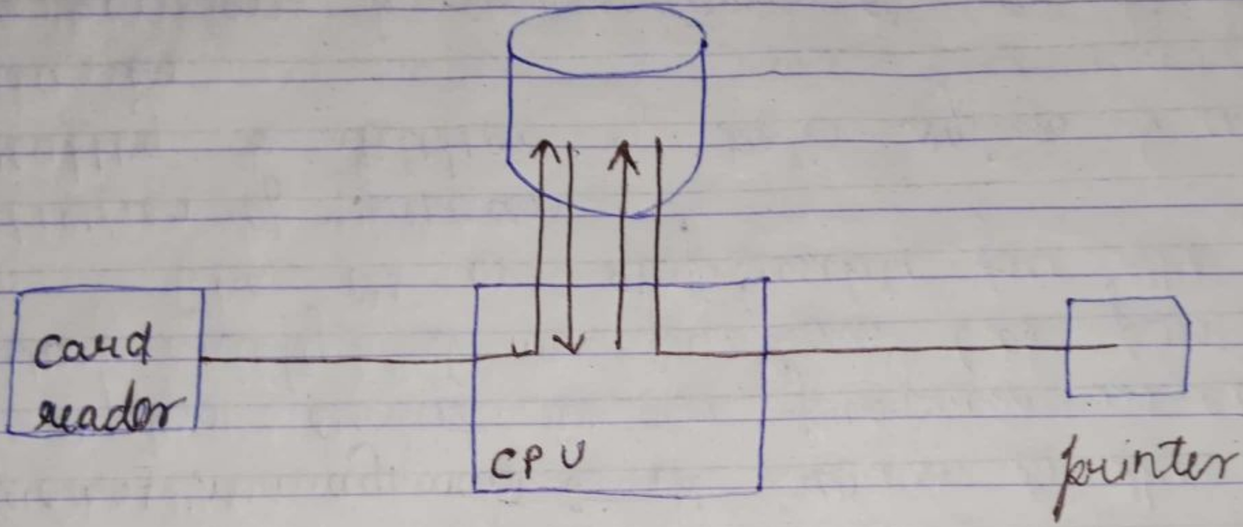
Soft

Systems are less restricted type where critical real time gets priority over other tasks & retains that priority until it complete

* Virtual memory is never found on R.T.S

⇒ some definition

→ Spooling (simultaneous peripheral operation on-line)

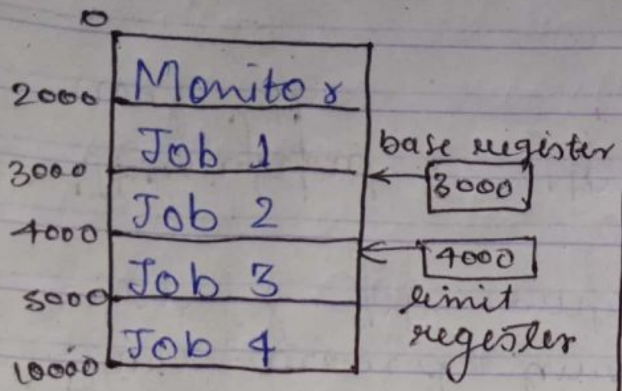


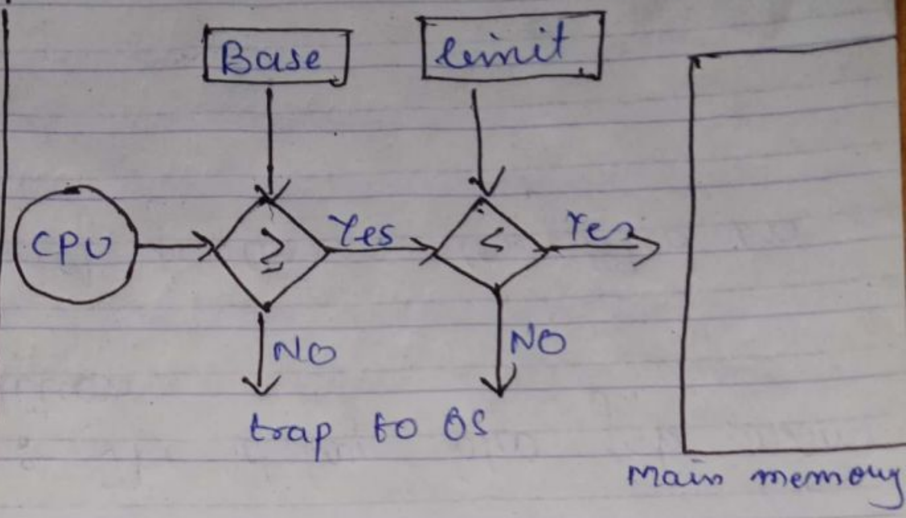
spooling uses the disk as a huge buffer, for reading as far ahead as possible on I/p devices & for storing o/p files until the o/p devices are able to accept them

