2 Marks:

1) What is operating system. List its goals.
An operating system is a program that act as interface betn the user and the computer hardware and controls the execution of all kinds of programs.
Main Goal of operating system.
1) Convenience — is of use.
2) Throughout — No. of task executed per unit time.

2) Explain any two functions of operating system.
i) Resource Management → Multiple resource like CPU, memory disk can have parallel access.
ii) Processor Management → As operating system manages the processor work by allocating various jobs to it and ensuring that each process recieve at the time from the processor to function properly.

3) Write the features of operating system.
(i) provides a platform for running application.
(ii) Provides file system abstraction.
(iii) Provides networking support.
(iv) Provides security features.
(v) Provides user interfaces.
(vi) Provides utilities and system services.
(vii) Supports application development.

3) Explain the two types of Real Time operating system.
(i) Hard Real time System:-
These systems have strict timing requirements and must be meet deadlines or an system can fail.
Eg:- Control systems for aircraft and medical devices.

② Soft Real time system :- These systems also have timing requirement but missing a deadline does not necessarily to result in huge failure.
Eg :- Multimedia applications & video games, online gaming.

⑤ Define the following.

ⓐ Process :- A process is a program in execution.
→ A process is more than the program code, which is sometimes known as text section.

ⓑ Thread :- A thread is unit of execution within the process. A thread is also called as light weight process.

ⓒ Process scheduling → It is an operating system that schedules process of different states like ready, waiting, running.

ⓓ Context switching → A context switch is a mechanism in a process control block stores and restores the state of context of a CPU so that a process execution can be resumed from the same point at a later time.

ⓔ IPC → It is a mechanism provided by the operating system that allows process to communicate with each other.

5 Marks.

11. Explain Distributed OS. Mention its advantages and disadvantages.

Distributed systems use multiple central processor to serve multiple real time applications and multiple user. Dataprocessing are distributed among the processors.

→ The processors communicate one another to various communication lines (such as, high speed buses or telephone lines. These are refered as lossy coupled system or distributed system.

→ processors in a distributed systems may vary in size and functions.

→ These processors are reffered as sites, notes, computer and so on.

Advantages.

→ Resource sharing → User at one side may be able to use resource available at another.

→ Faster data transfer.

→ Exchange of data happens quickly with one another via electronic mail.

→ Fault Tolarance → If one site fails, the remaining sites can continue operating.

→ There is less load On the single computer.

→ Data processing → There is reduction of deals in the processing.

Disadvantages

- Complex networking
- Security disk.

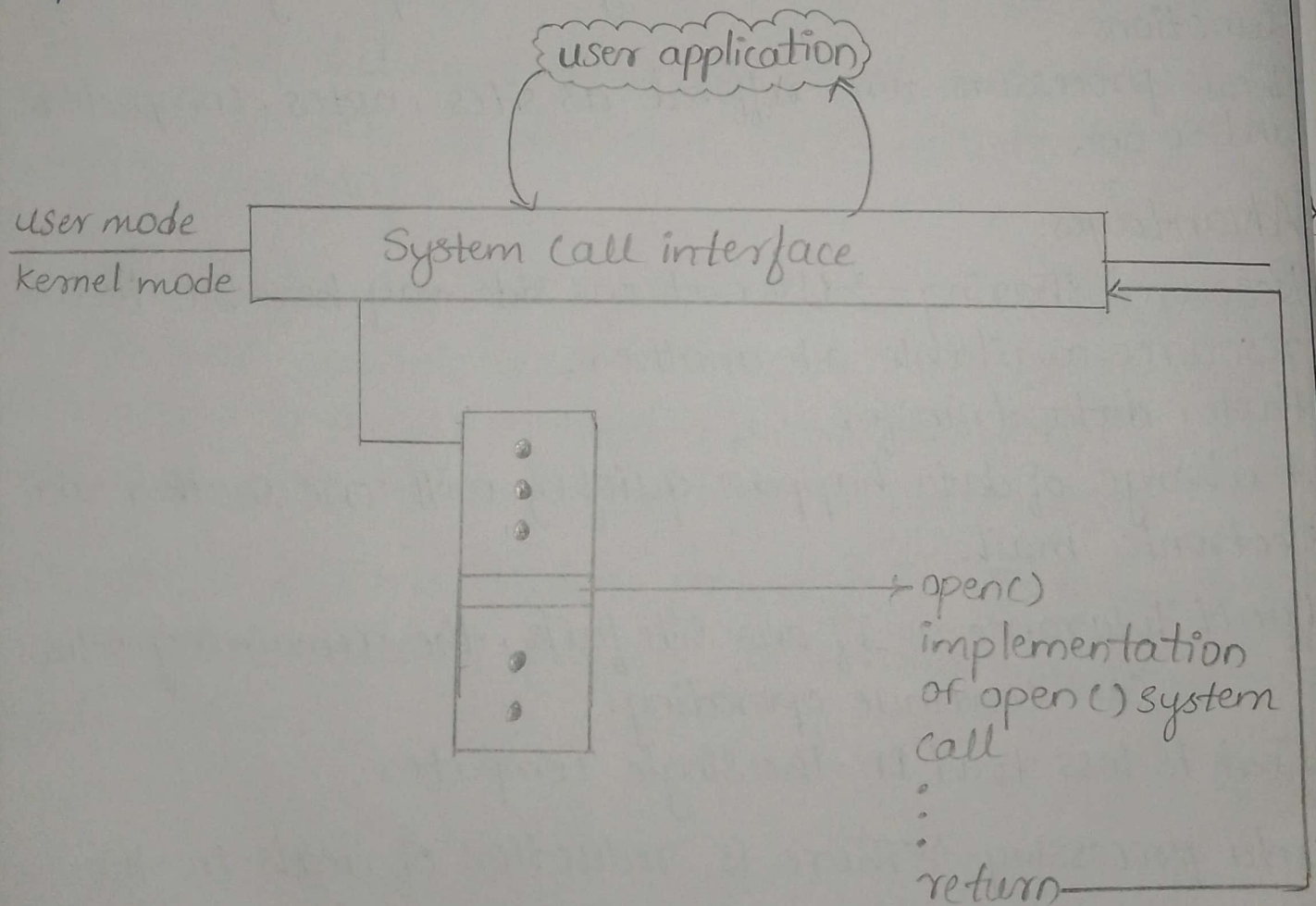
(2) Write a short note on system calls.

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

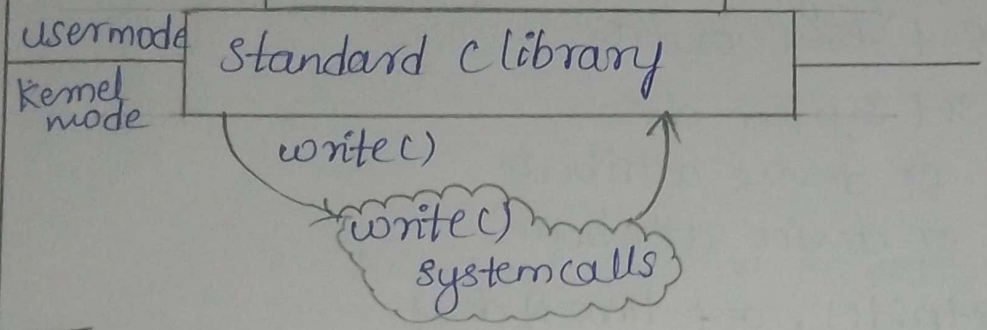
→ System call is a programatic way in which a computer program request a service from the kernel of a OS.

→ These calls are generally available as routine written in C & C++.

→ A computer program makes a system call when it makes a request ..



```
#include<stdio.h>
int main()
{
 **
 **
 *
 printf("Greeting");
 **
 **
 *
 return 0;
}
```



Types of System calls.

1. Process control:-
   These system calls manages processes.
2. Create process, terminate process load, execute.
3. Get process attribute, set process attribute.
4. Wait event, signal event.
5. Allocate and free memory.

System calls:- fork(), exec(), exit()

6. File management:-
   These system calls are responsible for file manipulation
7. Create file, delete file.
8. Open file, close file
9. Read, write, reposition.
10. Get file attribute, set file attributes.

3) Device Managment — Responsible for device manipulation.
1) Request device, Release device.
2) Read, write, reposition.
3) Get device attribute, set device attribute.
4) Logical attach or deattach device

System calls :- ioctl, write(), read()

4) Information maintaince — Handles information and transf-er between operating system and the user program.
1) Get time or date, Set time or date.
2) Get system data, set system data.
3) Get process, file or device attribute.
4) Set process, file or device attribute.

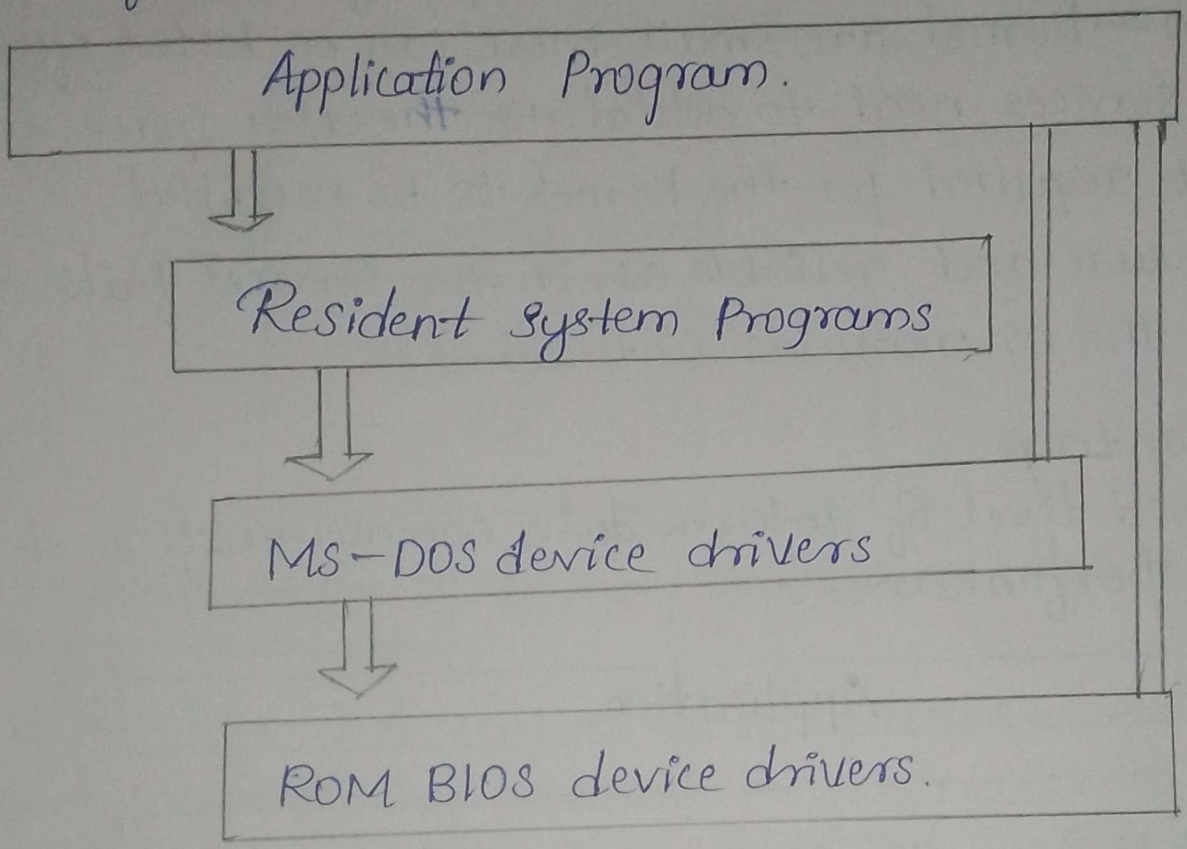
System calls :- getpid(), get time & day.

