UNIT-3

OPARATING SYSTEM

(OPARATING System)

PAR-A

1. Types of oparating System.
2. OS Concepts.
3. OS Oparations
4. OS Structures
5. OS Services
6. User Oparinting Systems Interface.

Introduction to System calls. and. Types of System calls.

PAR-B

1) process Management:- 1) process Concepts. Proces state diagram, process control block,

2) proces scheduling :- 3) Inter proces Communicaton. 4) Threads and Thread issue. 5)

3) scheduling.

a) Basic Concepts
b) Scheduling coriteria.
c) Scheduling Algorithms

1. Types of Operating Systems:

The different types of operating systems are,

1. Batch operating System:-

(i). An operator takes similar Jobs having the same requirement and group them into batches.

(ii). It is the responsibility of the operator to sort the Jobs with similar needs

(iii) It is very difficult to guess or know the time required for any Job to complete processors of the Batch Systems. know how long the Job would be when it is in queue.

(iv) multiple users can share the batch systems.

(v) The idle time for the batch system is very less.

(vi) it is easy to manage large work repeatedly in batch systems.

2 Time-Sharing Systems: operating:-

(i) Each task is given some time to Execute so that all the tasks work smoothly.

(ii) Each user gets the time of cpu as they use a single system These systems are also known as multitasking ~~users also~~. Systems

(iii) The Task can be from a single user & different users also

(iv) The time that each task gets to Execute is called quantum

(v) After this time interval is over OS switches over to the next task. CPU idle time can be reduced.

3. Distributed operating System:-

(i) Various Autonomous Interconnected computers communicates with each other using a shared communication network.

(ii) Independent Systems posses their own memory unit and CPU. These are referred to as loosely coupled Systems or distributed Systems.

(iii) These Systems's processors differ in size and function.

(iv) The major benifit of working with these types of the operating System is That it is always possible That one users can access The file or Software which are not actually present on his System but some other System connected within This Networks. i.e: remote access is Enabled within The devices connected in That Network.

Failure of one will not effect the other network Communicate as all systems are Independent from each other.

) Electronic mail Increses The data Exchanged Speed.

) Since resources are being shared, computation is highly fast and durable.

) Load on host Computer reduces

These Systems are easily scalable as many Systems can be easily added to The Networth. Delay in data processing

Network operating System:-

These Systems runs on a Server and provide The capability to manage data, users, groups, Security, applications, and Networking functions

(ii) These types of operating systems allows shared Access of files, printers, Security, applications and other networking functions over a small private network.

(iii) One more important aspect of network operating systems is that all the users are well aware of the underlying configuration, of all other users within the Network, These computers are also known as tightly coupled Systems.

5. Real-time operating System:-

(i) These types of operating systems serve - real time systems

(ii) The time interval required to process and respond to input is very small. This time interval is called response time

(iii) Real-time Systems are used when there are time requirements that are very strict like missile Systems, air traffic control systems robots, etc.

2. Operating Systems Concepts

Operating System:

A programme that acts as an intermediary between a user of computer and the computer hardware

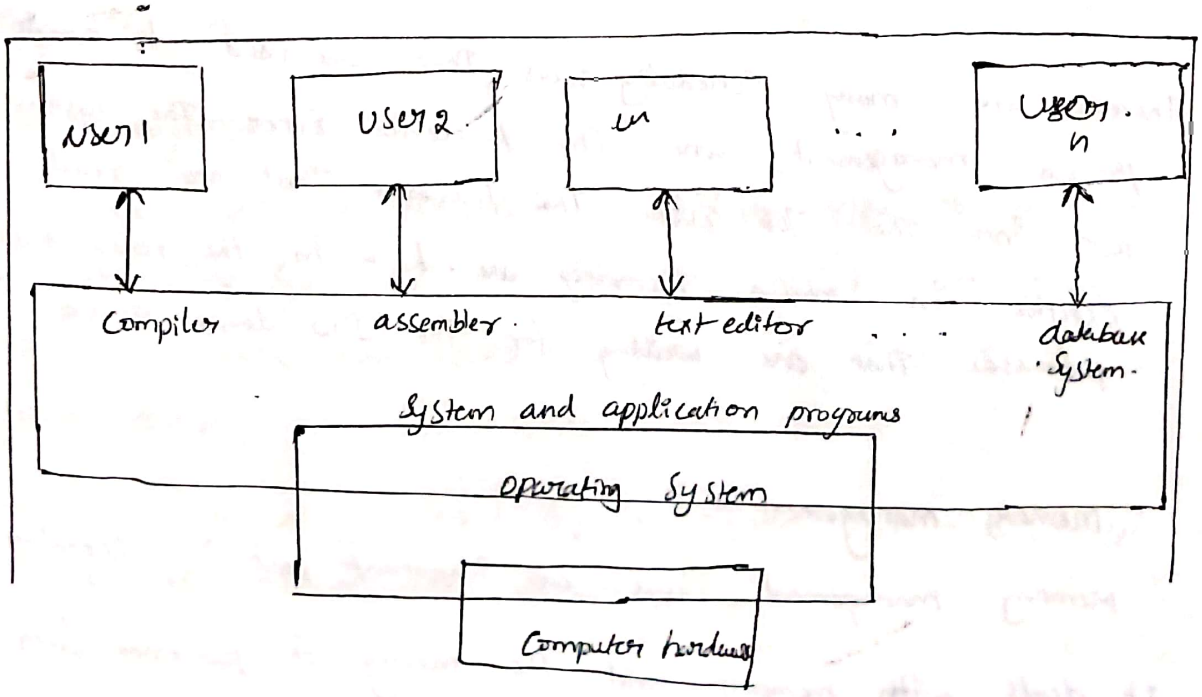
Computer System Components (Hardware, OS, Application programes, User)

Hardware: It provids basic computing resources (CPU, memory, I/O devices.

Operating System: It controls and coordinates the use of the hardware among the various application programe for the various users.

Applications programs: It defines the ways in which the system resources are used to solve the computing probelems of the users (Compilers, database systems, video games, business programs).

Users:- (people (single/multi), machine, other computers).



3. Operating System operations (Functions / Tasks)

The main operations performed by the Operating System are,

a. process management
b. Memory management
c. Device management
d. File management
e. Security management

a. Process management:

The Operating System is responsible for managing the processes ie assigning the processor to a processor at a time. This is known as process Scheduling

The different algorithms used for process scheduling are FCFS (First Come First Served), SJF (Shortest Job first), priority Scheduling, round robin scheduling etc.

There are many Scheduling queues that are used to handle process process management when the processes Enter the System. They put into the Job queue. The processes that are ready to Execute in main memory are kept in the ready queue. The processes that are waiting for the I/O devices queue.

b. Memory management :-

Memory management plays an important part in Operating System It deals with memory and the moving of processes from disk to primary memory for Executing and back ag...

The Activities performed by the operating System for memory management are,

The operating System assigns memory to the processes as required. This can be done using best fit, first fit and worst fit algorithms.

All the memory is tracked by the operating System. ie it notes what memory parts are in use by the processes and which are Empty

The operating System deallocating memory from processes as required. This may happen when a process has been terminated & it it no longer needs the memory.

c. Device management :-

There are many I|O devices handled by the operating Systems Such as mouse, keyboard, disk drive, etc.

There are different devices drivers that can be, Connected to the operating System to handle a Specific.

The device controller is an interface between the devices and the device driver. The user applications can Access all I|O devices using the devices drivers which are device Specific codes.

d. File management :-

Files are used to provide a uniform view of data Storage by the operating System. All the files are mapped onto physical devices that are usually non volatile so data is Safe in the case of System failure.

The files can be accessed by the System in two ways. ie: Sequential access and direct access.

Sequential Access: The information in a file is processed. Sequential

The records are

accessed one after another Eg: Editors, compilers.

Direct access:

In direct access or relative access, the files can be accessed

in random for read or write.

operations. The direct access model is based on the disk model

of a file, since it allows.

random access.

e. Security management:-

Security refers to providing a protection system to computer system

& resources such as cpu, memory, disk, softwares, programs, &

most importantly data / information stored in the computer system

if a computer programs is run by an unauthorized user, then

he/she may cause ~~Serve~~ severe damage to computer or

data stored in it. So a computer system must be

protected against unauthorized access; malicious access to

system memory, viruses, worms etc.

4. Operating Systems Structures

Multiprogramming :-

A programming operating System can run Several programmes on a Single processor machine.

If a Single application has to wait for I/O transfer in a multiprogramming operating system. other programmes are always ready to use the CPU. As a result, numerous Jobs can Share. The CPU's time. However, in a multiprogramming operating System. it is not predefined. That Their Jobs would be executed. Simultenously "

Advantages :-

1. it provides high CPU utilization
2. It has a shorter response time.
3. It can assign priority to The Jobs.

Multitasking :-

Multitasking means working on multiple tasks Simultenously, Such as using your Computer while listening to music. Also using a browser, Search for Something on The internet and Create a word document That is your assignment.

It appears That all of The tasks are taking place at the Same time. It is not all of The tasks happening Simultenously; The processor moves between them at Such a fast pace. That we feel realtive. they are happening Simultaneously.