→ Turn around time :- The delay b/w job submi-ssion & job completion

→ Throughput :- NO. of processes completed in unit time

Memory protection :-





H/w address protection with base & limit register

functions of OS (Component of OS)

→ Process management :- A process is the unit in a system or it is a program in execution (active entity) & a program (is passive)
The OS is responsible for the following activities of process
- Creating & deletion of both system & user processes
- suspension & resumption of process
- provision of mechanism for process synchronization
- provision of mechanism for process communication
- provision of mechanism for ~~process~~ deadlock handling

→ Main memory management :- OS is responsible for

- suspension & resumption of process
- provision of mechanism for process synchro-

nization
- provision of mechanism for process communicatio
- provision of mechanism for ~~process~~ deadlock handling

→ Main memory management :- OS is responsible for
- Keeping track of which part of memory are currently being used & by whom
Deciding which processes are to be loaded into memory when memory space becomes available
Allocates & deallocates memory space as needed

→ file management :- File is a collection of related information designed by its creator
- The OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage devices to define a logical storage unit the file. The OS maps files onto physical media & access these files via the storage devices
- Responsibilities of OS wrt file management -

1. Creation & deletion of files
2. Creation & deletion of directories
3. Support of primitives for manipulating files & directories
4. Mapping of files onto secondary storage
5. The backup of files on stable (non volatile) storage media

~~File system man~~

→ I/O system management :- OS hides the peculiarities of h/w from the user. Device drivers know about the h/w

→ Secondary storage management :- responsibilities of OS are
- free space management
- storage allocation
- Disk scheduling

Process description & control

→ The design of the OS reflects the requirements that it is intended to meet. And all the multiprogramming OS's from single user system

→ The design of the OS reflects the requirements that it is intendend to meet. And all the multiprogramming OS's from single user syste

(as MVS) that can support several thousands of users, are built around the concept of the process.

→ Thus the major requirement that the OS can meet with reference to the process is:

- The OS must interleave the execution of a no of processes to maximize processor use & providing resonable response time
- The OS must allocate resources to processes adopting some policy avoiding deadlocks

The OS must support interprocess communication & user creation of process.

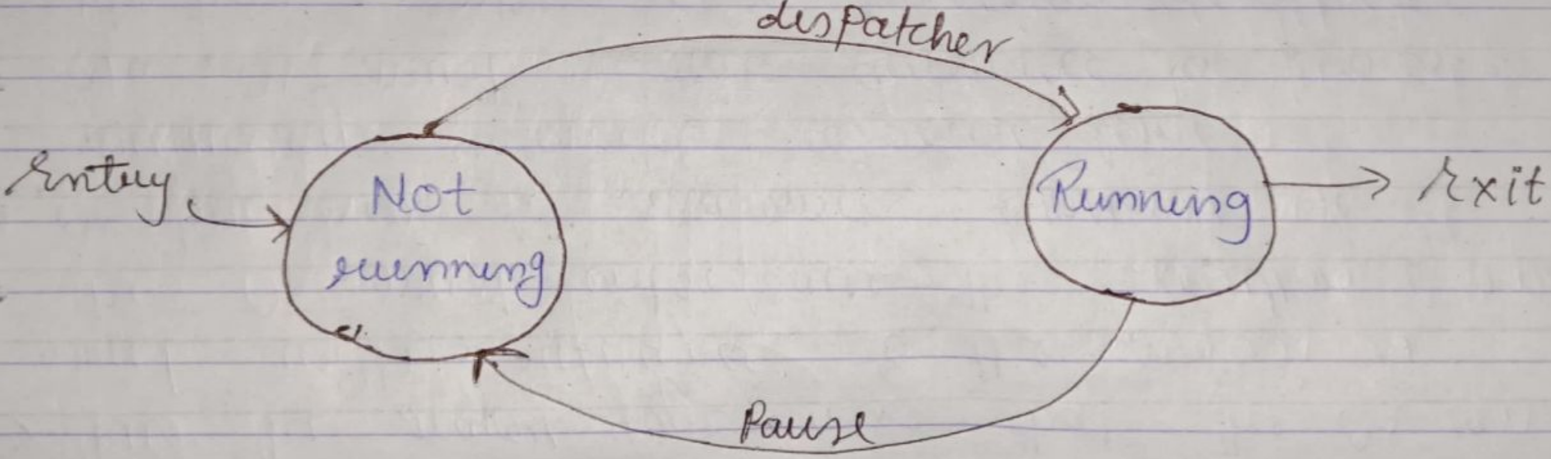
⇒ Process :- is a program in execution (active entity)