⑤ Communication :- Usefull of interprocessing for communica
1) Create delete, communication connection.
2) Send or recieve msgs.
3) Transfer status information.
4) Attach or deattach remote device.

System calls :- pipe(), Send(), Recv().

Explain Monolithic OS structure with a neat diagram.

Such OS do not have well defined structure & small, simple and limited system. The interfaces and levels of functionality are not well separated. MS DOS is an example of such OS. In MS DOC application programs are bulk able to access the basic IO routins. These types of OS cause entire system to crash if one of the user program fails.



Advantages.

It delivers better application performance because of betⁿ appⁿ program & hardware.

Easy for kernel developers to develop such an OS.

Disadvantages.

The structure is complicated as no clear boundaries exist betⁿ modules.

It does not unforce data hiding the OS.

④ Explain Microkernel OS structure with a neat diagram.

This structure designs operating systems by removing all non-essential components from the kernel and implementing as system and user programs. This results as a smaller kernel called the micro-kernel.

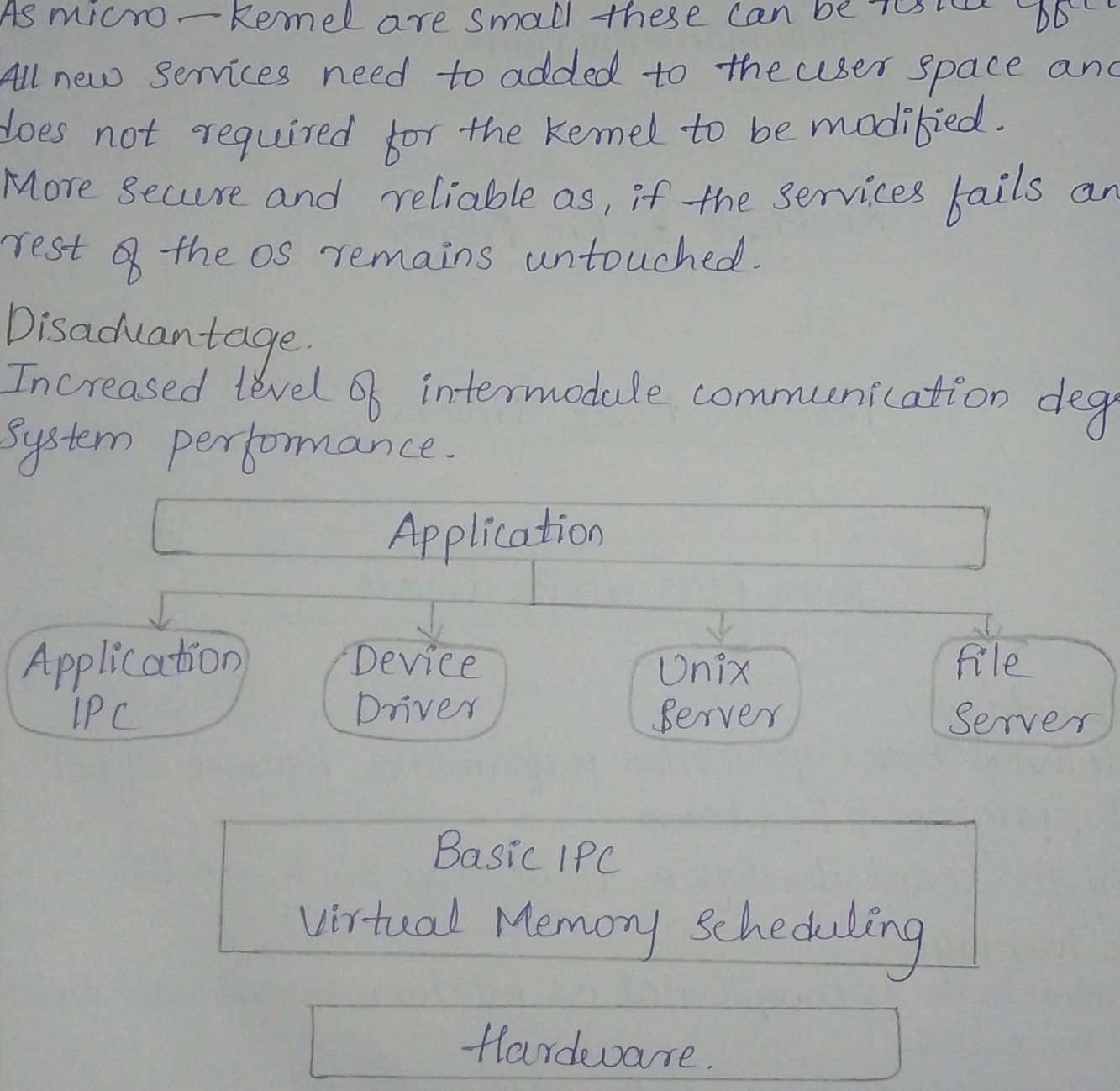
Ex:- MAC-OS.

Advantages.

→ It makes OS portable to various platforms.
→ As micro-kernel are small these can be tested effectively
→ All new services need to added to the user space and does not required for the kernel to be modified.
→ More secure and reliable as, if the services fails and rest of the OS remains untouched.

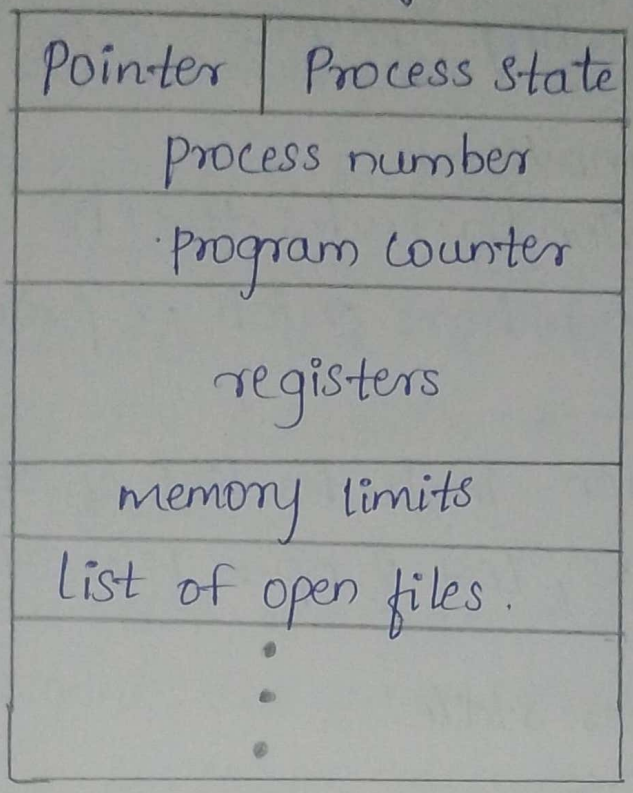
Disadvantage.

→ Increased level of intermodule communication degrad System performance.



Explain process control Block (PCB)

→ Each Process is represented in OS by the process control block (PCB also called as task control block) A PCB is as shown in the fig. below.



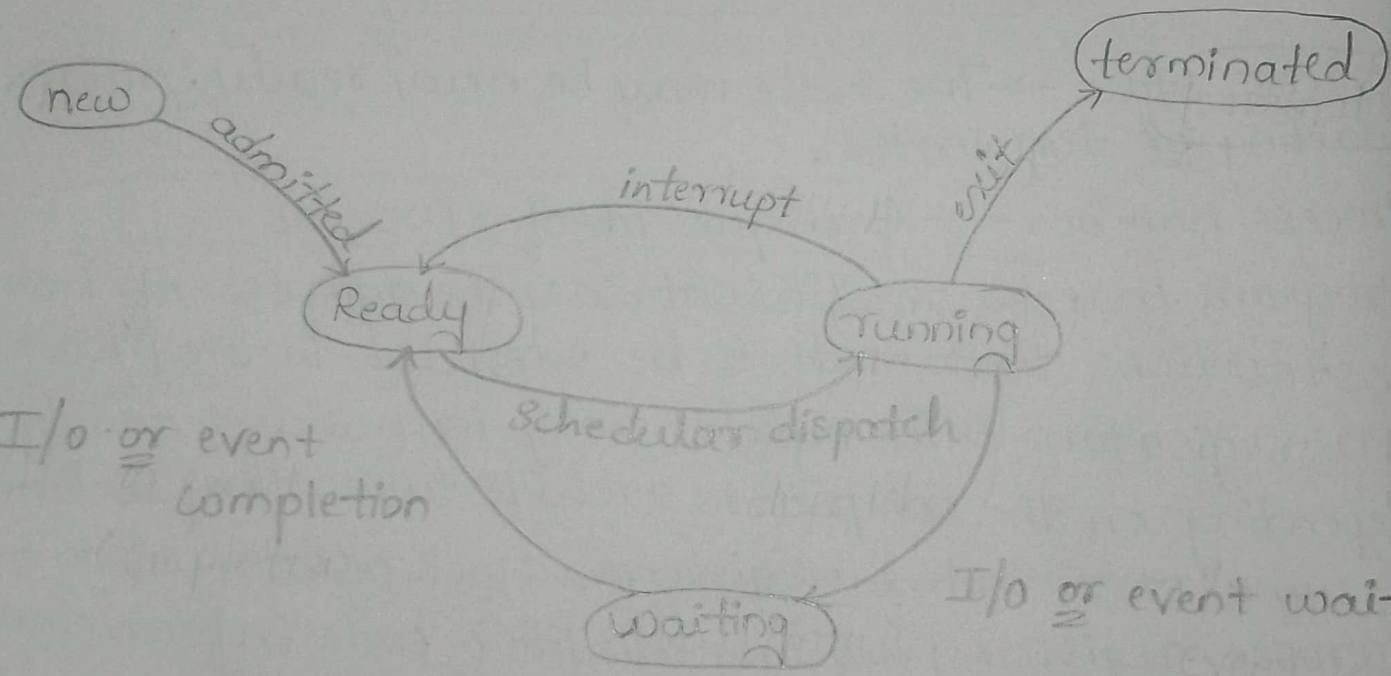
- Process state → The state may be new, ready, running, waiting or terminated.
- Process number → A unique id of a process.
- Program counter → It contains the address of next instructions that needs to be executed in the process.
- CPU regisators → The registor vary in number and type, depending on the computers architecture. They include accumalator, index registors, stack pointers & general purpose registor plus any condition code information.
- CPU Scheduling information :-
  This information includes process priority, pointers to scheduling queus & any other scheduling parameters.

