Synchronization/Critical Section

1) What is Race condition?

→ A situation like this, where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called a race condition.

2) What is concurrent process?

3) What is critical section? Explain with example how critical problem arises the two or more processes access a shared variable concurrently?

→ Consider a system consisting of n processes $\{P_0, P_1, \ldots P_{n-1}\}$. Each process has a segment of code, called a critical section, in which the process may changing common variables, updating a table, writing a file, and so on. The important feature of the system is that, when one process is executing in the critical section, no other process is allow to executing in it's critical section. That is no, two processes executing their critical section at the same time.

The critical-section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter it's critical section. The section of code implementing this request is the entry section. The critical section may be followed by an exit section. The remaining code is the remainder section. The general structure of a typical process is shown below, -

```
do {
    entry section
        Critical Section
    exit section
        Remainder section
} while (true);
```

4) What are the different ways to ~~above~~ solve a critical section problem?

→ • Mutual exclusion:- If process Pi is executing in its critical section, then no other processes can be executing in their critical sections.

• Progress:- If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection can not be postponed ~~indef.~~ indefinitely.

• Bounded waiting:- There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

(5) what is semaphore and binary semaphore? what does 0 and 1 represent in Binary semaphore?

→ A semaphore S is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: wait( ) and signal( ). The wait( ) operation was originally termed P; signal( ) was originally called V. The definition of wait( ) is as follows:

```
wait (S) {
    while (S <= 0)
      ; // busy wait.
    S--;
}
```

The definition of signal( ) is as follows:

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

All modifications to the integer value of the semaphore in the wait( ) and signal( ) operations must be executed ~~indivisi~~ indivisibly. That is, when one process modifies the semaphore value, no other process can simultaneously modify that same semaphore value. In addition, in the case of wait (S).

Operating systems often distinguish between counting and binary semaphores. The value of a counting semaphore can range over an unrestricted domain. The value of a binary semaphore can range only between 0 and 1. Thus, binary semaphores behave similarly to mutex locks. In fact, on systems that do not provide mutex locks, binary semaphores can be used instead for providing mutual exclusion.

6) Describe the P & V operation briefly.

7) Explain the use of Semaphore for process synchronization.

8) If a ~~binary~~ binary semaphore is is unintentionally ~~initialized~~ initialized to zero, then what will be its consequences? (os answer sheet)

→ Same as Q. NO - 2

9) write a solution of Dining philosopher problem.

→ Main copy

10) Reader, writer problem?

→ Main Copy

Deadlock

1) what is Deadlock?

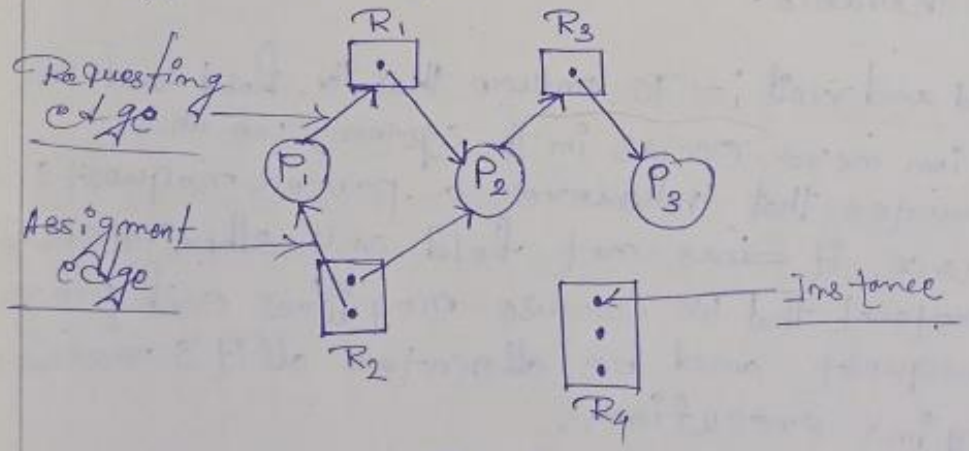
→ In a multiprogramming environment several processes may compete for a finite number of resources. A process requests resources; if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a deadlock.

2) what is Resource allocation Graph? Explain. Briefly discuss about the Connection with deadlock.

3) what is assignment and requesting edge?

→ Deadlocks can be described more precisely in terms of a directed graph called a system resource-allocation graph. This graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two different types of nodes; $P = \{P_1, P_2, \ldots P_n\}$ the set consisting of all the active processes in the system, and $R = \{R_1, R_2, \ldots R_m\}$, the set consisting of all resource types in the system.

A directed edge $P_i \rightarrow R_j$ is called a request edge; a directed edge $R_j \rightarrow P_i$ is called assignment edge.



Resource allocation graph

4) Briefly discuss the necessary and sufficient Condition for occurance of deadlock.

→ (i) A process can initially request any number of instances of a resource type — say, R_i. After that, the process can request instances of resource type R_j if and only if $F(R_j) > F(R_i)$.

(ii) Alternatively, we can require that a process requesting an instance of resource type R_j must have released any resources R_i such that $F(R_i) \geq F(R_j)$.

5) Briefly discuss about the deadlock prevention conditions?

→ • Mutual Exclusion :- The mutual exclusion condition must hold. That is, at least one resource must be nonshareable. Shareable resources, in contrast, do not require mutually exclusive access and thus cannot be involved in a deadlock.