⇒ Process states :- The execution of an endividual program is referred to as a process, or task

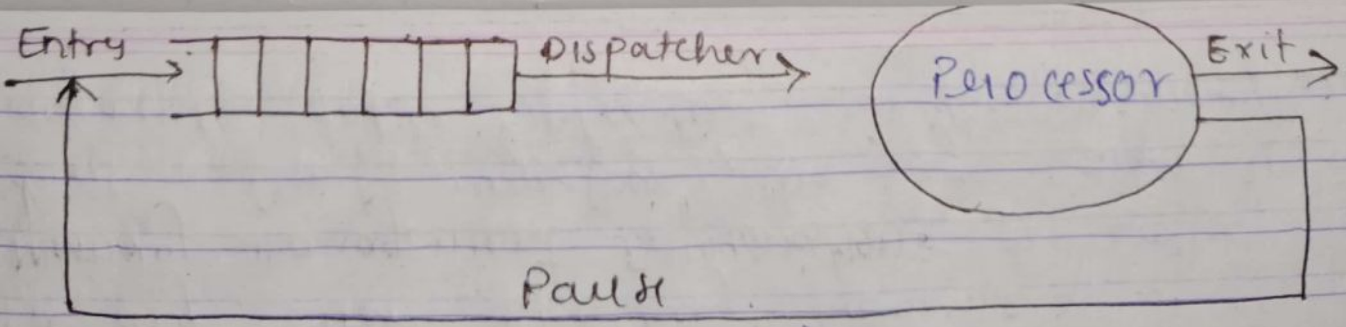
* The principal function of a processor is to execute m/c code instruction residing in the main memory.

* The enstructions are provided in the form of programs containing sequence of instructions

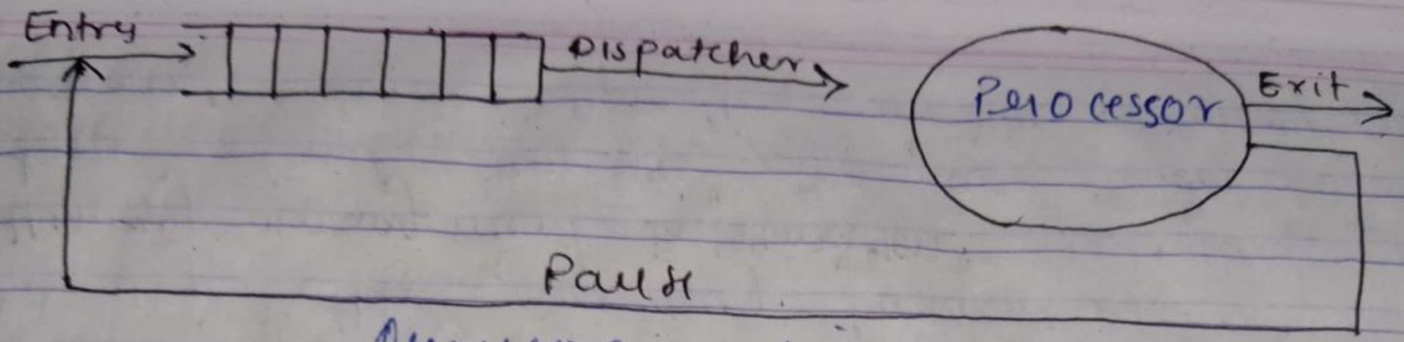
* The os allows a process to execute for a fixed slice of time and then is interrupted [when preserve any single process from monupolizing processor time] & the dispatcher program moues the processor from one process to another.
* Any Os creates a new process & puts it in the queues of not running process which is a linked list of data blocks in which each block represent one process
* The dispatcher selects a ~~program~~ process from the queue to execute, after the time out / completion of a process