Memory Mangment —
This information includes process priority may include value of the base and limit registor page tables or segment tables, depending on the memory system used by the operating system.

Accounting information —
This information includes the CPU used, time limits, accounts numbers or job or process numbers.

I/o information —
This information includes list of I/o devices used by the process, list of open files etc.

⑥ Explain the process state.



As a process executes it change state. The state of a process is defined in part by the current activity of that process.

New — The process is being created (this is a inital state when the process is first created.

Ready — The process is to be assign to a processor. the process is ready for execution. The process may come to a ready state after the new state or when the it is interpreted by the schedular to assign the CPU to some other process.

Running — Instruction are executed. If once the process as assigned, the process state is send to the running and the processor executes the instruction.

Waiting — The process is waiting for some event to occur (such as IO completion, repectition of the signal).

Termination — The process has finished execution. once the process finishes its execution or it terminated by the OS it is moved to the terminated place where it is waits to be removed from the main memory.

Explain the types of process scheduling queues with a neat diagram.

It is an OS task that schedules process of different states like ready, waiting, running.

Process Scheduling allows OS to allocates a time

interval of CPU execution for each process.

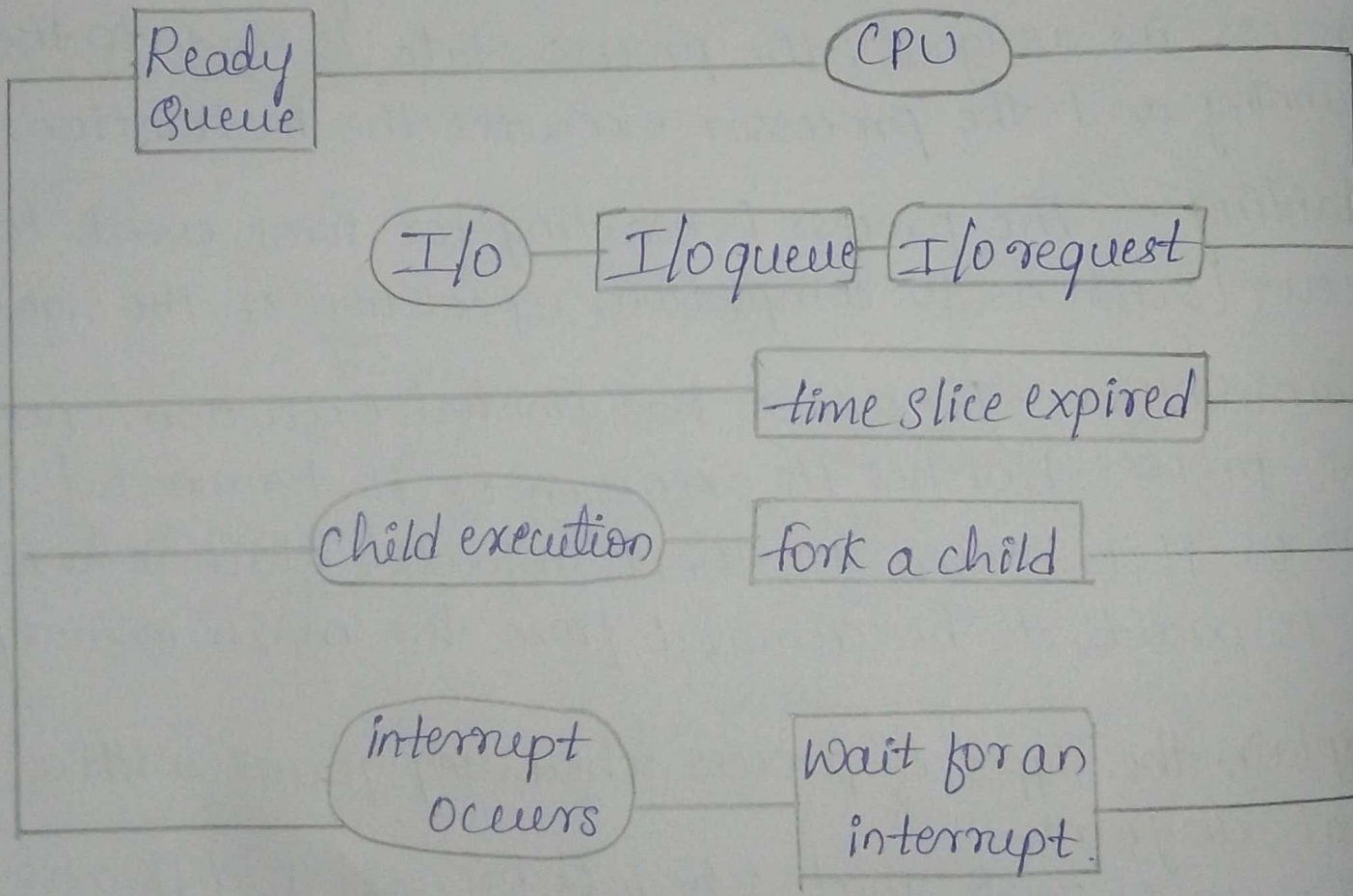
→ Process schedular selects among available process for next execution on CPU.

Process scheduling queues.

① Job Queue →
Whenever any process enters the system, it is there in the job queue.

② Ready Queue →
Processes in the ready queue are waiting for run time or execution.

③ Device Queue →
Some processes may be waiting for some I/O operation Such process are in the device queue.



The ready queue is where a new process is initally placed. It sits in the ready queue waiting to be choosen execution or dispatch.

one of following occurs can happen once the process has been assigned to the CPU and is running.

The process could send out an I/O request before we placed in the I/O device or queue.

The procedure could start one then wait for it for finish.

As a result of an interrupt, the process could be forcily removed from the CPU and returned to the ready queue.

The process finally moves from the waiting to the ready state in first two circumstance & then return to the ready queue.

This cycle is repeated until the process is terminated at which point is withdrawn from all queues and its BCP & resources are reallocated.

Explain the types of process schedulers.

A scheduler is a type system service that allow us handle process scheduling.

there primary responsibility is to choose which jobs to be submit into the system & which process to run.

Types of 3 process schedulers.

① long term scheduler.
② short term scheduler.
③ Medium term scheduler.

① Long term scheduler.

→ It is also known as job scheduler.

→ It selects the process that is to be placed in the ready queue.

→ A long term schedular determines which program can be accepted for processing into the system.

→ It picks process from the queues than loads into the queue so that they can be executed for CPU scheduling a process loads into the memory.

② Short term scheduler.

→ It is also known as CPU scheduler.

→ The main goal of the schedular is to boost the system performance according to set criteria. This helps to select from the group among the processes that are ready to execute and allocates CPU to one of them.

→ The dispatch gives control of CPU to the process selected by the short term schedular.

③ Medium term scheduler:

→ Medium term scheduling is an important part of swap. It enables us to handle the swap in and swap out process.

→ Swapping clears the memory of the processes.

→ The degree of multi-programming is reduce.

→ The switch out process are handled by the medium term schedular.

9) Explain message passing.

→ It is a type of a mechanism that allows processes which synchrones and communicate with each other. However by using message passing, the processes can communicate with each other without restoring the shared variables.

→ It provides two operations, as follows,

(i) Send (Messages (ii) Receive (Message)

following are the methods logically implementing a link it may send(msg) and receive(msg) operation.

(i) Direct or indirect communication.

(ii) Synchrones or Asynchrones communication

(iii) Automatic and explicit buffering.

(i) Direct Communication.

→ In this type of interprocess communication process should make each other explicity. In this method the link is established betn one pair only one link exists.

Indirect communication.

→ Indirect Communication established only when processes share a common mailbox betn each pair of processes, Sharing several communication links. A link can communicate with many processes. The link may be bydirectional or unidirectional.

Synchrones and Asynchrones communication.

Communication betn process takes place send(msg) and receive to calls throughs to send (msg) and receive (msg) premitted.

Blocking send

The Sending process is blocked until the msg is received by the receiving process or by the mailbox.

2. Non-blocking send
The sending process sends the msg and resumes operation

3. Blocking receieve.
The receiever blocks until the msg available.

4. Non-blocking receieve.
The receiever retieves either a valid msg or a null

(iii) Buffering.

→ Whether communication is direct or indirect, msgs exchanged by communicating processes reside from the temporary queue. Such queue can be implemented by 3 ways.

(i) Zero Capacity :-

The queue as a maximum hence, the link cannot have any msg waiting in it. In this case, the Sender must block until the recepient the msg.

(ii) Bounded Capacity.

The queue has finite length hence, the atmost unnumber can reside in it. If the queue is not full when new msg is send the msg is placed in the queue, and the Sender can continue execution without waiting. The links capacity is finite. If the link is full, the Sender must block until the space is available in the queue.

(iii) Unbounded Capacity.

The queue length is potentially infinite. Hence any number of msgs can wait in it. The Sender never blocks.

Explain shared memory.

IPC using shared memory requires communication process to establish a region of shared memory.