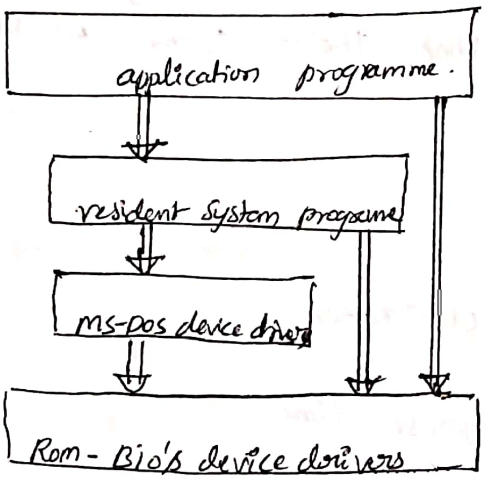
Multitasking is Similar to multiprogramming where the CPU is assigned to a processor for a Specified period of a time. ie: it runs Various programmes at The Same time.

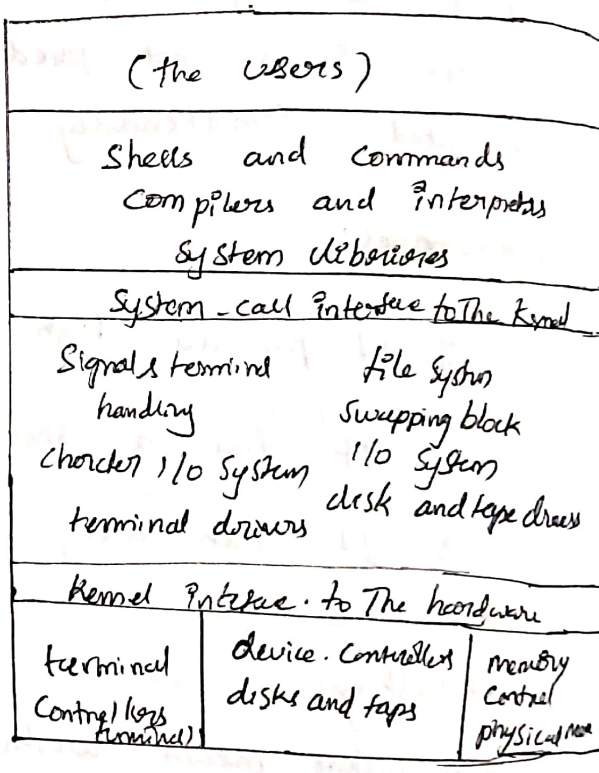
Operating System Structure :-

for efficient performance and implementation an OS should be partitioned into separate Sub Systems, each with carefully defined tasks, inputs outputs, and performance characteristics.

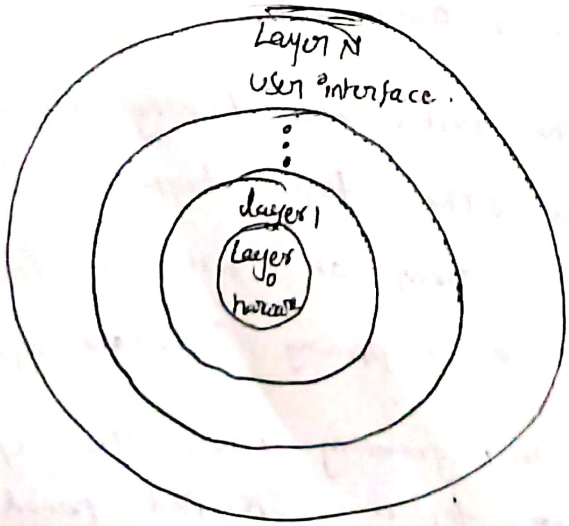
These Sub Systems can then be Arranged in various architechural configurations :

MS - DOS Layer Structure.





Layered Operating System.



5. Operating System Services

The various services of operating systems are,

a. User Interface.

b. ~~(User Interface)~~ — programme Execution ✓

c. I/o operations

d. File-System Manipulations

e. Communications

f. Resource allocation

g. Accounting

h. Error detection

i. protection and Security.

a. User Interface.

It is means by which users can issue commands to the Systems. Depending on the System. These may be a Command-line interface. Eg. Sh, csh, tcsh, etc...), a GUI interface. (eg. windows, X-windows, KDE, Gnome, etc...) & a batch command Systems.

b. programme Execution:-

The OS must be Load a programme into RAM, run the programe and terminates the programe, either normally & abnormally.

c. I/o Operations:-

The OS is responsible for transfering data to and from I/o devices, including keyboards terminals, printers, and storage devices.

d. File System manipulation.

In addition to raw data storage, the OS is also responsible for (~~maintaing~~)(maintaining) directory and Sub directory structures, mapping file names to specific blocks of data storage, and providing tools for navigating and utilizing the file systems.

e. Communications:-

Inter process communication (IPC), either between processes running on the same processor, or between processes running on separate processors or separate machines. May be implemented as either Shared memory or message passing (or Some Systems may offer both)

f. Resource Allocation:-

CPU cycles, main memory, storage space, and peripheral devices are some resources which are managed with generic systems and others with very carefully designed and specifically tuned systems. customized for a particular resource and operating ~~Error~~ Environment.

g. Accounting:-

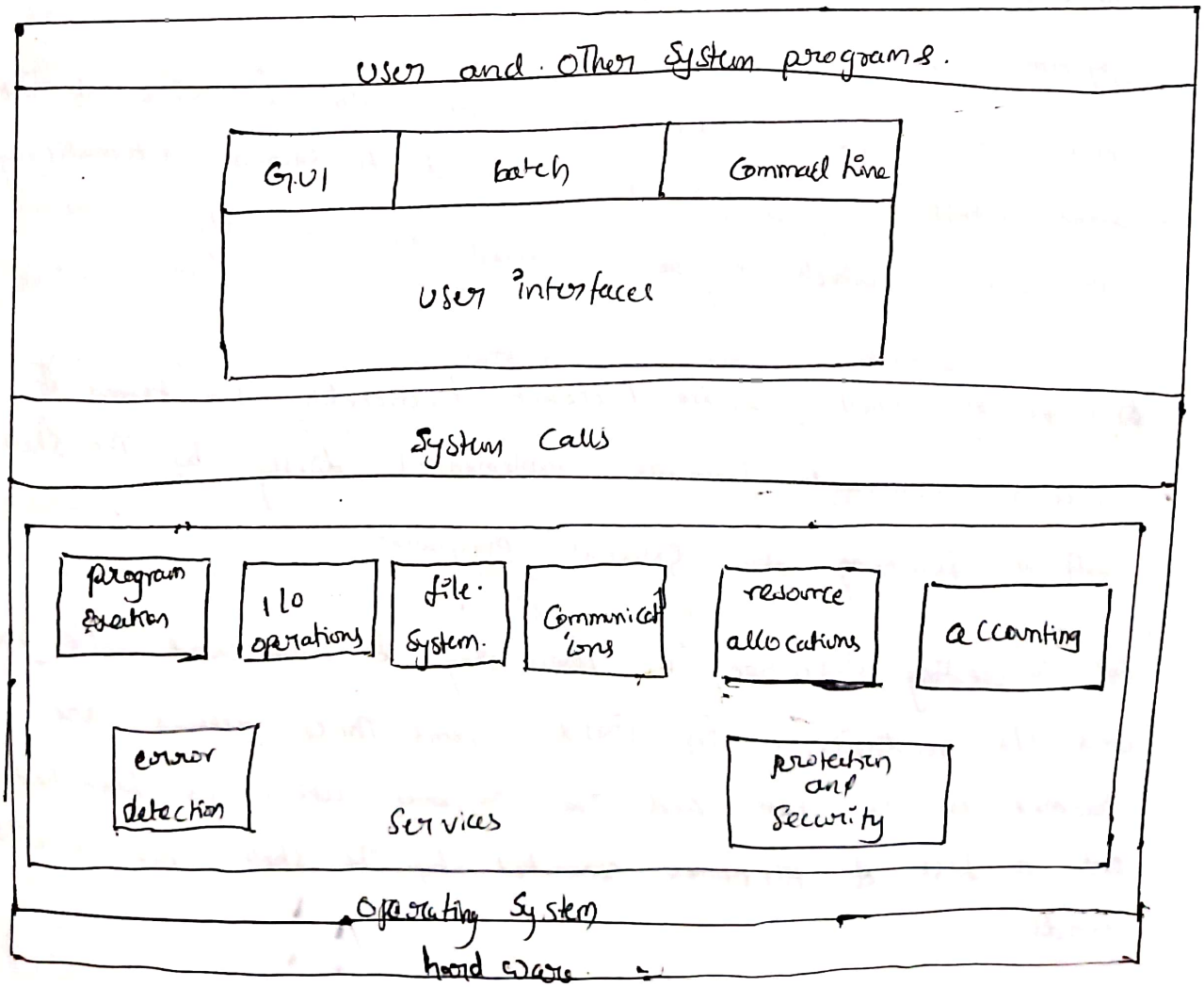
Keeping track of Systems Activity and resource usage, either for billing purpose or for Statistical record keeping that can be used to optimize future performance.