Transition diagram



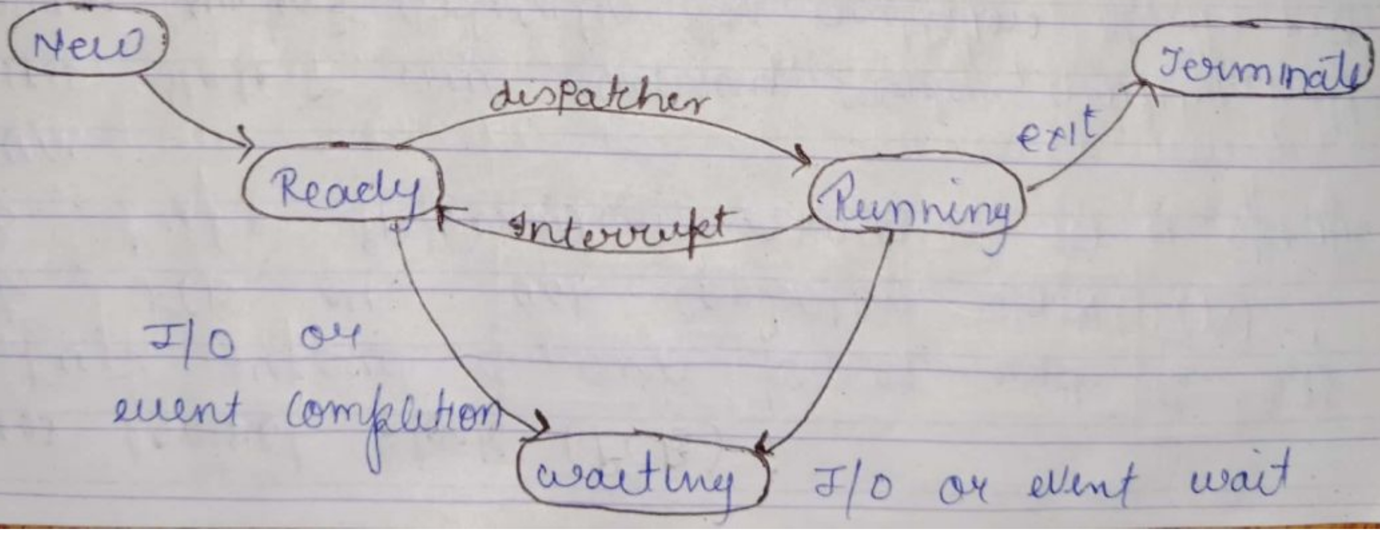
Queuing diagram



Queuing diagram

⇒ Process has 5 states :- When process executes it changes state, it may be in one of the following state

- New :- The process is being created
- Running :- being executed
- Waiting :- process is waiting for some events to occur
- Ready :- the process is waiting to be assigned to a processor
- Terminated :- process has finished execution



⇒ Process control block (PCB) :

- OS Keeps record of each process with the help of its PCB or task Controller block (TCB)
- PCB Keeps information associated to a specific process.

* Process state :- new, running, ready, waiting, halted.

* Program Counter :- indicates the addition of the next instruction to be executed for the process.

* CPU registers :- The registers vary in no. & type depending on computer architecture. They include accumulator index register plus any condition code information

Along with the program counter, this state information must be saved when an interrupt occurs to allow the process to be continued correctly afterward.