A shared memory region resides in the address space of the process creating a shared memory segment other process that wish to communicate using a this shared memory Segment must attach to this address space.

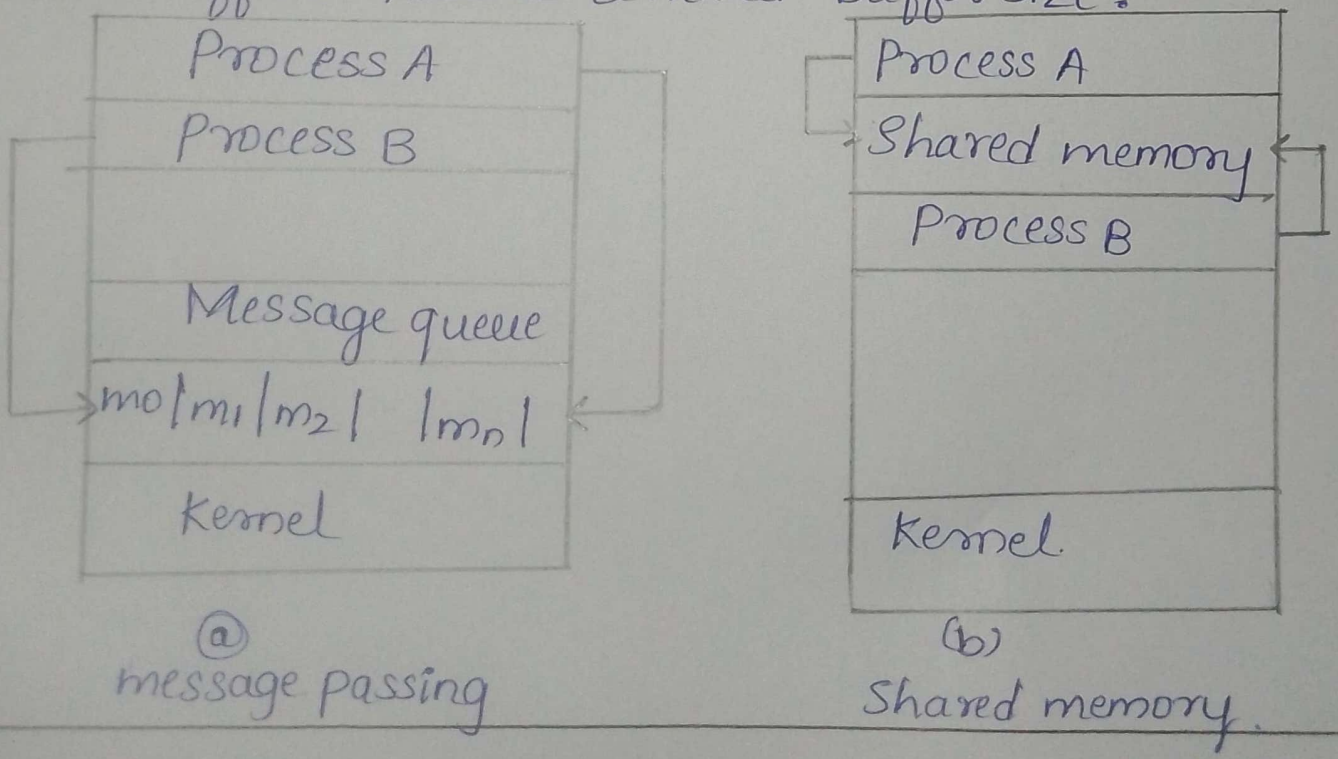
The processes are also responsible for ensuring that they are not writing to the same location simulanteously.

Consider producer consumes problem. A producer process produces information that is consumed by consumer process. A compile may produce assembly code which is consumed by assembler: Therefore, shared memory is used by almost all POSIX and windows operating system as well.

The two types of Buffer are used,

Unbounded Buffer :- places no pratical limit on the size of the buffer.

Bounded Buffer :- Assume a mixed buffer size.



2 Marks.

① Define Deadlock.

→ Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

② What is short term schedular.

→ A short term schedular also known as CPU schedular is a component of the operating system responsible for selecting which process in the ready queue should be executed next by the CPU.

③ What is semaphore? Mention it's type.

A semaphore is a special variable or abstract data type used in operating systems to control access to shared resources by multiple users to prevent race conditions.

Types of semaphore.

(i) Binary semaphore.

(ii) Counting semaphore.

④ Define file. List the different file attributes.

A file is a collection of related information that is stored on secondary storage. File represents both program as well as data.

File attributes:

- Name
- Size
- Identifier
- Type
- Location.
- Time and date

⑤ State the benefits of threads.

- Responsiveness
- faster context switching
- Resource sharing
- Scalability.

⑥ Define Turnaround time.

Turnaround time in operating system is the total time taken to execute a particular process from the moment it is submitted to the moment it is completed.

⑦ Explain the System structure.

File system structure provide efficient access to the disk by allowing data to be stored, located and retrieved in a convinient way. A file system must be able to store the file, locate the file and retrieve the file.

⑧ Distinguish betn Semaphore and monitor.

Semaphore	Monitor.
Low-level synchronization primitive using counters.	High-level synchromiza-tion.
Data type: integer variable + two automic operations.	Data type: abstract data type with built in mutual exclusion.
Harder to use, error prove.	Easier to use, safer du to built in synchromi-zation.

⑨ What is directory?

A directory is a special type of file that contains a list of other files and subdirectories. It helps organize and manage data in a heirarchical file system.

3) What do you mean by SJF.

* SJF is a scheduling algorithm that selects the process with the shortest execution time to run next.
* It can be preemptive or non-preemptive in nature.

---

1) What do you mean by co-operating process.

Co-operating processes are processes that can affect or be affected by other processes during execution. They are designed to work together and share information, resource or co-ordinate their activities.

---

5 Marks questions.

---

1) Explain mutual exclusion, using synchronization hardware.

Mutual exclusion ensures that only process can access a critical section (shared resource) at time. one way to achieve this is by using special hardware instructions that support atomic operations.

Key hardware-based synchronization.

1) Test and set instructions:

→ Atomic instruction that lead & modify a value simultaneously.
→ Prevents race condition.

* Swap:

→ Compares a memory value with an expected value & swaps if they match.

* Unlock and lock.

* lock() acquires lock before entering the critical section.
* Unlock() releases it after exiting, allowing others to enter.

Advantages:

① No need for o.s intervention.
② Very fast, efficient in short critical sections.

Disadvantages:
1) Busy waiting wastes CPU cycles.
2) More suited for kernel-level or low-level concurrency.

② Explain various file operations.

1) Create operation:
→ Creates a file in system.
→ Program calls the file system space is allocated to the file
→ Entry of new file is made into the appropriate directory

② Open operation:-
→ A file name is provided to open the particular file in the system.
→ It tells the operating system to invoke the open system call and passes the file name.

③ Write operation:-
→ Used to write the information into the file.
→ this system call specifies the name of file & length of the data has to be written to the file.

④ Read operation:-
→ Reads the content from the file.
→ The read pointer is maintained by the OS, pointing to the position upto which the data has been read.

⑤ Reposition or seek operation:-
→ Repositions the file pointers from current position to a specific place in the file.

⑥ Truncate operation:-
→ deleting the file except deleting attributes.
→ The file is not completely deleted although the info is replaced.

7) Delete operation:-
→ Used so that disk space occupied by it is freed.
→ After the directory is searched & located, all the associated file space & directory entry is released.

8) close operation:-
→ on closing the file, it deallocates all the internal descriptors that were created when the file was opened.

9) Append operation :-
→ Adds data to the end of the file.

10) Rename operations:-
→ Used to rename the existing file.

) Explain the condition to avoid deadlock.

> Deadlock is a state in which a process is waiting for the resource that is already used by another process and that another process is waiting for some another resource.
> Deadlock avoidance requires that the system has some additional a priori info available.
> Simplest and most useful model requires that each process declare the max no. of resource of each type that it may need.
> The deadlock avoidance algorithm dynamically examines the resource allocation state to ensure that there can never be a circular-wait condition.
> Resource-allocation state is defined by the number of available & allocated resources & the maximum demands of the process.

Following techniques used to avoid deadlock, ensures that the system is in safe state:
Resource Allocation Graph.
Banker's Algorithm

④ Write a short note on file access.

File access:

File access in an OS refers to the methods used to retrieve and manipulate data stored in files. File access methods are:

1. Sequential Access :-

→ Reads data in a linear order, from beginning to end.

→ Commonly used in app's like text editors & log processing where data is to be read sequentially.

Advantages → i) Simplicity
ii) optimized for sequential tasks.

Disadvantages → i) slow for random access.
(ii) Limited flexibility.

2. Direct Access:

→ Allows access to any part of the file directly, without reading previous data.

→ Useful for applications that require quick retrieval of data.

Advantages → (i) speed
(ii) Flexibility.

Disadvantages → (i) Complex implementation.
(ii) higher storage overhead.

---

⑤ Explain FCFS and SJF scheduling with its advantages and disadvantages.

* First come First Serve (FCFS)

→ One of the simplest CPU scheduling algorithms.

→ Works like a queue – processes are executed in order they arrive without preemption.

Characteristics of FCFS:
1. Non-preemptive
2) FIFO
3. High waiting time.

Advantages — 1. Simplicity
2. Fairness
3. No Starvation.

Disadvantages — 1. Poor avg waiting time.
2. Not suitable for time sharing systems

* Shortest Job First.
- Selects the process with shortest execution time sharing to be executed next.
- It can be preemptive or non-preemptive.

Advantages — 1. Min avg waiting time.
2. Efficient use of resources.
3. Reduced turnaround time.

Disadvantages — 1. Difficult to predict burst time.
2. Starvation of long process.

* Characteristics of SJF
1. Minimize avg waiting time.
2. Can cause starvation.

Explain necessary characteristics of deadlocks.
Characteristics of deadlocks.

Mutual exclusion:
only one process at a time can use a resource. If another process requests that resource, requesting process must until the resource has been released.

② Hold and wait

A process holding atleast one resource is waiting to acquire additional resources held by other processes.

③ No preemption

Resources can be released only voluntarily by the process holding it, after that process has completed its task.

④ Circular wait:

There exists a set {p0, p1, ---Pn} of waiting process such that P0 is waiting for a resource that is held by P1, P2, P3 waiting for a resource that is held by P2---- Pn-1 is waiting for a resource that is held by Pn & Pn is waiting for a resource that is held by P0.