(h) error detection:-

Both hardware and software errors must be detected and handled appropriately. Some systems may include complex error avoidance & recovery systems, including backups, RAID drivers, and other redundant systems. Debugging and diagonostic tools aid users and administrators in tracing down the cause of probelms.

(i) protection and Security:-

Authentication, ownership, and restricted access are obivious parts of this system which prevent harm to the system and to resources.



6. User operating - System Interface.

User operating - System interface can be of,

a. Command interpreter [Character user interface. (CUI)

b. Graphical user interface. (GUI)

a. Command Interpreter / character user Interface (CUI) :-

It process the users request which is given as a command, and Lanuches the requested programmes

In Some Systems the CUI may be incorporated directly into the Kernel

More Commanly the CUI ~~may be~~ is a Separate programe. That Lanches once the user logs in or otherwise. Accesses the System.

UNIX, for Example, provides the user with a choice of different shells, which may either be configured to launch automatically at login, or which may be changed on the fly.

Different Shell provides Different functionality, In terms of Certain commands that are implemented direlty by the Shell without lanching any External programes.

An interesting distnction is processing of wild card file naming and I/o re-direction. on UNIX Systems those details are handled by the Shell, and the programes which is Lanched see only a list of file names generated by the Shell From the wild cards.

On a DOS System, The wild cards are passed along to The programs, which can interpret The wild cards as The program.

b. Graphical User Interface:- (GUI):-

GUI is Implemented as a desktop metaphor, with file folders, trash cans, and resource icons.

Icons represent some item on The System, and respond accordingly when The icon is Activated.

First developed in early 1970's at Xerox PARC research facility.

In Some Systems The GUI is Just a front end for activating a traditional command line interpreter running in The background. In others The GUI is true graphical shell in its own right

Because Mice and Keyboards are impractical for Small mobiles devices, These normaly use a touch Screen interface today. That responds to various patterns of Swipes or "gestures", when These first came out They often had a physical keyboard and/or trackball of Some kind built in, but today a Virtual Keyboard is more commonly implemented on The touch Screen.

7. Introduction to System Calls & Types of System Calls

System Calls :-

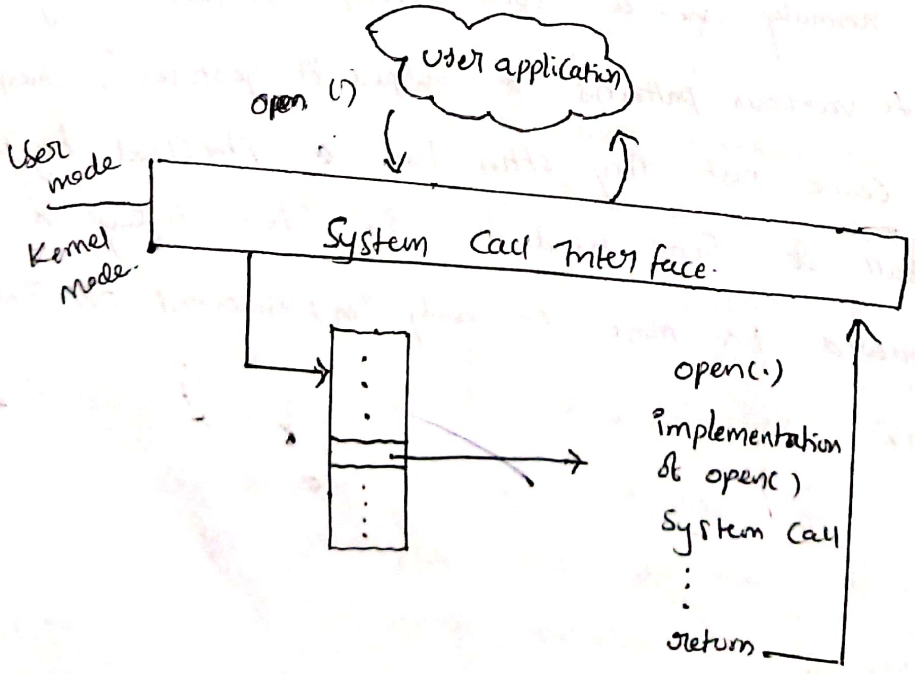
System calls provide a means for user or application programs to call upon the services of the operating system

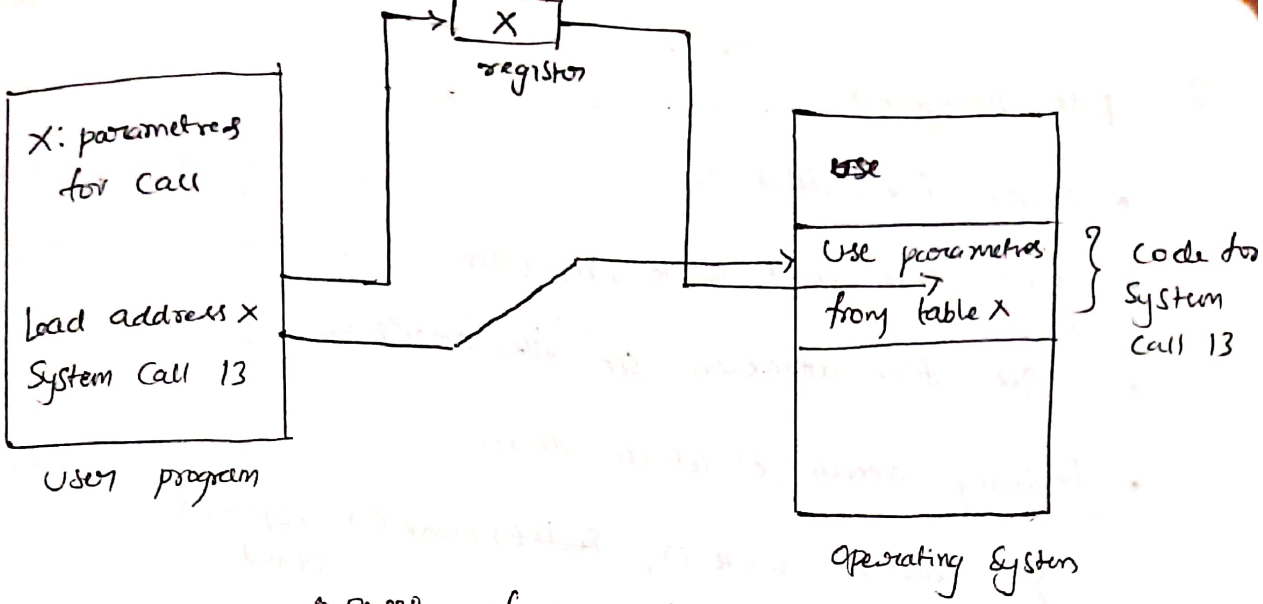
System calls are written in C or C++, although some written in assembly for optimal performance.

most programmers do not the low-level system calls directly, but instead use an "Application programming interface", API the diagram shows the read() call avaible in the API on UNIX based systems.

The use of APIs instead of direct system calls provides for greater programe portability between different systems.

The API then makes the appropriate system calls through the system call interface, using a table lookup to access specific numbered system calls.





—: Passing of parameters as a table :—

Types of System Calls :—

System Calls are categorized into 6 types They are.

1. process Control
2. File management
3. Device management
4. Information Management
5. Communication
6. protection.

1. Process Control :—

- end, abort, load, executive
- Create process, terminate process
- get process attributes, Set process attributes
- wait for time, wait event, Signal event
- allocate and free memory

Eg: fork( ), wait( ), exec( ).

2. File management :=

- Create file, delete file.
- open, close, read, write, reposition
- get files attributes, set file attributes
- logically attach & detach devices

Eg: read(), write(); ~~ioctl~~ close(), open()

3. Device management:

- request device, release device.
- read, write, reposition
- get devices attributes, set devices attributes
- logically attach & deatch devices

Eg: read(), write() ioctl()

4. Information maintance:

- get time & date, set time & date.
- get System data, Set System data.
- get process, file, & devices attributes
- Set process, file, & device attributes.

Eg: getpid(), alarm(), sleep()

5. Communications:

- Create, delete. Communication connection
- Send, receive message.
- Transfer status information
- attach & detach remote devices

Eg: pipe(), shmpget(), msgsnd(), msgrcv()

6. protection

- initialize. File. Security
- set file. security

Eg: chmod(), chown(), umask()

PART-B

1. Process Concept, process State diagram, process Control block

Process Concept :-

A process is an instance of programme in execution.

Batch Systems work in terms of "Jobs". Many modern process Concept are still expressed in terms of Jobs, (eg Job Scheduling), and the two terms are often used interchangeably.

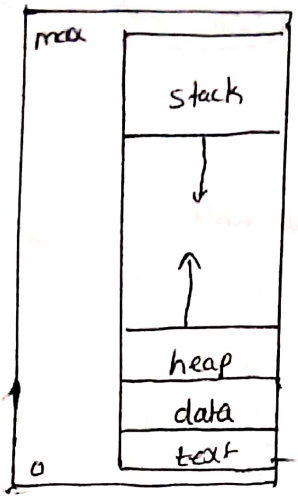
process memory is divided into four Sections (Text, Data, Heap, Stack)

The text Sections (stores (global) comprises the Compiled programe code, read in from Non-volatile Storage when the programe is launched.