* CPU scheduling information :- which includes a process priority pointer to scheduling queues, & any other scheduling parameter

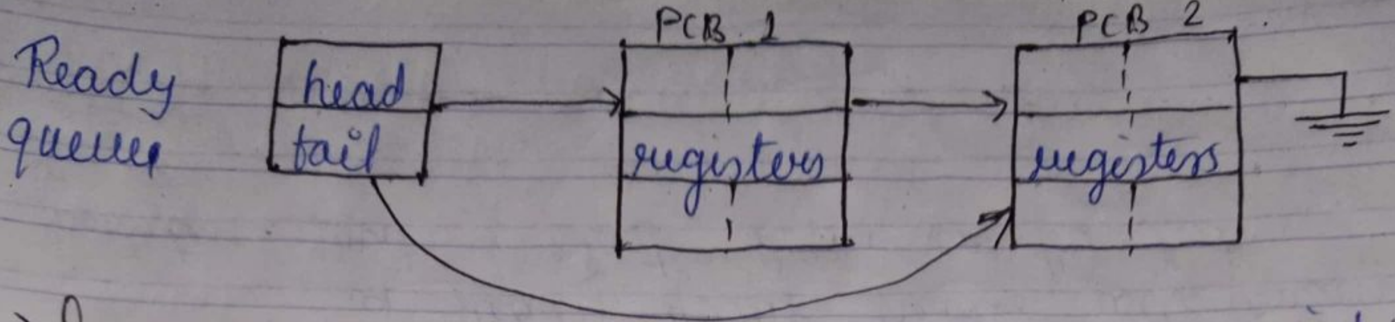
* Memory management information :- the value of the base limit registers, the page tables on the segments table depending on the memory system

uses OS.
* Accounting information :- includes the amount of CPU its real time used, time limit, account no job or process.
* I/O states information :- which includes the lists of I/O devices allocated to this process, a list of files etc.

06 FEB 2024

Process scheduling :-
The aim of multiprogramming is to have some process running at all times, to maximize the utilization.

⇒ scheduling queues :-
↪ Job queue :- formed when process enter the system
→ Ready queue :- made key process residing in main memory & are ready & wait to execute (in linked list form)
A ready queue header will contain pointers first & last PCB's in the list



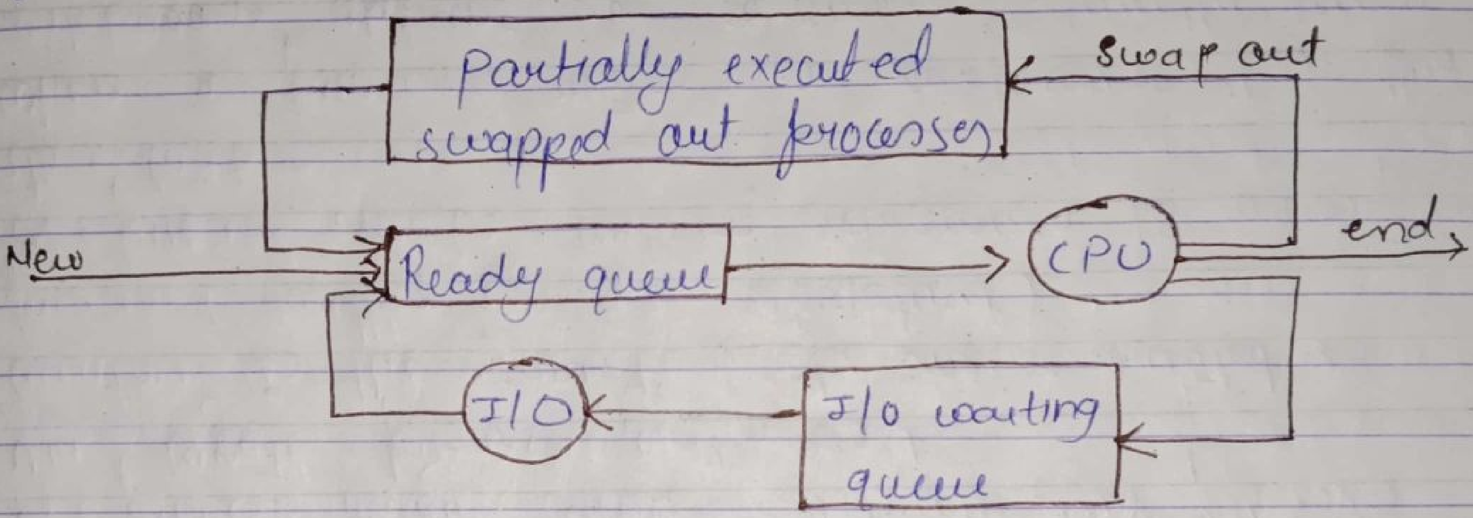
→ Device queue :- is the list of processes waiting for a particular I/O device
when a process terminates it is removed from queues & has its PCB & resources allocated