---

⑦ Explain critical section.

Critical Section.

A critical section is a part of a program where the process accesses shared resources. Since these resources can be accessed by more than one process, accessing them simultaneously may lead to data inconsistency or corruption.

⑧ Key problem.

RACE CONDITION

If two processes access and modify a shared resource at the same time, the result depends on the timing of their execution, a situation called a race condition. The critical section problem is about preventing this.

Deadlock prevention techniques.

Deadlock prevention works by negating atleast one of four conditions.

① Eliminate Mutual Exclusion.

Not usually practical since many resources must be used exclusively.

② Eliminate Hold and wait:

Require processes to request all required resource at once.

③ Eliminate no preemption.

If a process holding resources that is not immediat available, force it to release its current resources.

④ Eliminate circular wait:

Impose a strict ordering of resource acquisitions prevents the chain of waiting.

---

⑨ Explain Dinning philosopher's problem of Synchronization.

→ Consider 5 philosopher's to speed their lives in thinking and eating.

→ A philosopher shares common circular table surrounded by 5 chairs each occupies by one philosopher.

→ In the center of the table, there is a bowl of rice & the table is laid with 5 chopsticks arranged adjacent to plates.

→ When a philospher thinks she does not interact with her colleagues.

→ From time to time a philosopher gets hungry & tries to pickup two chopsticks that are closest to her.

2. Define Peterson's Solution.

Peterson's solution is a classical software-based solution to the critical section problem. In peterson's solution, we have two shared variables.

- boolean flag[i]: Initialized to false, initially no one is interested in entering the critical section.
- int turn: The process whose turn is to enter the critical section.

Peterson's solution preserves all three conditions.

1. Mutual exclusion.
2. Progress
3. Bounded waiting.

Disadvantages of peterson's solution.

1. Involves busy waiting.
2) It is limited to 2 processes.
3. Peterson's solution cannot be used in modern CPU architecture.

Describe Sequential file access and direct file access methods of file.

Sequential file access

Sequential access refers to a method of reading or writing data to a file or device in a linear, ordered manner from the beginning to the end, one record or byte at a time.

Advantages:

1. Efficiency.
2. Simplicity
3. lower overhead.
4. Suitable for Streaming.

Disadvantages.
1) Slow for random access.
2) Limited flexibility.

(11) Direct access.

Direct access refers to the ability to read or write data at any location in a file or storage device without having to process all the data before it.

Advantages.
1) Fast random lookups.
2) Ideal for large databases.
3) Efficient memory paging and file updates.

Disadvantages.
1) Complex implementation.
2) higher storage overhead.

(12) Explain Banker's algorithm of deadlock avoidance.

Banker's Algorithm.

- The Bankers algorithm get it's name because it is method that bankers could use to assure that when they lend out resources they will still be able to satisfy all their clients.
- When a process starts up, it must state in advance the maximum allocation of resources it may request, upto the amount available on the system.
- When the request is made, scheduler determines whether granting the request would leave the system in a safe state. If not, then the process must wait until the request can be granted safely.
- Available [m] – how many resources are currently available of each type.

- Max[n][m] – max demand of each process of each resource.
- Allocation[n][m] – no. of each resource category allocated to each process.
- Need[n][m] – remaining resources needed of each type for each process.

Need[i][j] = Max[i][j] – Allocation[i][j] for all i, j.

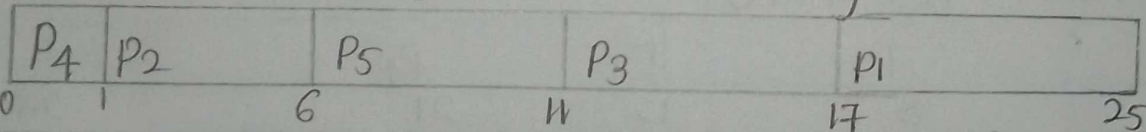
10 Marks questions.

Consider the following set of processes with CPU Burst time.

Process	Burst time
P_1	8
P_2	4
P_3	6
P_4	2
P_5	5

ⓐ Draw a Gantt chart to show execution using SJF scheduling
ⓑ Calculate avg Turnaround Time for SJF scheduling.
ⓒ Calculate avg Waiting Time & average response time using SJF scheduling.

Gantt chart [using SJF scheduling]



Process	Burst time	completion time	Turnaround time	waiting time.
P_1	8	25	25-0=25	25-8=17
P_2	4	6	6-0=6	6-4=2
P_3	6	17	17-0=17	17-6=11
P_4	2	2	2-0=2	2-2=0
P_5	5	11	11-0=11	11-5=6

Average Turn Around time = (25+6+17+2+11)/5

$= \frac{61}{5} = 12.2$

∴ Average waiting Time $= \frac{(17+2+11+0+6)}{5}$

$= \frac{36}{5} = 7.2$

② Consider the following set of processes with CPU burst time and arrival time given in milliseconds.

Process	Arrival Time	Burst time
P_1	0	8ms
P_2	1	4ms
P_3	2	9ms
P_4	3	5ms

ⓐ Draw Gantt charts illustrating the execution of the processes with

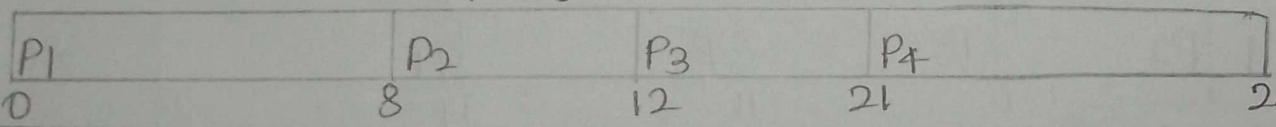
(i) FCFS

(ii) Preemptive SJF

ⓑ Calculate average waiting time for each

ⓒ Calculate average Turn around time for each.

→ Gantt Chart for FCFS.



Process	Arival time	Burst time	completion time	Turn around time	waiti time
A	0	8ms	8	8-0=8ms	8-8=0
P_2	1	4ms	12	12-1=11ms	11-4=
P_3	2	9ms	21	21-2=19ms	19-9=
P_4	3	5ms	26	26-3=23ms	23-5

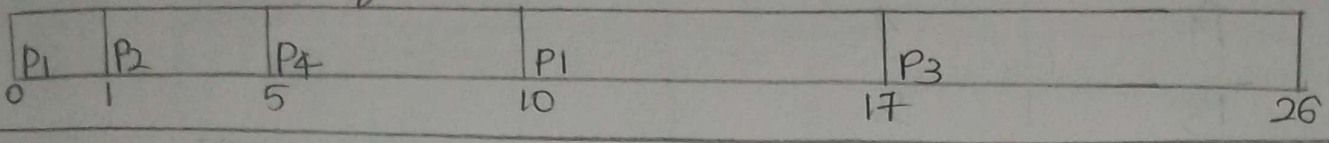
∴ Average Turn around time = $\frac{(8+11+19+23)}{4}$

$= \frac{61}{4} = 15.25$ ms.

∴ Average waiting time = $\frac{(0+7+10+18)}{4}$

$= 35/4 = 8.75$

Gantt chart for SJF.



Process	Arrival time	Burst time	completion time	Turn Around time.
P1	0	8ms	17ms	17-0 = 17ms
P2	1	4ms	5ms	5-1 = 4ms
P3	2	9ms	26ms	26-2 = 24ms
P4	3	5ms	10ms	10-3 = 7ms

Waiting time for $P_1 = 10-1-0 = 9$ms

Waiting time for $P_2 = 1-0-1 = 0$ms

Waiting time for $P_3 = 17-0-2 = 15$ms

Waiting time for $P_4 = 5-0-3 = 2$ms

∴ Average Waiting Time = $\frac{(9+0+15+2)}{4}$

$= \frac{26}{4}$

$= 6.5$ms

∴ Average Turn Around time = $\frac{(17+4+24+7)}{4}$

$= 52/4$

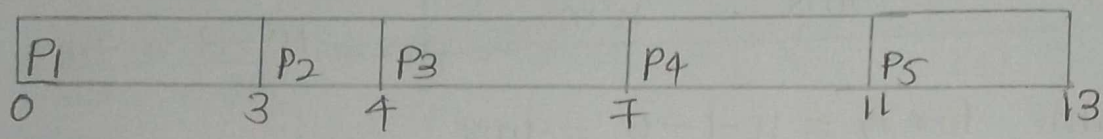
$= 13$ms.

③ Consider the following set of processes, with the length of CPU Burst time given in milliseconds.

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

ⓐ Draw Gantt chart illustrating the execution of these Process using FCFS & Round Robin (quantum time = 1ms) method. ⓑ Calculate the average waiting time & average turnaround time for both of these scheduling method.

Gantt chart for FCFS.



Process	Burst time	completion time	Turnaround time	waiting time.
P_1	3	3	3-0 = 3ms	3-3 = 0
P_2	1	4	4-0 = 4ms	4-1 = 3
P_3	3	7	7-0 = 7ms	7-3 = 4
P_4	4	11	11-0 = 11ms	11-4 = 7
P_5	2	13	13-0 = 13ms	13-2 = 11

∴ Average Turn around time = $\frac{(3+4+7+11+13)}{5}$

= 38/5 = 7.6

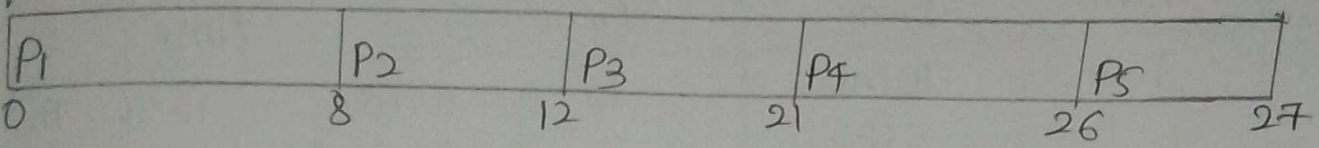
∴ Average waiting time = $\frac{(0+3+4+7+11)}{5}$ = 25/5

= 5

(b) Calculate average waiting time for each scheduling algorithm.

(c) Calculate average turnaround time for each scheduling algorithm.