The data Section Stores global and Variables, allocated and intialized prior to executing main

The heap is used for dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.

The Stack is used for local variables, Space on the stack is reserved for local Variables when they are declared (at function entrance & elsewhere, depending on the language), and the space is freed up when the Variables go out of Scope.

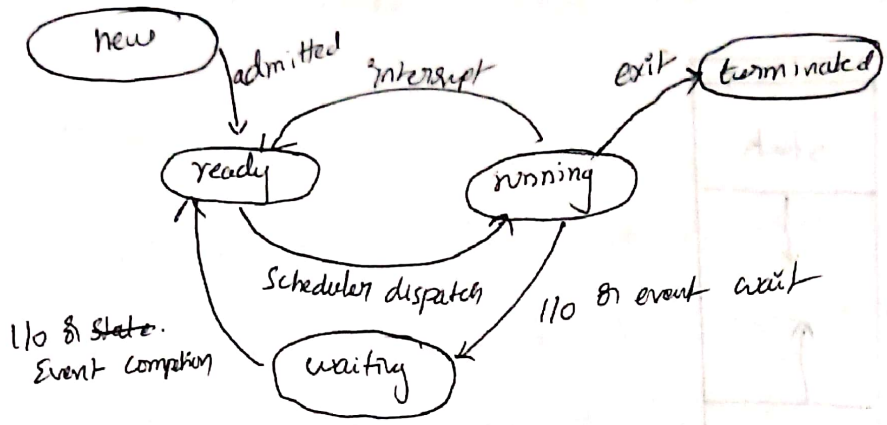


A process in memory

* process State Diagram:

process may be One of The 5 States, They are New, Ready, Running, Waiting, terminated.

1. New: The process is being created.
2. Ready: The process is (being Executed) waiting to be assigned. toa proc son.
3. Running: The process is being Executed.
4. Waiting: The process is waiting for some event to occur
5. Terminated: The process has finished. execution.



process state diagram

Explanation of process states:-

a) New > Ready : OS creates process and prepares the process to be executed; Then OS moves the process into ready queue.

b) Ready > Running : OS selects one of the Jobs from ready Queue and move them from Ready queue to Running.

c. Running > terminated : When the execution of a process has Completed, OS terminates that process from running State. Sometimes OS may terminates the process for Time Exceeded, memory unavailble, access violation, protection errors, I/O failure and soon

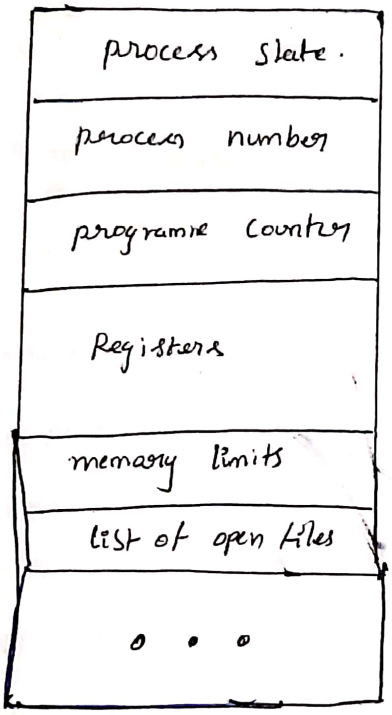
d. Running > Ready : When the time slot of the processor expired

e. Running -> waiting : If the process need an event or an I/O request

f. Waiting > ready : A process in the waiting state is moved to ready state when the event for which it has been completed.

process control block (PCB):-

For each process there is a process control Block, PCB, which stores the following information



process control block. (PCB)

1. Process State: The ~~process~~ state may be new, ready, running, and waiting, Terminated.
2. Process Number: process ID, and parent process ID.
3. program counter: indicates the address of the next instruction to be executed.
4. CPU registers: registers include accumulators, stack pointers, General purpose. Registers.
5. Memory limits: include page table, Segmentation tables, value of base. and limit registers.
6. CPU - Scheduling info:- includes a process priority, pointers to scheduling Queues. etc. Such as priority information

7. Accounting Information : includes amount of CPU used, time limits, Jobs (or) process numbers

8. I/O Status Information : Includes The list of I/O Devices Allocated to The processes, list of open files.

— o —

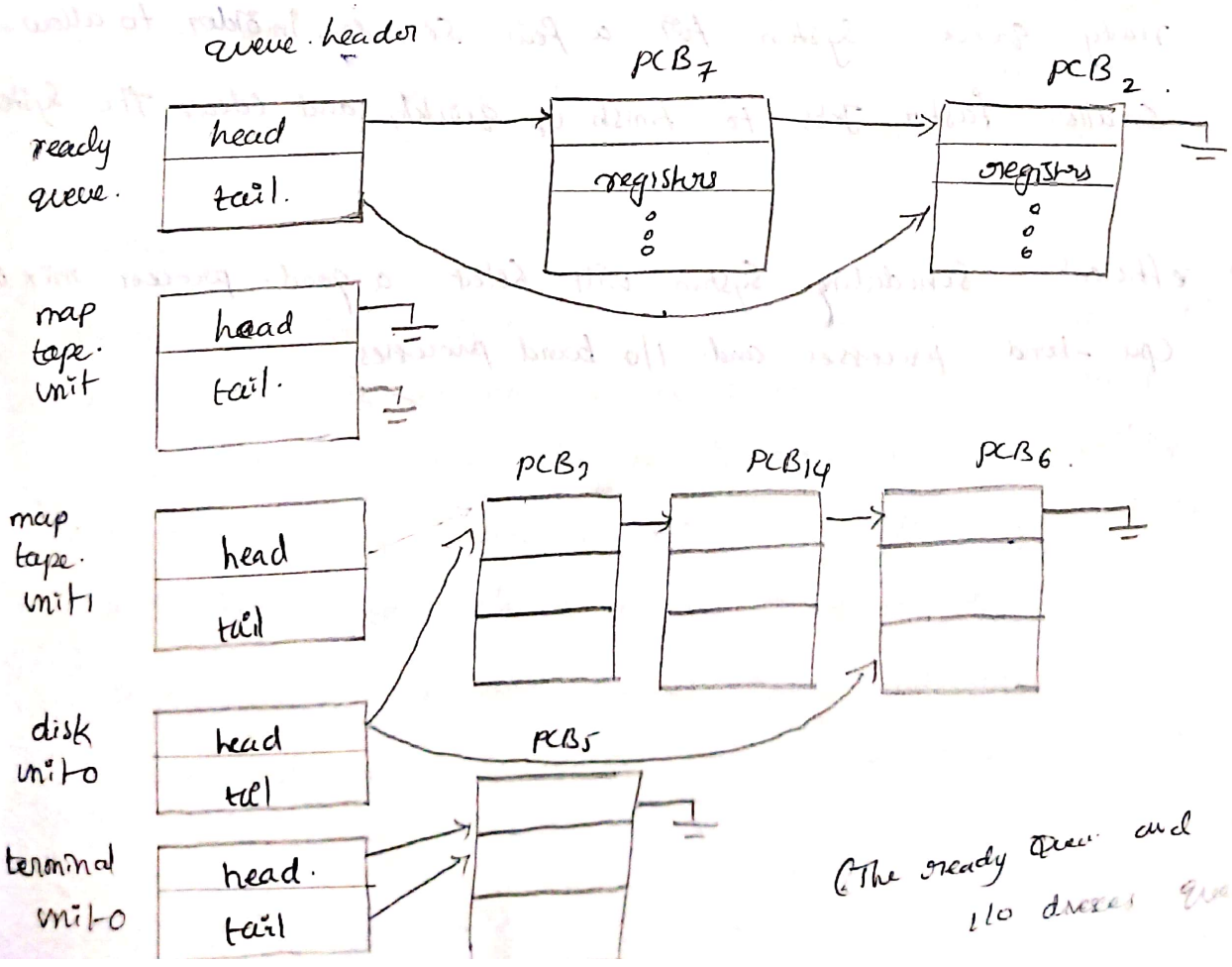
2. Process Scheduling

The two main objectives of the process scheduling system are to keep the CPU busy at all times and to deliver "acceptable" response times for all programs, particularly for interactive ones.

The process scheduler must implement suitable policies for swapping processes in and out of the CPU.

Scheduling Queues:-

- All processors are stored in the Job queue.
- processes in the Ready state, are placed in the Ready state, are placed in the ready queue.
- processes waiting for a device to become available & to deliver data, are placed in device queues. There is generally a separate device queues. (~~That is~~ (generally) for each device.
- other queues may also be created and used as needed.



(The ready queue and various I/O device queues)

Schedulers:

Schedulers are of 3 types, They are.