⇒ schedulers :- A process migrates b/w the various scheduling queues throughout its life time. The OS must select the processes by the appropriate schedulers

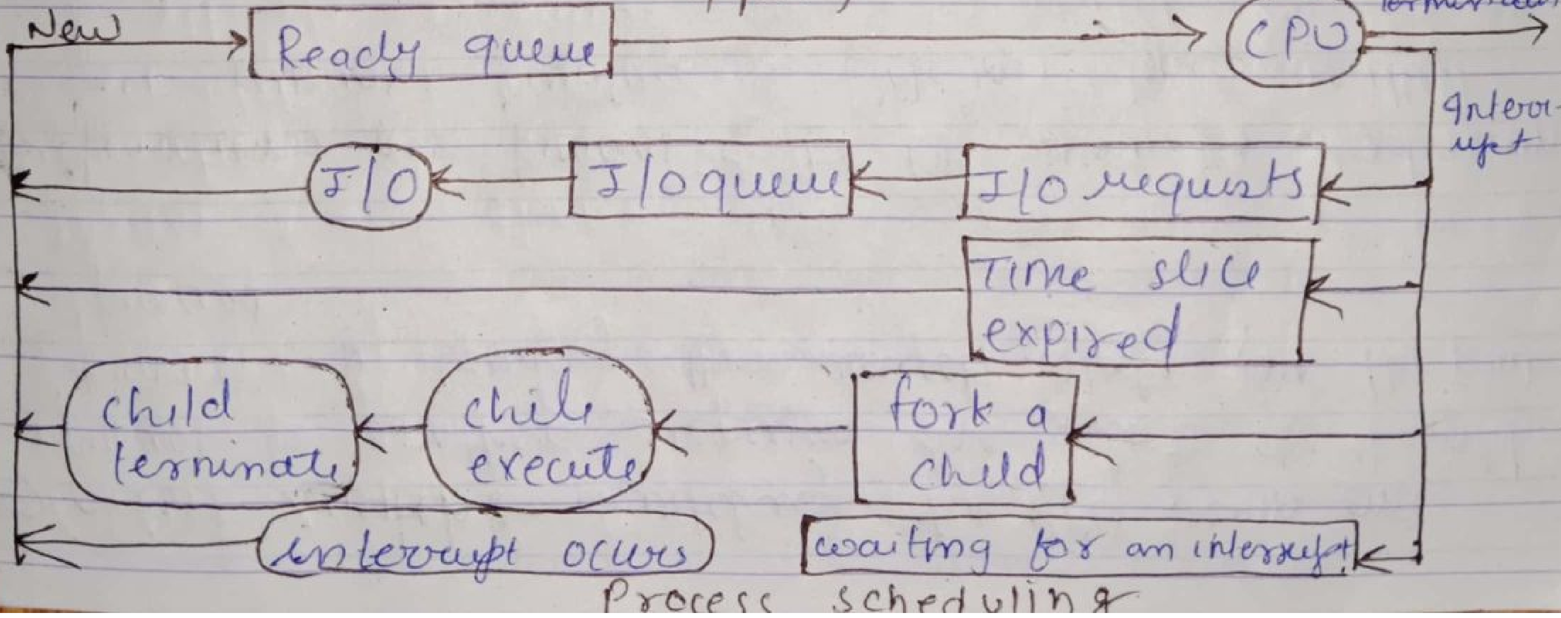
→ Long term schedulers :- It selects processes from the job pool (stored in hard disk) & loads in the memory for execution. It controls the degree of multiprogramming

→ Short term scheduler / CPU scheduler) :- selects process from among the process that are ready to execute & allocate CPU to one of them

→ Medium - term scheduler (swapping) :-
Time sharing system often have no long term scheduler they simply put all process in memory
The time - sharing os introduces intermediate of scheduling that is the medium - term schedul



Swapping



Process scheduling

⇒ Context switch :- Switching the CPU from one process to another requires the state of the old process & loading the saved state for the new process

⇒ Operations on process :- The OS provides mechanism for process creating & deletion dynamically

→ Process creation :-

- A process may create several new processes via create process system call.
- Creator is the parent & the new created is process (further in the structure)
- A process needs certain resources to accompany its task
- When a process creates a subprocess then this subprocess obtains its resources directly from the OS or it may be constrained to a subsequent of the resources of its parent process either by sharing or partitioning
- This restricting of a child process to a subset of the parent's resources, prevents any

process from overloading the system by creating too many sub processing processes.