→ Gantt chart for FCFS



Process	Burst time	Completion time	Turn Around time	waiting time.
P1	8ms	8ms	8-0 = 8ms	8-8 = 0ms
P2	4ms	12ms	12-0 = 12ms	12-4 = 8ms
P3	9ms	21ms	21-0 = 21ms	21-9 = 12ms
P4	5ms	26ms	26-0 = 26ms	26-5 = 21ms
P5	1ms	27ms	27-0 = 27ms	27-1 = 26ms

∴ Average Turn Around time = $\frac{(8+12+21+26+27)}{5}$

= 94/5

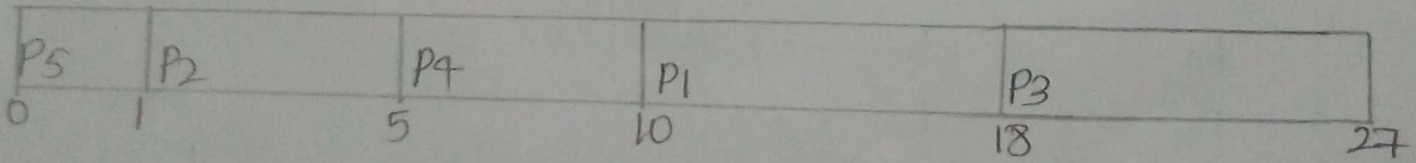
= 18.8

∴ Average Waiting time = $\frac{(0+8+12+21+26)}{5}$

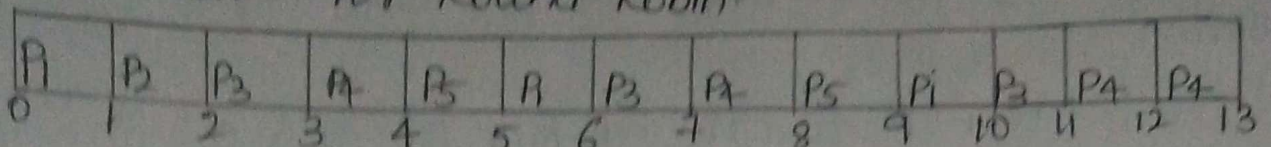
= 67/5

= 13.4

Gantt chart for SJF



Gantt chart for Round Robin



Process	Burst time	completion time	Turn Around time	waiting time
P_1	3	10ms	10-0 = 10ms	10-3 = 7ms
P_2	1	2ms	2-0 = 2ms	2-1 = 1ms
P_3	3	11ms	11-0 = 11ms	11-3 = 7ms
P_4	4	13ms	13-0 = 13ms	13-4 = 9ms
P_5	2	9ms	9-0 = 9ms	9-2 = 7ms

∴ Average Waiting time = $\frac{(7+1+7+9+7)}{5}$

$= \frac{31}{5} = 6.2$

∴ Average Turn Around time = $\frac{(10+2+11+13+9)}{5}$

$= \frac{45}{5} = 9$

④ Consider the following process, with CPU Burst time given in milliseconds.

Process	Burst time
P_1	8
P_2	4
P_3	9
P_4	5
P_5	1

@ Draw Gantt charts to show execution using FCFS & SJF scheduling.

process	Burst time	completion time	Turn around time	waiting time.
P1	8	18ms	18-0=18ms	18-8=10ms
P2	4	5ms	5-0=5ms	5-4=1ms
P3	9	27ms	27-0=27ms	27-9=18ms
P4	5	10ms	10-0=10ms	10-5=5ms
P5	1	1ms	1-0=ms	1-1=0ms

∴ Average Turn Around time = $\frac{(18+5+27+10+1)}{5}$

$= 61/5$

$= 12.2$

∴ Average waiting time = $\frac{(10+1+18+5+0)}{5}$

$= \frac{34}{5} = 6.8$

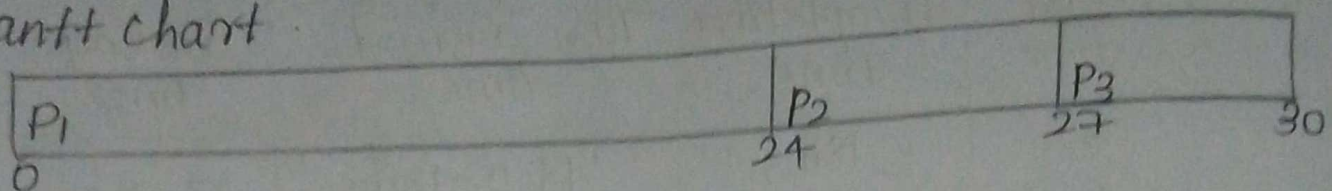
Consider the following set of processes with CPU.

Process	Burst time (in ms)
P1	24
P2	3
P3	3

(a) Draw Gantt charts illustrating the execution of these process using FCFS and SJF scheduling.

(b) Calculate average waiting and average turnaround time in each case.

Gantt chart



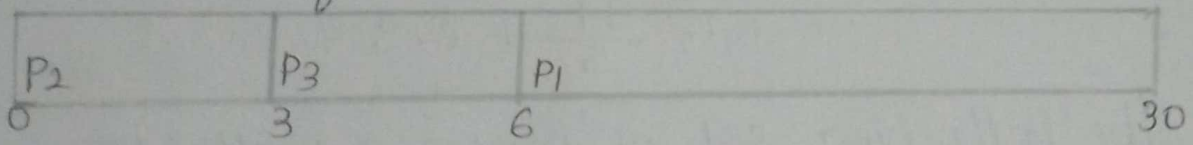
Process	Burst time	completion time	Turnaround time	waiting time
P1	24	24	24-0=24	24-24=0
P2	3	27	27-0=27	27-3=24
P3	3	30	30-0=30	30-3=27

∴ Average Turn Around time = $\frac{(24+27+30)}{3}$

$= 81/3 = 27$

∴ Average waiting time = $\frac{(0+24+27)}{3} = 51/3 = 17$

Gantt chart for SJF



Process	Burst time	completion time	Turnaround time	waiting time.
P1	24	30	24-0=24	30-24=6
P2	3	3	3-0=3	3-3=0
P3	3	6	6-0=6	6-3=3

∴ Average Turn around time = $\frac{(24+3+6)}{3}$

$= 33/3 = 11$

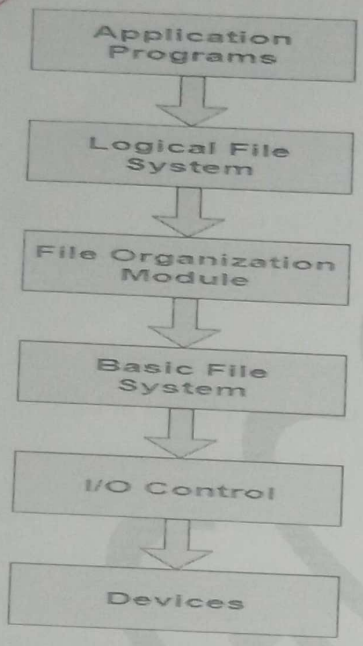
∴ Average waiting time = $\frac{6+0+3}{3} = 9/3 = 3$

File System Structure

File System provide efficient access to the disk by allowing data to be stored, located and retrieved in a convenient way. A file System must be able to store the file, locate the file and retrieve the file.

Most of the Operating Systems use layering approach for every task including file systems. Every layer of the file system is responsible for some activities.

The image shown below, elaborates how the file system is divided in different layers, and also the functionality of each layer.



- When an application program asks for a file, the first request is directed to the logical file system.
- The logical file system contains the Meta data of the file and directory structure. If the application program doesn't have the required permissions of the file then this layer will throw an error. Logical file systems also verify the path to the file.
- Generally, files are divided into various logical blocks. Files are to be stored in the hard disk and to be retrieved from the hard disk. Hard disk is divided into various tracks and sectors. Therefore, in order to store and retrieve the files, the logical blocks need to be

mapped to physical blocks. This mapping is done by File organization module. It is also responsible for free space management.

- Once File organization module decided which physical block the application program needs, it passes this information to basic file system. The basic file system is responsible for issuing the commands to I/O control in order to fetch those blocks.
- I/O controls contain the codes by using which it can access hard disk. These codes are known as device drivers. I/O controls are also responsible for handling interrupts.

Files attributes and its operations:

Types of file attributes (Imp 2M 06 SM)

Imp

Attributes	Types	Operations
Name	Doc	Create
Type	Exe	Open
Size	Jpg	Read
Creation Data	Xls	Write
Author	C	Append
Last Modified	Java	Truncate
protection	class	Delete
		Close

When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. On closing it deallocates all the internal descriptors that were created when the file was opened.

9. Append operation:

This operation adds data to the end of the file.

10. Rename operation:

This operation is used to rename the existing file.

Attributes of the File

2M

1.Name

Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.

2.Identifier

This unique tag, usually a number, identifies the file within the file system. It is non-human-readable name for the file.

3.Type

In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.

4.Location

In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.

5.Size

The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.

6.Protection

Each file carries its own set of permissions to the different group of Users. This information determines who can do reading, writing, executing and so on.

7.Time and Date

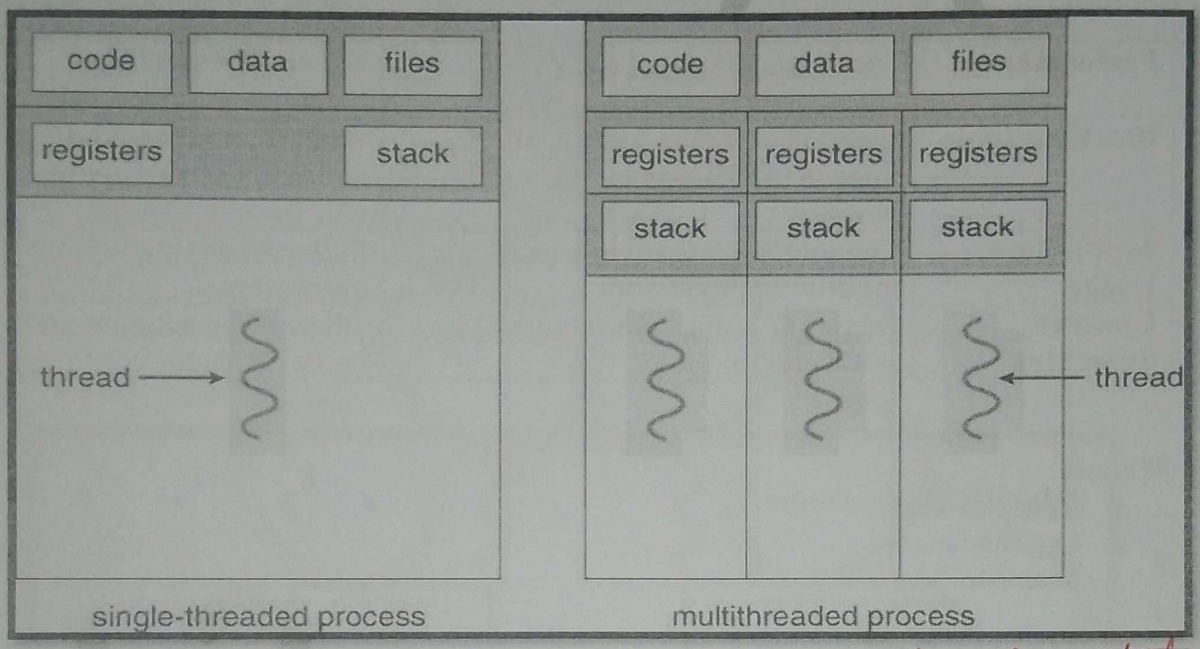
Sanjana Menarlu
Sanjana. Menashi

2M

Multithreaded Programming: Introduction to Threads; Types of Threads; Multithreading- Definition, Advantages; Multithreading Models; Thread Libraries; Threading Issues.
CPU Scheduling: Basic concepts; Scheduling Criteria; Scheduling Algorithms; Multiple-processor scheduling; Thread scheduling; Multiprocessor Scheduling; Real-Time CPU Scheduling.