Long-term,

Short term,

Medium-term

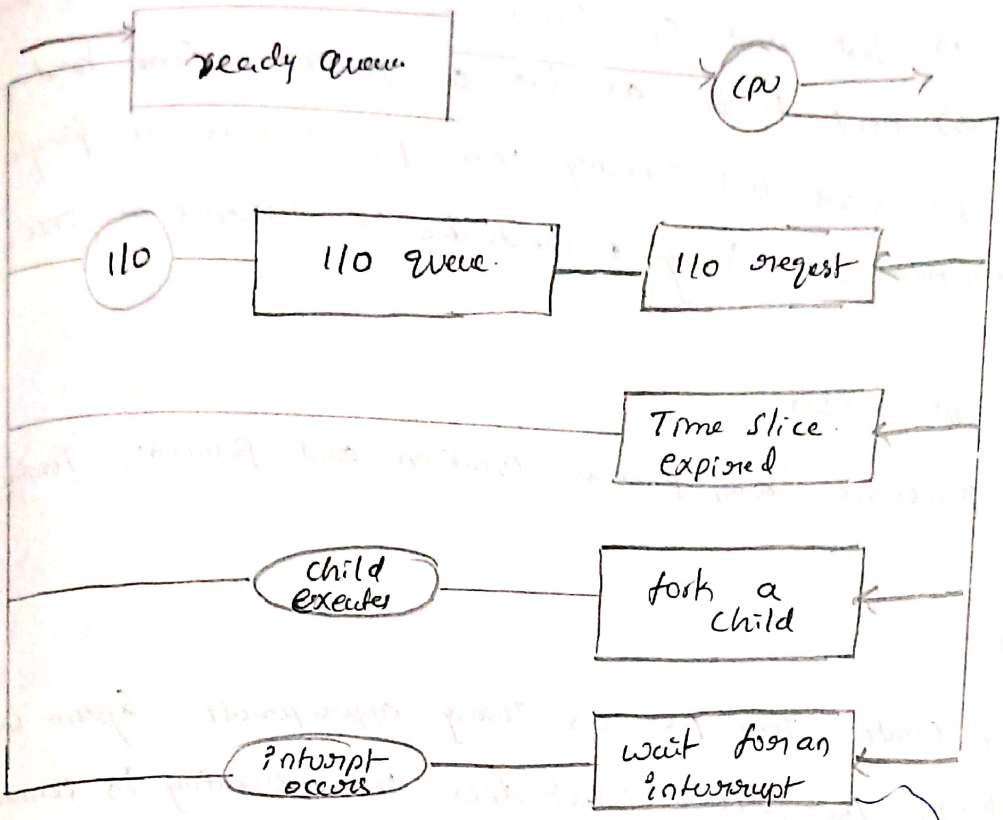
- A long-term Scheduler: This a type of Batch System & a very heavily loaded System. It runs infrequently

- The Short term Scheduler & Cpu Scheduler: It runs very frequently on the order of 100 milliseconds, and must very quickly Swap one process out of the cpu and Swaps in another one.

- Medium-term Scheduler.: when System loads get high. This Scheduler will Swap one & more processes out of the ready queue System for a few seconds, in Order to allow Smaller faster Jobs to finish up quickly and clear the System

An efficient Scheduling System will Select a good process mix of cpu-bond processes and I/o bound processes.

diagram



Queueing - diagram of processing scheduling.

Context Switch :-

Whenever an interrupt arrives, the CPU must do a State - Save of the currently running process, then Switch into kernel mode to handle the interrupt, and then do a state restore of the interrupted process.

Similarly a Context Switch occurs when the time slice for one process has expired and a new process is to be loaded from the ready queue. This will be initiated by a timer interrupt, which will then because the current process's state to be saved and the new process's state to be restored.

Saving and restore states involves saving and restoring all of the registers and program counter(s), as well as the process control blocks.

Context Switching happens very frequently, and the overhead of doing the Switching is just dost CPU time; so context Switches (State saves, & restores) need to be as fast as possible. Some hardware has special provisions for speeding this up, such as a single machine instruction for saving & restoring all registers at once.

operations on processes.

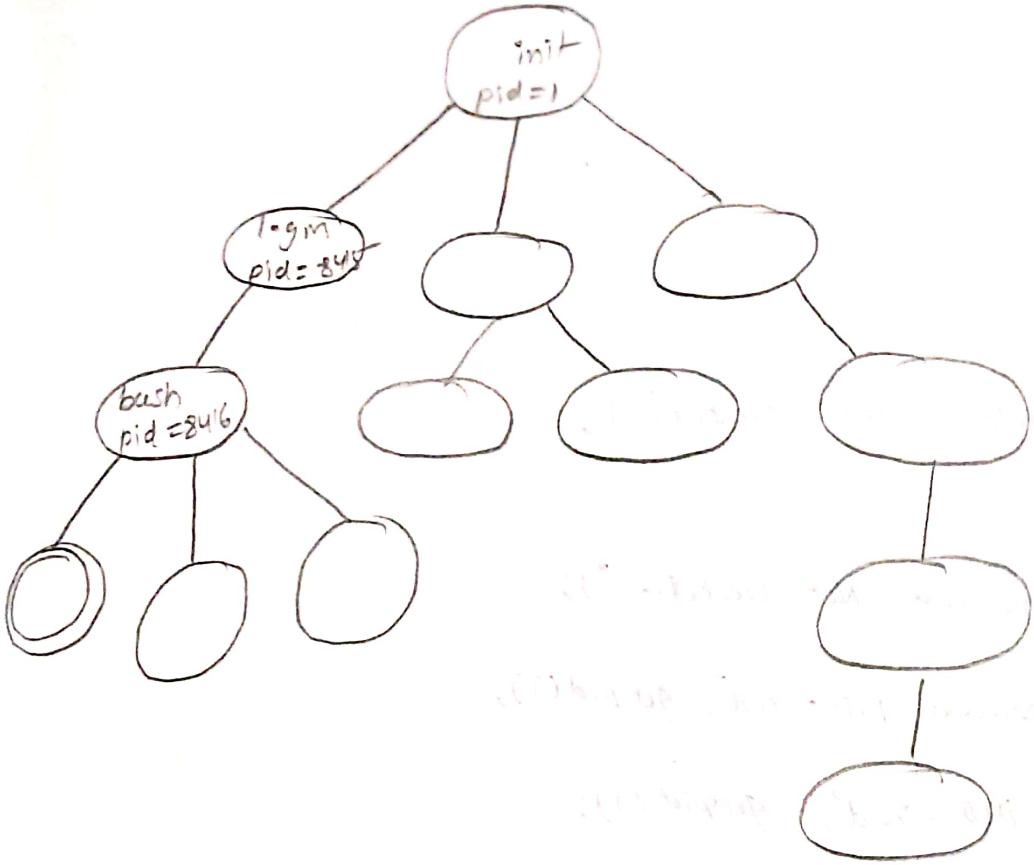
Operations on processes are, process creation and process Termination

process creation!

processes may create other processes Throug appropriate System calls, such as fork(). The process which does the creating is termed the parent of the other process, which is termed its child.

Each process is given an integer identifier, termed its process identifier, & PID. The parent PID (PPID) is also stored for each process.

On UNIX Systems The process Scheduler is termed Sched, and is given PID 0. The first Thing it does at System startup time is to Lanuch init, which gives That process PID 1. Init then Launches all System daemons and User login, and becomes the Ultimate parent of all other processes.



The fork System Call returns The PID of The processes child. to each process it returns a zero to The child process and a non-zero child PID to The parent, so The return value indicates. which process is which. process ID's can be looked up anytime. for the current process or its direct parent using The getpid() and. getpid() System calls respectively.

Eg: process creation using The fork() System Call.

```
# include <stdio.h>
# include <stdlib.h>
# include <sys/types.h>
```

```
main()
{
int pid;
pid = fork();
if (pid == 0)
        printf("child process created..");
else
    printf ("child process. Not created...");
    printf ("child process PID =%d", getpid());
    printf ("parent PID =%d", getppid());
    }.
```

process Termination:

processes may request Their own termination by making exit() system call, typically returning an int. This int is passed along to the parent if it is doing a wait(), and is typically zero on successful completion and some non zero code in the event of problems.

```
Chid Code:
        int exit code;
        exit (exit code);
parent Code;
        pid_t pid;
        int status
        pid = wait(& status);
```

5. Inter process Communication.

processes That execute Concurrently in the operating System may be either independent processes or cooperating processes. If a process cannot affect or be affected by The others processes executing in The System. Then The processes is said to be independent.

A process is said to be cooperating if it can affect or be affected by The other process exeuting in The System. which leads to inter process Communication. (IPC).

The reasons for IPC are,

Information Sharing :- Since several users may be interested in the same peice of information (for instance, a shared file), we must provide an envorinment to allow concurrent access to such information.

Computation Speedup :- If we wanlt particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with the others.

Modularity :- we can construct The System in a modular fashion dividing The System functions into Separate processes or Threads.

Convenience :- Even an indidual user can work on many tasks at The same time. like a user can be editing, listening to Music, and Compiling in parallel..

Cooperating processes need an inter-process communication (IPC) mechanism That will allow Them to exchange data and information.