- when a process creates a new process, 2 possibility exists in term of execution :

1. The parent continues to executes concurrently with its children
2. The parent waits until some or all of its children have terminated.

⇒ Process termination :-

→ Normal completion → Time limit exceeded
→ Memory unavailable → Bounds violation. (of memory)
→ Protection error (access to forbidden file)
→ Arithmetic error (divided by zero)
→ Time overrun (process has waited longer for an event to occur)
→ I/O failure (eg: unable to find a file)
→ Invalid instruction (eg process attempts to executes a non-existing instruction)
→ Privilaged instruction.
→ operator / os intervention (os terminates due to deadlock)
→ Parent terminates → Parent requests

Reasons for process suspension :-
1. Swapping 2. Pre empted
3. Parent process requests

Process termination :- takes place when it finishes executing its last statement & asks the OS to delete it by using the exit system call & all its resources are deallocated by the OS
* when a process creates a new child process the identity of the newly created process is passed to the parent
* A parent may terminate the execution of one of its child for a variety of reasons
• The child has executed its usage of some of the resources it has been allocated
• The task assigned to a child is no longer needed
• The parent is exiting, thus OS does not allow a child to continue if its parent terminate
• ~~The child has executed its usage of some of the~~

Co Operating Processes

#

* A process is independent if it cannot affect or be affected by the other processes executing in the system.
* A process is co-operating if it can affect or be affected by the other processes executing in the system / processes that share data with other processes are cooperating process

⇒ Reason for an environment that allows process cooperation :-

→ Information sharing :- eg shared files
→ Computational speedup :- If we want a particular task to run faster we must break it into subtasks, each of which will be executing in parallel with the others
→ Modularity :- dividing the system function into separate processes
→ Convienience :- even an individual user may have many tasks to work on at one time. eg :- a user may be editing, printing & compiling in parallel.

Inter process communication (IPC) :-

* Cooperating processes can communicate in a shared memory environment which requires that these processes shares a common buffer pool, & that the code for implementing the buffer by explicitly written by application programmes.

OR

* IPC is for the OS to provide the means for cooperating processes to communicate with each other via an interprocess communication facility.
* IPC provides a mechanism to allow processes to communicate & to synchronize their actions
* IPC is best provided by a message system

⇒ Basic structure :- An IPC facility provides at least the 2 operations

$$\begin{tikzcd} \text{operations} \arrow[r] \arrow[dr] & \text{Send (message)} \\ & \text{Receive (message)} \end{tikzcd}$$

message sent by a process can be either fixed or variable size

- fixed sized messages :- their physical implementation are straight forward but makes programming more difficult
- Variable size messages :- requires a more complex physical implementation but programming task is simpler.

* If P & Q are 2 processes, who wish to communicate then there must be a communication link b/w them

Unidirectional link :- is if each process connected to the link can either send/receive operations

- Direct/ Indirect communication
- Symmetric / Assymetric communication
- Automatic / Explicit buffering
- Send by copy / send by reference
- fixed sized / Variable sized messages

→ Direct / Indirect :-

$$\begin{tikzcd}
 & \text{① for direct} \\
\text{Naming} \arrow[ur] \arrow[dr] & \\
 & \text{② for Indirect}
\end{tikzcd}$$

Process that want to communicate must have a way to refer to each other either directly/ Indirectly

① Direct Communication :- each process that wants to communicate must explicitly name the recipient /sender of the communication.