Threads:

1. Thread is basic unit of execution
2. It comprises of Thread Id, PC, Register Set and Stack
3. It is called Light weight process (LWP)



If they asked only state means Example Explain them.

Benefits of multithreaded programming:

1. Responsiveness – may allow continued execution if part of process is blocked, especially important for user interfaces
2. Resource Sharing – threads share resources of process, easier than shared memory or message passing
3. Economy – cheaper than process creation, thread switching lower overhead than context switching
4. Scalability – Different threads can run in parallel on different processors, increasing speed and efficiency.

Running multiple threads in parallel to improve performance, especially on multi-core systems is called multithreading.

increasing speed and efficiency.

Multithreading Models:

1. User threads- management done by user-level threads library
2. Kernel threads- Supported by the Kernel

Difference Between User Thread and Kernel Thread

Feature	User Thread	Kernel Thread
Managed By	User-level thread library (inside the application)	Operating System (OS) Kernel
OS Awareness	OS is unaware of user threads	OS knows and manages kernel threads
Performance	Faster (no OS overhead)	Slower due to OS management overhead
Blocking Issue	If one user thread blocks, all threads in that process may be blocked	One blocked thread does not affect others
Multi-Core Usage	Cannot take full advantage of multi-core processors	Fully utilizes multi-core processors
Context Switching	Faster, as it does not require OS intervention	OS knows and manages kernel threads

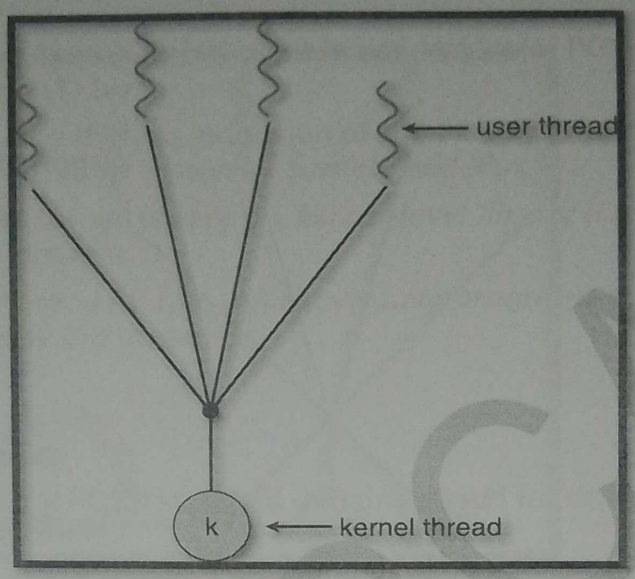
Models

1. Many-to-One
2. One-to-One
3. Many-to-Many

I. Many-to-One:

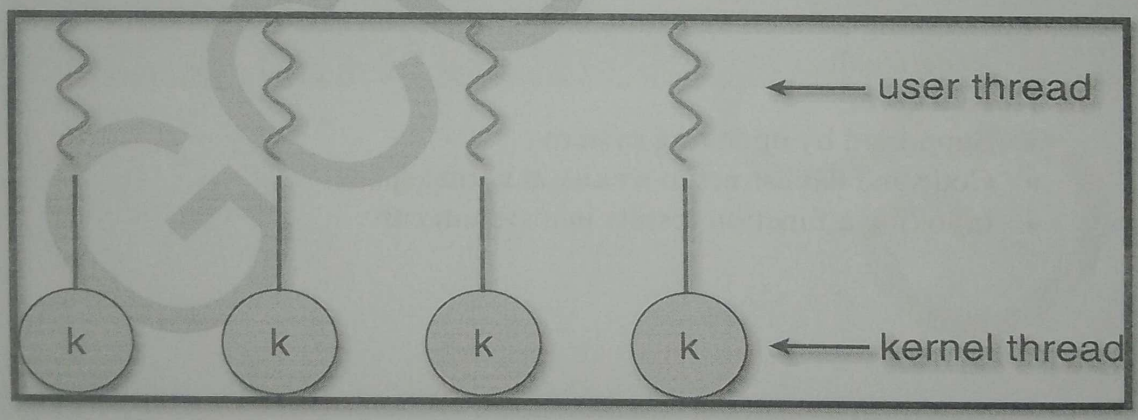
1. Many user-level threads mapped to single kernel thread
2. Thread management is done by the thread library in user space
3. One thread blocking causes all to block
4. Multiple threads may not run in parallel on multicore system because only one kernel thread may exist at a time

Deadlock →



II. One-to-One with diagram

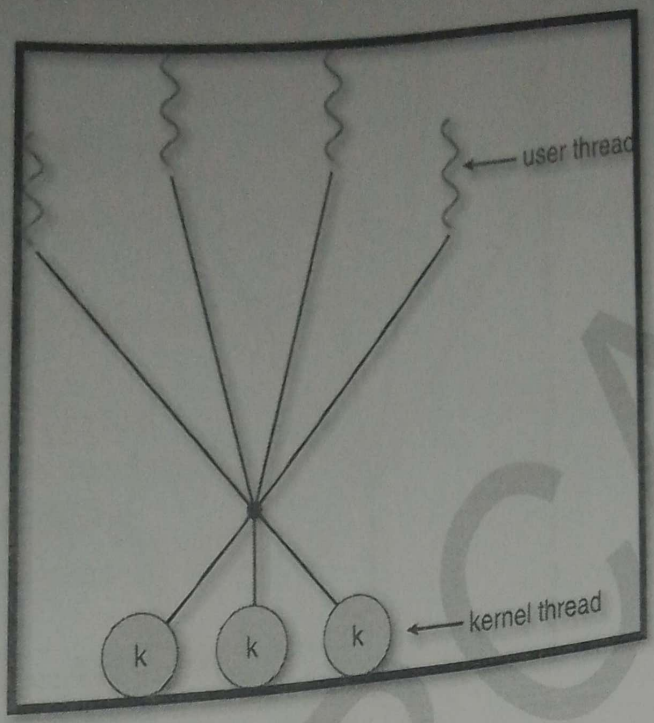
1. Each user-level thread maps to kernel thread
2. Creating a user-level thread creates a kernel thread
3. More concurrency than many-to-one
4. Number of threads per process sometimes restricted due to overhead



III. Many-to-Many:

1. Allows many user level threads to be mapped to many kernel threads
2. Allows the operating system to create a sufficient number of kernel threads

What is semaphore? Mention its types



Thread Libraries:

1. A thread library provides an API to the programmer for creating and managing threads.
2. Two methods to implement thread library
   a. **User Level**
      - Supported by users, with no kernel support
      - Invoking a function results in local function call in user space not a system call
   b. **Kernal level**
      - Supported by operating system
      - Code and data structures exist in kernel space
      - Invoking a function results in a system call

- Creating a **Thread object** does not create the **new thread**; rather, it is the ***start()*** method that creates the **new thread**.
- Calling the ***start()*** method for the new object does two things:

1) It allocates memory and initializes a **new thread** in the JVM.
2) It calls the ***run()*** method, making the thread eligible to be run by the JVM. (Note that we never call the *run()* method directly. Rather, we call the *start()* method, and it calls the *run()* method on our behalf.)

Threading Issues: Imp

1. The fork() and exec() system calls
2. Signal handling
3. Thread cancellation
4. Thread local storage
5. Schedular Activation

The fork() and exec() system calls:

1. fork() creates a new process by duplicating the calling process, The new process, referred to as child, is an exact duplicate of the calling process, referred to as parent.
2. The exec() family of functions **replaces** the current process image with a new process image. It loads the program into the current process space and runs it from the entry point.

Thread cancellation:

Terminates a thread before completion.

2 ways of cancellation:

Asynchronous cancellation – thread terminates immediately

Deferred cancellation – checks if it should terminate

Signal Handling:

Pattern to follow:

1. A signal is generated by the occurrence of a particular event
2. A generated signal is delivered to a process
3. Once delivered, the signal must be handled

Example: illegal memory access, division by 0

Signal Handlers:

1. A default signal handler (kernel)
2. A user-defined signal handler

Thread Pools:

A thread pool reuses previously created threads to execute current tasks and offers a solution to the problem of thread cycle overhead and resource thrashing

Thread-Specific Data:

1. Multi-threaded programming shares data
2. Some threads might need separate copies of the data
3. Example: transaction processing
4. Win32, Java and Pthread supports thread-specific data

Scheduler Activations:

1. One technique for communication between the user-thread library and the kernel is known as **scheduler activation**.
2. The kernel provides an application with a set of virtual processors (LWPs), and the application can schedule user threads onto an available virtual processor. Moreover, the kernel must inform an application about certain events. This procedure is known as an upcall.
3. Upcalls are handled by the thread library with an upcall handler, and upcall handlers must run on a virtual processor.

CPU Scheduling: 2M

1. **Definition**: It is the process of selecting a process from the ready queue and allocating the CPU to it. The time that we are assigning to different process for their utilization of the CPU is called as CPU Scheduling. It is the fundamental function in a multiprogramming OS.
2. The OS switches the CPU among processes to maximize CPU utilization.
3. In a single processor system, only one process runs at a time; others wait until the CPU is available.