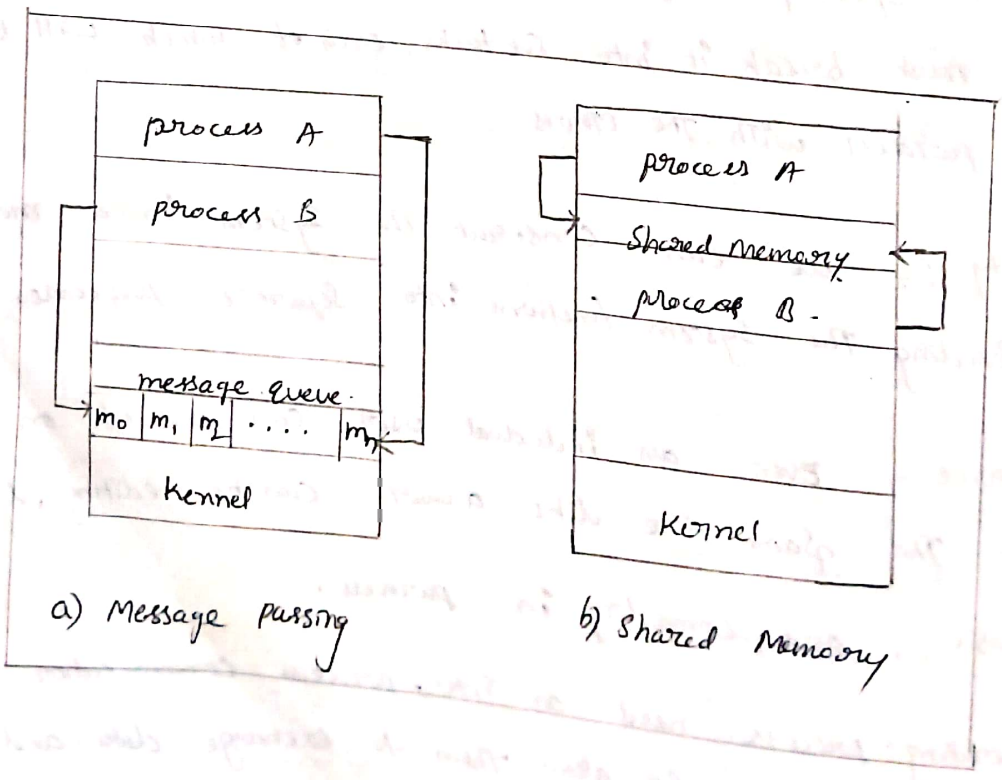
Two basic models of inter-process Communication are Messaging-passing and Shared memory

Message passing :- Message passing provides a mechanism to allow processes to communicate and to synchronize their actions without sharing the same address space and is particularly useful in a distributed environment, where the communicating processes may reside on different computers connected by a network.

Shared memory :-

A shared-memory region resides in the (network) address space of the process creating the shared-memory segment. other processes that wish to communicate using this shared memory segment must attach it to their address space.

Convenience :-



a) Message passing b) Shared Memory

4. Threads - Threading issues

Threads:

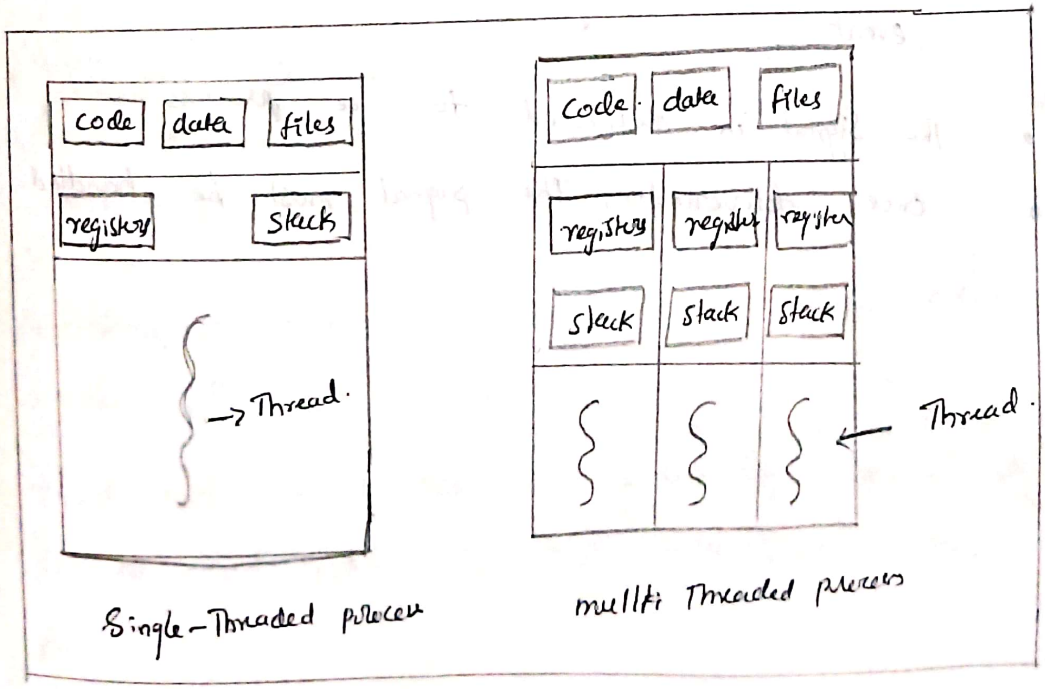
A process is divided into number of smaller tasks. each task is called a Thread.

A Thread is a basic unit of cpu utilization. A Thread consists of a Thread ID, a programe counter, a register set, and a stack.

It shares with other Threads belonging to the same process its code section, data section, and other operating system resources, such as open files and signals.

A traditional (or heavy weight) process has a single thread of control. if a process has multiple Threads of control, it can perform more than one task at a time..

Number of Threads with a process executive at a time is called Multi Threading. If a programe, in multi Threaded even when some portion of it is blocked, The whole program is not blocked. The rest of program continues working if multiple cpu's are avaible.



Threads issues:

The fork() and exec() System Calls:

The fork() is used to create a duplicate process. The meaning of the fork() and exec() System calls change in a multi threaded program

If One Thread in a program which calls fork(), does the new process duplicate all threads.

If a Thread calls the exec() System call, the program specified in the parameter to exec() will replace the entire process which includes all Threads.

Signal Handling:

Signal is used in Linux Systems to notify a process that a particular event has occured.

All Signals, whether Synchronous & asynchronous, follow the same pattern as given below.

- A Signal is generated by the occurance of particular event
- The Signal is delivered to a process
- Once delivered, the Signal must be handled.

Cancellation:-

Thread Cancellation is the task of terminating a thread before it has completed.

A target thread is a thread is to be cancelled, Cancellation of target thread may occur in two different scenarios,

Asynchronous Cancellation:- One thread immediatly terminates the target thread.

Deffered Cancellation: The target thread periodically checks whether it should terminate, allowing it an opportunity to terminate itself in an ordinary fashion

Thread pools:-

multiThreading in a web server, whenever the server receives a request it creates a separate thread to services the request.

Some of probelms that arise in creating a thread are,

- The amount of time required to create the thread prior to serving the request together with the fact that this thread will be discarded once it has completed its work.
- If all concurrent requests are allowed to be serviced in new thread, thre is no bound on the number of threads concurrently active in the system.
- Ultimated thread could exhaust system resources like CPU time or memory

A thread pool is to create a number of threads at process start-up and place them into a pool, where they sit and wait for work.

5. Scheduling (Basic concepts, Scheduling criteria, Scheduling Algorithms).

Basic Concepts :-

Almost all programs have some alternating cycle of CPU Number crunching and waiting for I/O of some kind. (Even a simple fetch from memory takes a long time. relative to CPU speeds.

In a simple system running a single process, the time spent waiting for I/O is wasted, and those CPU cycles are lost forever.

A scheduling system allows one process to use the CPU another is waiting for I/O, there by making full use of otherwise lost CPU cycles.

CPU – I/O Burst cycles

Almost all processes alternate between states in Counting cycles

- A CPU burst of performing calculations, and
- An I/O burst, waiting for data transfer in or out of the system

CPU bursts vary from process to process, and from program to program.

CPU Scheduler :- Whenever the CPU becomes idle, it is the job of the CPU scheduler (short-term scheduler) to select another process from the ready queue to run next

storage structure for the ready queue and the algorithm used to select the next process are not necessarily a FIFO queue. There are several alternatives to choose from, as well numerous adjustable parameters for each algorithm.

preemptive Scheduling :-

- CPU Scheduling ~~for~~ decisions take place under one of four conditions:

1. when a process switches from the running state to the waiting state, such as for an I/O requests or invocation of the wait() system call.
2. when a process switches from the running state to the ready state, for example in response to an interrupt
3. when a process switches from the waiting state to the ready state, say at completion of I/O or a return ~~to an~~ from wait().
4. when a process terminates.

- For conditions 1 and 4 there is no choice - A new process must be selected.
- For conditions 2 and 3 there is a choice - To either continue running the current process, or select a different one.
- If scheduling takes place only under conditions 1 and 4, the system said to be non-preemptive, or cooperative. Under these conditions, once a process starts running it keeps running until it either voluntarily blocks or until it finishes otherwise. The system is said to be ~~prem~~ preemptive.