The primitives are send & received

— Send (P, message) [send a message to P]

— Receive (Q, message) [receive a message from process Q]

★ Symmetric addressing :- A communication link for the following properties :-

↪ A link is established automatically & the processes need to know only each other's identity to communicate

↪ A link is associated exactly 2 processes

↪ b/w each pair of processes, there exists exactly one link

↪ Link may be unidirectional but is usually bidirectional.

This scheme exhibits a symmetry in addressing, that is both the sender and the receiver processes have to name each other to communicate.

symmetry in addressing

, that is both the sender and the receiver processes have to name each other to communicate.

★ Assymetric addressing :- A variant of previous scheme. Here only the sender names the recipient is not required to name the sender.

eg :- — Send (P, message) [send a message to process P]

— Receive (id, message) [receive a message from a process, the id is set to the name of the process with which communication has taken place]

Disadvantages :- in both schemes is the limited modularity of the resulting process definitions (eg: changing the name of the process may necessiate examining all other process definition. All references to old names must be modified to new names)

② Indirect communication :-

→ The message are sent to & received from mailboxes/ port. A mailbox can be ~~viewed~~ viewed abstractly, as an object into which messages can be placed by processes & from which messages can be removed.

→ each mailbox has a unique identification Process can communicate with some other processes via a no. of different mail boxes.

→ Two processes can communicate only if the processes have a shared mailbox

→ The primitives are

- Send (A, message) [send a message to mailbox A]
- Receive (A, message) [receive a message from mailbox A]

→ In this scheme the communication link has the following properties

- A link is established b/w a pair of processes only if they have a shared mailbox
- A link may be associated with more than 2 processes
- b/w each pair of communicating processes, there may be a no. of different links each links corresponds to one mailbox
- A link may be unidirectional or bidirectional

→ A mailbox may be owned either by a process or by the

→ If the mailbox is owned by a process then it is attached ~~a~~to it & is defined as part of the process.

→ The

...no. of ...municating processes, there ...
of different links, each link
Corresponds to one mailbox
• A link may be unidirectional or bidirectional
→ A mailbox may be owned either by a process or by the
→ If the mailbox is owned by a process then it is attached ~~a~~to it & is defined as part of the process.
→ The owner (unique) can only receive message through mailbox
→ The user can only send messages to the mailbox
→ When a process that owns a mailbox terminate, its mailbox also desappears & thus all the users are notify who send message to this mailbox that it is no longer exists (through exception handling)
→ Subprocess of a parent process can share a mailbox
→ Mailbox that are owned by Os has an existance of its own, they are independent & not attached to only process.
→ When a mailbox is no longer accessible by any process than it is recorded by OS [garbage collection]

★ Buffering / Capacity of a link :-

A link has some capacity that determines the no. of message that can reside in it temporarily.

$$\begin{tikzcd} & \text{Type of Queue} \arrow[dl] \arrow[d] \arrow[dr] & \\ \text{zero Capacity} & \text{Bounded capacity} & \text{unbounded capacity} \end{tikzcd}$$

① zero capacity :- • The queue has length 0
- The sender must wait until the recipient receives the message
- The 2 processes must be synchronized

② Bounded Capacity :- • The queue has finite length n.
- ~~The sender must wait until the recipient receives~~. If the link is full then the sender must be delayed until space is available in the queue

~~Bounded~~ Capacity :- • The queue ... of
n.
• ~~The sender must wait until the recipient receives~~. If the link is full then the sender must be delayed until space is available in the queue

③ Unbounded capacity :- • Queue has finite length
• Any no. of messages can wait in it
• The sender is never delayed

Exception Condition :-
① Process termination :- either a sender or receiver may terminate before a message is processed.
2 cases :-
ⓐ A receivers process P may wait for a message from a process Q ~~should~~ that has terminated. If a no action is taken the P is blocked for ever, thus the system may either terminate P or notify it that Q has terminated

ⓑ Process P may send a message to process Q that has terminated, in automatic buffering scheme there is no harm, P will continue but if there is no buffering & P is waiting for acknowledgement then either P must be notified of Q's termination or terminated itself.

② Lost messages :- from P to Q in the n/w lines there are 3 methods for delaying this event.

a) The OS must detect it & resend this message.

The sending process must detect it & retransmit the message, if it wants to do s

The OS must detect it, then notify the sending processing the message has been lost, then the sending process can proceed as it chooses

Scanned by CamScanner