CPU Scheduler

Whenever the CPU becomes idle, the OS must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects a process from the processes in memory that we ready to execute and allocates the CPU to that process

CPU scheduling decisions may take place when a process:

1. Switches from running to waiting state
2. Switches from running to ready state (ex: when an interrupt occurs)
3. Switches from waiting to ready (ex: at completion of I/O)
4. Terminates

Decisions:

For situation 1 and 4, there is no choice in terms of scheduling. A new process (if one exists in the ready queue) must be selected for execution. However, there is a choice for situations 2 and 3.

When scheduling takes place only under circumstances 1 and 4, we say that the scheduling scheme is non-preemptive or cooperative, otherwise it is preemptive

...ing are: Shortest Job First (SJF Preemptive) and Priority (non-preemptive version), etc.

Difference between Preemptive Scheduling & Non-Preemptive Scheduling:

Preemptive Scheduling	Non-Preemptive Scheduling
In this resources(CPU Cycle) are allocated to a process for a limited time.	Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state
Process can be interrupted in between.	Process can not be interrupted until it terminates itself or its time is up
Low priority process may starve	Short timed process may starve
Flexible	Rigid
Waiting time is less	Waiting time is more
Response time is less	Response time is more
It has overheads of scheduling the processes and high overhead due to context switching	It does not have overheads since context switching is less frequent
CPU utilization is high	CPU utilization is low
Decisions are made by the scheduler and are based on priority and time slice allocation	Decisions are made by the process itself and the OS just follows the process's instructions
Examples of preemptive scheduling are Round Robin and Shortest Job First	Examples of non-preemptive scheduling are First Come First Serve and Shortest Job First

Multiple-Processor Scheduling: (SMP Imp)

In multiple-processor scheduling multiple CPU's are available and hence Load Sharing becomes possible. However multiple processor scheduling is more complex as compared to single processor scheduling

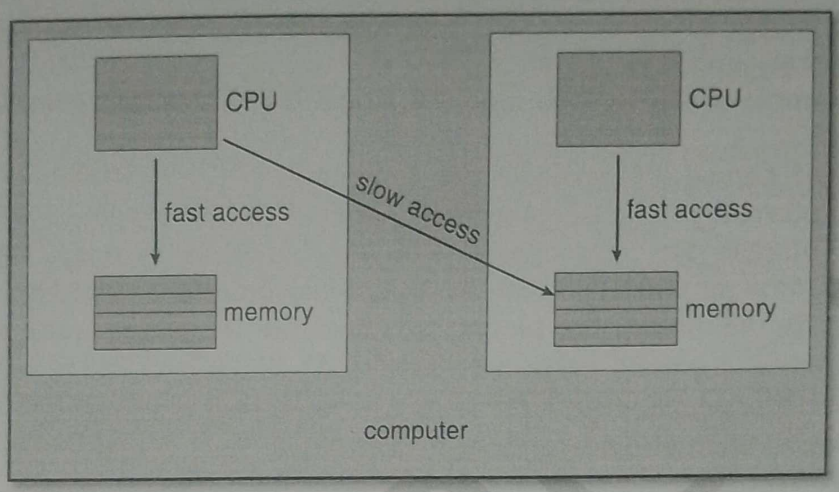
Approaches to Multiple-Processor Scheduling:

1. **Asymmetric multiprocessing** – only one processor accesses the system data structures, reducing the need for data sharing
2. **Symmetric multiprocessing (SMP)** – each processor is self-scheduling, all processes in common ready queue, or each has its own private queue of ready processes

Processor affinity –

1. Binds the processes or threads to a specific CPU with the help of cache memory(L1, L2, L3).
2. Process has affinity for processor on which it is currently running
3. The 2 types of affinity are:
   a. **soft affinity:** processes run on the same processor
   b. **hard affinity:** processes keep switching between different processor
4. The main-memory architecture of a system can affect processor affinity issues.
5. The figure below illustrates an architecture featuring **Non-Uniform Memory Access (NUMA),** in which a CPU has faster access to some parts

of main memory than to other parts.



6. The memory access time depends on the memory location.
7. It will have multiple nodes and each of it will have a processor or group of processors and a local memory.

Load Balancing:

1. **Load balancing attempts to keep workload evenly distributed across all processors in a SMP system.**
2. There are 2 general approaches to load balancing:
   a. **Push migration** – a specific task periodically checks the load on each processor and if it finds an imbalance, it evenly distributes the load by moving (or pushing) processes from overloaded to idle or less-busy processors.
   b. **Pull migration** – occurs when an idle processor pulls a waiting task from busy processor.

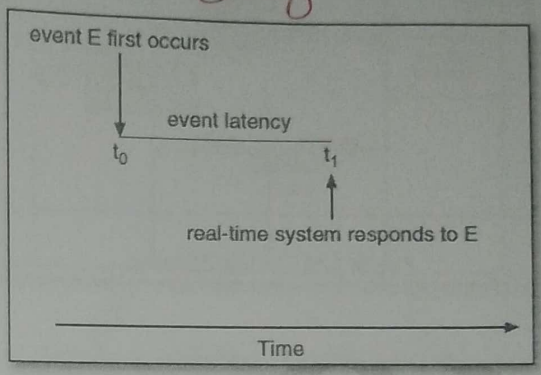
Real-Time CPU Scheduling

1. CPU scheduling for real-time operating systems involves special issues. In general, we can distinguish between soft real-time systems and hard real-time systems.
2. Soft real-time systems provide no guarantee as to when a critical real- time process will be scheduled.
3. They guarantee only that the process will be given preference over noncritical processes.

4. Hard real-time systems have stricter requirements. A task must be serviced by its deadline; service after the deadline has expired is the same as no service at all.

Minimizing Latency

SRA with diagram @write its types.



1. Consider the event-driven nature of a real-time system.
2. The system is typically waiting for an event in real time to occur.
3. Events may arise in software - as when a timer expires - or in hardware - as when a remote-controlled vehicle detects that it is approaching an obstruction.
4. When an event occurs, the system must respond to and service it as quickly as possible.
5. We refer to event latency as the amount of time that elapses from when an event occurs to when it is serviced
6. Usually, different events have different latency requirements.
7. For example, the latency requirement for an antilock brake system might be 3 to 5 milliseconds.
8. That is, from the time a wheel first detects that it is sliding, the system controlling the antilock brakes has 3 to 5 milliseconds to respond to and control the situation.
9. Any response that takes longer might result in the automobile's veering out of control.
10. In contrast, embedded system controlling radar in an airliner might tolerate a latency period of several seconds.

Two types of latencies affect the performance of real-time systems:

Interrupt latency

Dispatch latency

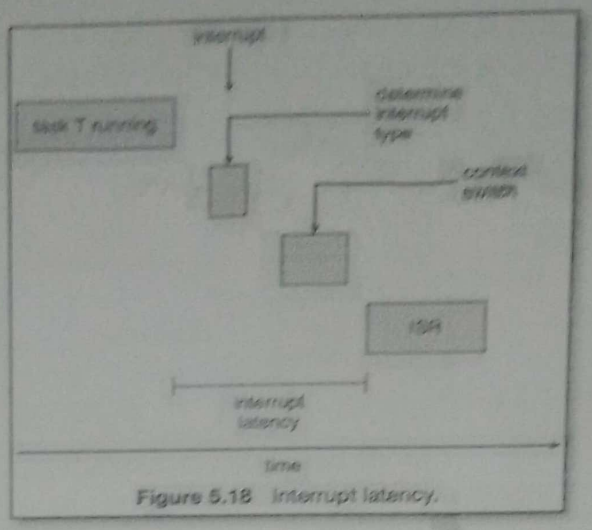
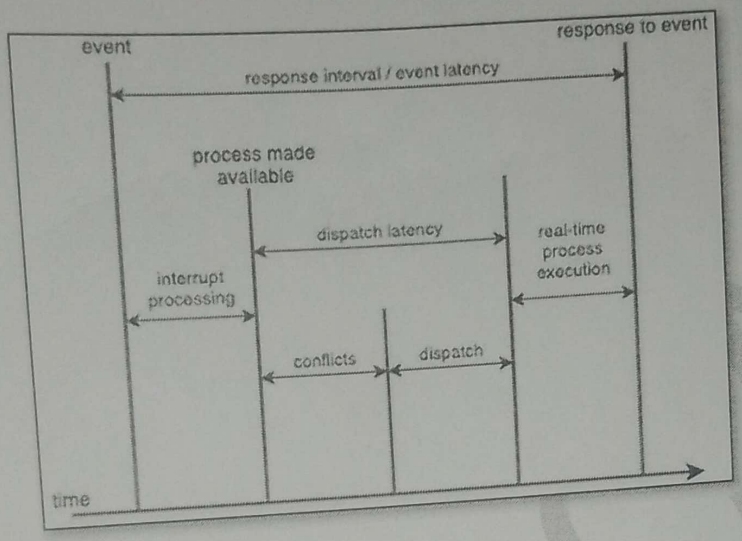


Figure 5.18 Interrupt latency.

1. Interrupt latency refers to the period of time from the arrival of an interrupt at the CPU to the start of the routine that services the interrupt.
2. When an interrupt occurs, the operating system must first complete the instruction it is executing and determine the type of interrupt that occurred.
3. It must then save the state of the current process before servicing the interrupt using the specific interrupt service routine (ISR).
4. The total time required to perform these tasks is the interrupt latency
5. Obviously, it is crucial for real-time operating systems to minimize interrupt latency to ensure that real-time tasks receive immediate attention.
6. Indeed, for hard real-time systems, interrupt latency must not simply be minimized; it must be bounded to meet the strict requirements of these systems.
7. One important factor contributing to interrupt latency is the amount of time interrupts may be disabled while kernel data structures are being updated.
8. Real-time operating systems require that interrupts be disabled for only very short periods of time.
9. The amount of time required for the scheduling dispatcher to stop one process and start another is known as dispatch latency.
10. Providing real-time tasks with immediate access to the CPU mandates that real-time operating systems minimize this latency as well.
11. The most effective technique for keeping dispatch latency low is to provide preemptive kernels.
12. For hard real-time systems, dispatch latency is typically measured in several microseconds.



The above figure shows dispatch latency. The conflict phase of dispatch latency has two components:

1. Preemption of any process running in the kernel
2. Release by low-priority processes of resources needed by a high-priority process

Following the conflict phase, the dispatch phase schedules the high-priority process onto an available CPU.