Dispatcher :-

- The dispatcher is the module that gives control of the CPU to the process selected by scheduler. This function involves

  - Switching context
  - Switching to user mode.
  - Jumping to the proper location in the newly loaded program.

- The dispatcher needs to be as fast possible, as it is run on every context switch. The time consumed by the dispatcher is known as dispatch latency.

Scheduling criteria :-

Different CPU-scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another.

In choosing which algorithm to use in particular situation, we must consider the properties of the various algorithms.

Many creatierica have been suggested for comparing CPU-scheduling algorithms,

They are :

- CPU-Utilization
- Throughput
- Turn around time
- waiting Time and
- Response Time.

CPU utilization:- we want to keep the CPU as possible.

Conceptually.

CPU utilization can range from 0 to 100 percent. ie the percentage of CPU being utilized.

Throughput:- If the CPU is busy executing processes, then work is being done. One measure of work is the Number of processes / Jobs that are completed per unit, called Throughput. ie: Number of Jobs completed by the CPU with in a time period.

Turn around time: The interval from the time of submission of a process to the time completion is turn around time. Turn around time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

Waiting Time: waiting time is the sum of the periods spent waiting in the ready queue.

waiting time = starting time - arrival time.

Response time:- It is time from the submission of a request until the first response is produced.

Burst Time / Execution Time: it is time required by the process to complete execution. It is also called running time.

Arrival Time: when a process enter in a ready state.

Finish Time: when process is complete and exit from a system.

Scheduling Algorithms :

The various Scheduling Algorithms are.

1. FCFS
2. SCJF
3. priority.
4. RR,
5. multilevel Queue Scheduling.
6. multilevel Feedback - Queue Scheduling.

1. FCFS :-

FCFS stands for First come First Serve.

It is the easiest and most simple cpu Scheduling algorithm.

Here, The process which requests the cpu gets the cpu allocation first.

As the process enters the ready queue, its PCB (process Control Block) is linked with the tail of the queue. So when cpu becomes free, it should be assigned to the process at the beginning of the Queue.

It offers non-preemptive and pre-emptive Scheduling algorithm this method is poor in performance, and the general wait time is quite high.

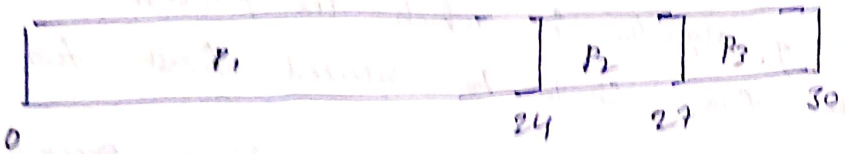
Ex:. Consider 3 processes with Burst time Mentioned Below,

Process	Burst time
P_1	24
P_2	3
P_3	3

The process enters the ready queue, its PCB (process control block) is linked with the tail of the queue. So, when CPU becomes free, it should be assigned to the process at the beginning of queue.

It offers non-preemptive and pre-emptive scheduling if no waste.

If the processes arrive in the order P1, P2, P3, and are served in FCFS order, then the Gantt chart is;



Waiting time is the sum of periods spent waiting in the Ready Queue.

The waiting time is 0 milliseconds for process P1,

24 milliseconds for process P2 and,

27 milliseconds for process P3.

Thus, the average waiting time is. $(0+24+27)/3 = 17$ milli-seconds.

Turn around time (TAT) is the total of waiting Time & Burst Time.

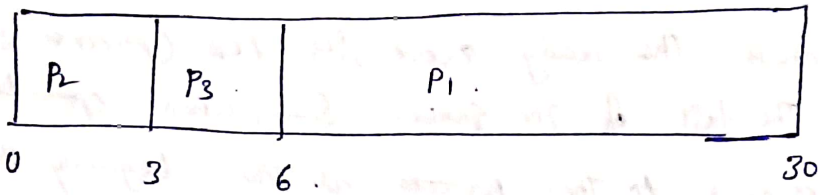
The TAT of P1 is $0+24 = 24$.

The TAT of P2 is. $24+3 = 27$

The TAT of P3 is. $27+3 = 30$

Therefore, Avg TAT is. $(24+27+30)/3 = 26$ Milliseconds.

If The processes arrives in The order P2, P3, P1, . Then The Gantt chart is



The average waiting time is now (6+0+3)/3 = 3 milliseconds.

2. SJF:

SJF stands for Shortest Job first

it is Scheduling algorithm in which The process with The Shortest execution time Should be Selected first for execution.

This Scheduling method can be preemptive or non-preemptive and is implemented in non-preemptive method.

it Significantly reduces The average waiting time for other processes awaiting execution.

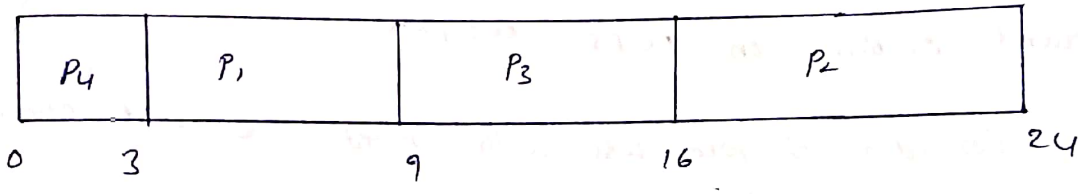
it is useful for batch-type processing, where waiting for Jobs to Complete is not critical

it improves Job output by offering Shorter Jobs, which b should be executed first, which mostly have a shorter turnaround time.

eg: Consider 4 processes with burst time mentioned below,

process	burst time
P_1	6
P_2	8
P_3	7
P_4	3.

Then The Grantt chart is,



Waiting time is The Sum of periods spent waiting in The Ready queue.

The waiting time is 3 milliseconds for process P1,
16 milliseconds for process P2,
9 milliseconds for process P3 and
0 milli seconds for process P4.

Thus, The average waiting time is (3+16+9+0) /4 = 7 milliseconds

Turn around time is The total of waiting time & Burst time.

The TAT of P1 is 3+6=9
The TAT of P2 is 16+8 = 24
The TAT of P3 is 9+7 = 16
The TAT of P4 is 0+3 = 3.

Therefore, Avg TAT is (9+24+16+3) /4 = 13 milliseconds.

3. priority Scheduling :-

priority Scheduling is a method of Scheduling processes based on priority

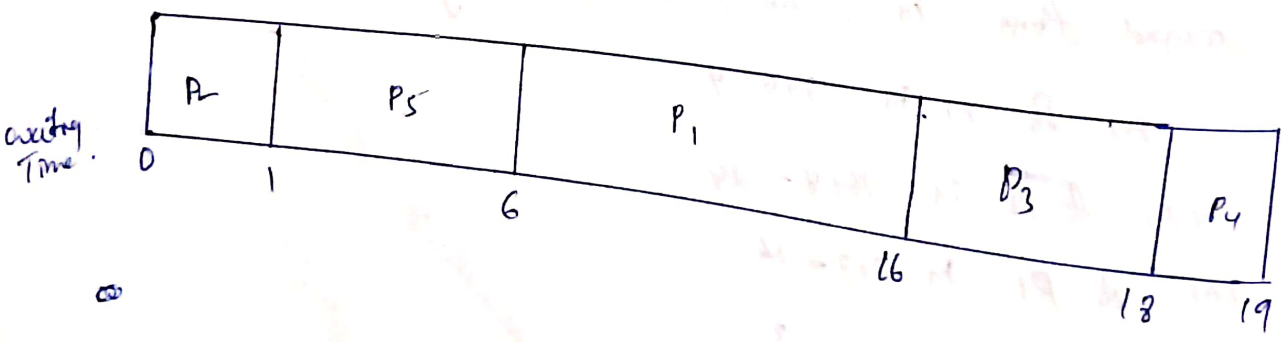
In This Method, The Scheduler Selects The Task to work as per the priority

priority Scheduling also helping OS to involve priority assignments The processes with higher priority should be carried out first, whereas Jobs with equal priorities are carried out on a round robin or FCFS basis.

Eg: Consider 5 processes with burst time and priority mentioned below,

process	Burst time	priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Then, The Gantt chart is



waiting in the Ready Queue.

The waiting time is 6 Milliseconds for process P1,
0 Milliseconds for process P2,
16 Milliseconds for process P3,
18 Milliseconds for process P4. and
1 Milliseconds for process P5

Thus, the average waiting time is (6+0+16+1)/5 = 8.2 Milliseconds.

Turn around time is the Total of waiting Time & Burst time.

The TAT of P1 is. 6+10 = 16
The TAT of P2 is 0+1 = 1.
The TAT of P3 is. 16+2 = 18
The TAT of P4 is 18+1 = 19
The TAT of P5 is 1+5 = 6.

Therefore, Avg TAT is (16+1+18+19+6)/5 = 12 Milliseconds.

4. Round Robin Scheduling:

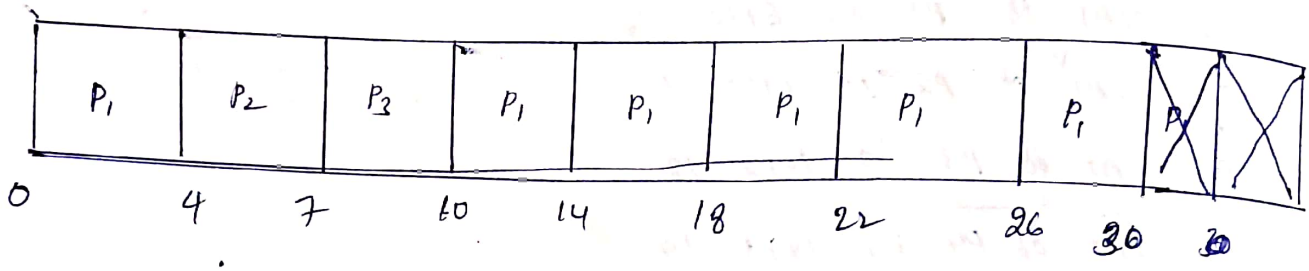
Round Robin is the oldest, Simplest Scheduling algorithm
The name of this algorithm comes from the around-Robin princip
where each process gets an equal share of time.
It is mostly used for Scheduling algorithm in multitasking
This algorithm method helps for Starvation free execution of processes.

It is a real time System which responds to the event within a Specific time limit

Eg1:- Consider 3 processes with burst time mentioned below,

process	Burst time
P_1	24
P_2	3
P_3	3
P_4	2

If we use a Time Quantum of 4 milliseconds, Then The Gantt Chart is.,



waiting time is The Sum of periods spent waiting in The Ready Queue.

The waiting time for p1 is 6milliseconds (10-4) = 6.

P2 is 4 milliseconds and

P3 is 7 milliseconds

Thus., The average waiting time is. 17/3 = 5.66. Milliseconds

Turn around time is The. The Total of waiting Time & Burst time

The TAT of P1 is. 6+3 = 9

The TAT of P2 is 9+6 = 15

The TAT of P3 is. 9+4 = 13

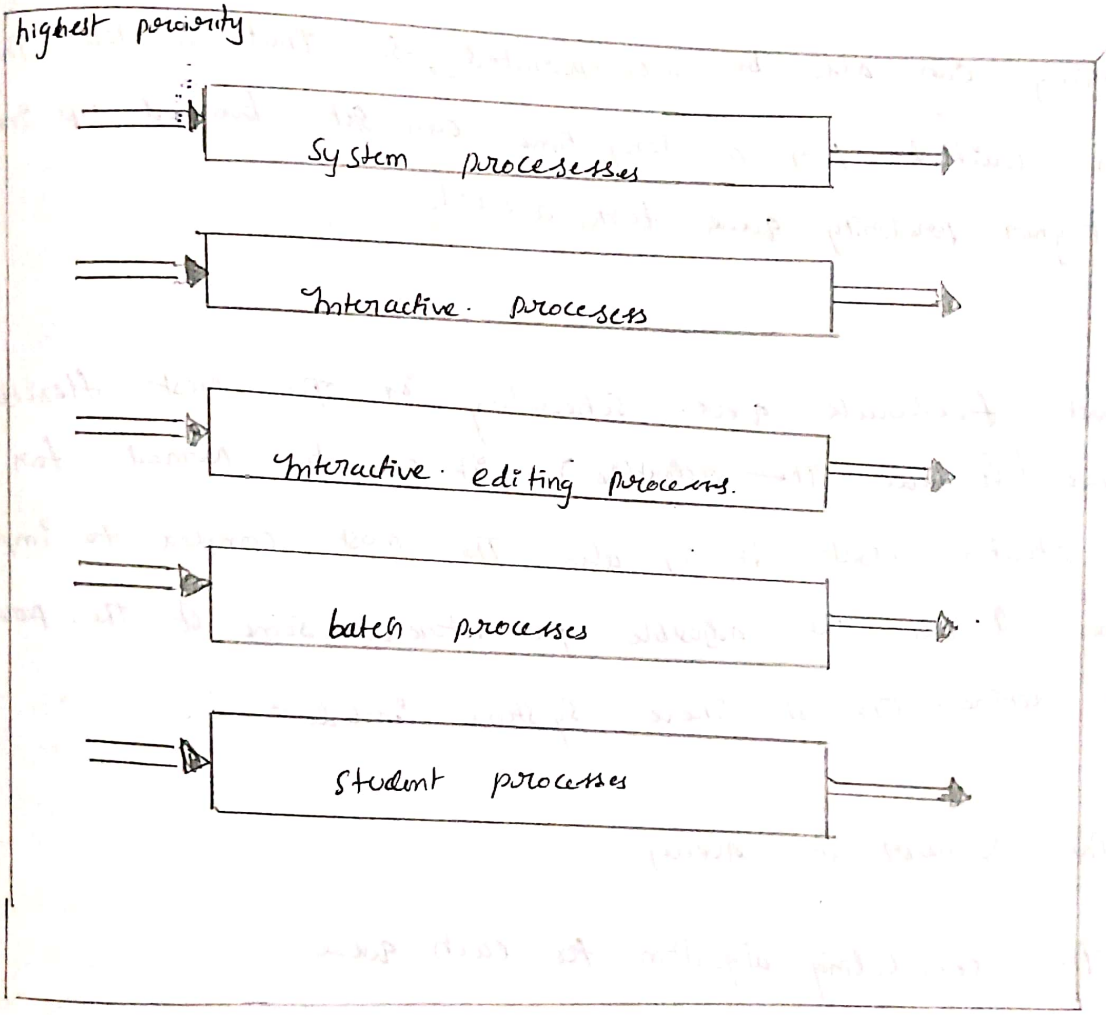
The TAT of P4 is. 6+2 = 8.

Therefore, Avg TAT is $\frac{(9+15+13+8)}{4}$ = 11.25 milliseconds.

Multilevel [illegible] scheduling :-

when processess can be readily categorized, Then multiple separate queues can be established, each implementing whatever scheduling algorithm is most appropriate for that type of Job, and /8 with different parametric adjustments.

Scheduling must also be done between queues, That is scheduling one queue to get time relative to other queues.

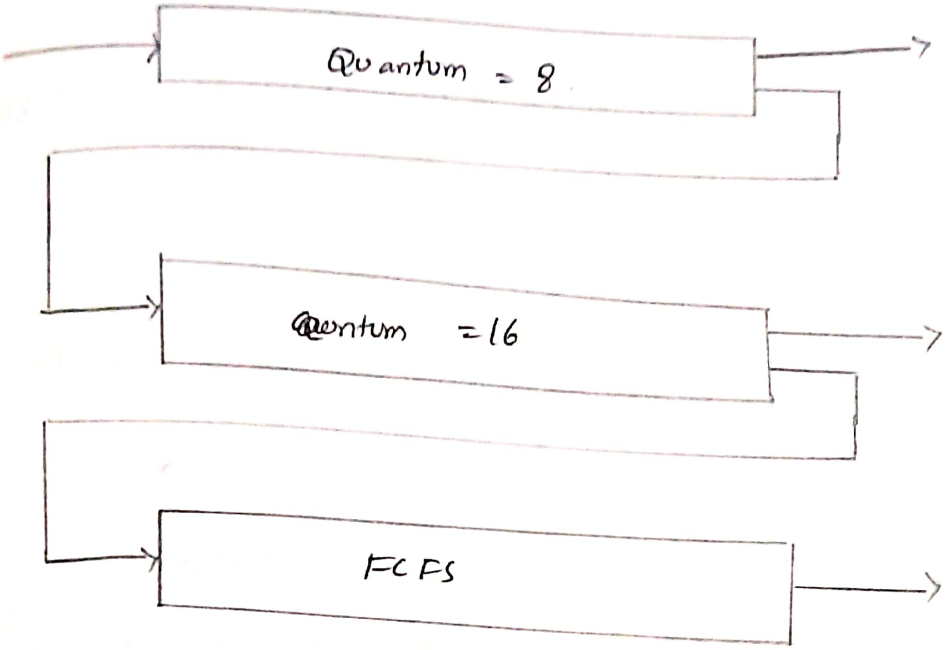


Multi Level queue scheduling.

6. Multilevel Feedback – Queue Scheduling

- Multilevel feedback queue scheduling is similar to the ordinary multilevel queue scheduling, except jobs may be moved from one queue to another for a variety of reasons.
  - If the characteristics of a job change between CPU-intensive and I/O intensive, then it may be appropriate to switch a job from one queue to another.
  - Aging can also be incorporated, so that a job that has waited for a long time can get bumped up into a higher priority queue for a while.

- Multilevel feedback queue scheduling is the most flexible, because (of all these adjustable) it can be tuned for any situation, But it is also the most complex to implement because of all the adjustable parameters. Some of the parameters which define one of these systems include.
  - The number of queues.
  - The scheduling algorithm for each queue.
  - The methods used to upgrade or demote processes from one queue to another.
  - The method used to determine which queue a process enters initially.



Multi Level Feedback queues.