In general, however we cannot prevent deadlocks by denying the mutual-exclusion condition, because some resources are intrinsically nonshareable.

• Hold and wait :- To ensure that the hold-and-wait condition never occurs in the system, we must guarantee that whenever a process requests a resource, it does not hold any other resources. One protocol that we can use requires each process to request and be allocated all it's resources it begins execution.

No preemption

- To ensure that this condition does not hold, we can use the following protocol. If a process is holding some resources and requests another resource that cannot be immediately allocated to it, then all resources the process is currently holding are preempted.

- Circular wait :- To ensure that this condition never holds is to impose a total ordering of all resource types and to require that each process requests resource in an increasing order of ~~enum~~ enumeration.

To illustrate, we let $R = \{R_1, R_2 \ldots, R_m\}$ be the set of resource types. We assign to each resource type a unique integer number, which allows us to compare two resources and to determine whether one precedes another in our ordering. Formally, we ~~dff~~ define a one-to-one function $F: R \to N$, where N is the set of natual numbers.

6) Briefly discuss different deadlock handling mechanisms

→
- We can use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlocked state.
- We can allow the system to enter a deadlock state, detect it and recover.
- We can ignore the problem altogether and pretend that deadlocks never occur in the system.

7) Briefly discuss deadlock avoidance condition/ Bankers algorithm (Safety and resource request algorithm)

→ A state is safe if the system can allocate resources to each process in some order and still avoid a deadlock. More formally, a system is in a safe state only if there exists a safe sequence. A sequence of processes $<P_1, P_2, \ldots P_n>$ is a safe sequence for the current allocation state if, for each P_i, the resource requests that P_i can still make can be satisfied by the currently available resources plus the resources held by all P_j, with $j < i$.

A safe state is not a deadlocked state. Conversely, a deadlocked state is unsafe state. Not all unsafe states are deadlocks, however an unsafe state may lead to a deadlock. As long as the state is safe, the operating system can avoid unsafe states. In an unsafe state, the operating system cannot prevent processes from requesting resources in such a way that a deadlock occurs. The behavior of the processes controls unsafe states.

• Safety algorithm :-

We can now present the algorithm for finding out whether or not a system is in a safe state. This algorithm can be described as follows;

1. Let work and finish be vectors of length m and n respectively. Initialize
   Work = Available and Finish[i] = false for $i = 0, 1, \ldots, n-1$.

2. Find an index i such that both
   a. finish[i] == false
   b. Need $i \leq$ work

   If no such i exists, go to step 4.

3. work = work + Allocation[i]
   finish[i] = true
   go to step 2.
4. If finish[i] == true for all i, then the system is in a safe state.

8) Briefly discuss about different deadlock recovery strategy? (CP - 383)

→

□ Process Termination :-

- Abort all deadlocked processes.
- Abort one process at a time until the deadlock cycle is eliminated.

□ Resource preemption :-

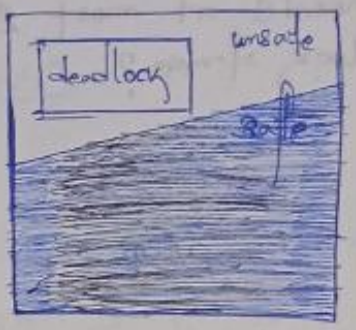
- Selecting a victim.
- Rollback.
- Starvation.

■ Deadlock Recovery :=

1) An algorithm that examines the state of the system to determine whether a deadlock has occurred.
2) An algorithm to recover from the deadlock.

9) what is safe state / safe sequence ?

→ Same as Q.NO (7)



10) what is mutual exclution ?

→ Same as Q.NO (5) 1st point.

11) List two ways we can break the circular wait condition to prevent deadlock, using resource ordering ?

→ same as Q.NO. (4)

12) what do you mean by live lock ?

→ A live lock is similar to a dead lock acept that the state of the processes involved in the live lock, Constantly change with regard to one another, none progressing. This can be avoided by ensureing that one processes takes action.

13) what is bounded waiting ?

→ [critical section chapter] Q.NO - (4) 3rd point.

14) Consider a system consisting of 4 resources of the same type that are shared by 3 processes each of which needs at most 2 resources. show that system is deadlock free? (COPY)

The one resource type is ~~one~~ allocated money that can be allocated to any 3 resources after getting 2 resources the process will terminate & return back 2 resources. Now the Minimum available resources quite large ~~sum~~ that can be ~~available~~ allocated ~~to any~~ rest of the 2 processes. In this way we allocate the resources to make the system deadlock free.

15) what do you mean by preemptive resource?

→ Same as Q.No. 5 2nd point.

↓

(circular wait)

Disk scheduling

1) Define seek time, rotational latency time, rotational delay, sector, track, cylinder.

→

A read write head "flies" just above each surface of every platter. Each disk platter has a flat circular shape like a CD. The heads are attached to a disk arm. That moves all the heads as a unit. The surface of a platter is logically divided into circular tracks, which are sub divided it into sectors. The set of tracks that are at one arm position makes up a cylinder. There may be thousands of cylinders in a disk drive, and each track may contain hundred of sectors. The storage capacity of a common disk drives is measured in gigabytes.

when the disk is in use a drive motor spins it at high speed most drives rotate 60-250 per second. specified in terms of rotation per minute (RPM). The transfer rate is a rate at which data flow between the drive and the computer. The positioning time or random access time consist of two parts. The time necessary to move the disk arm to the desire cylinder, called the seek time. (seek time)

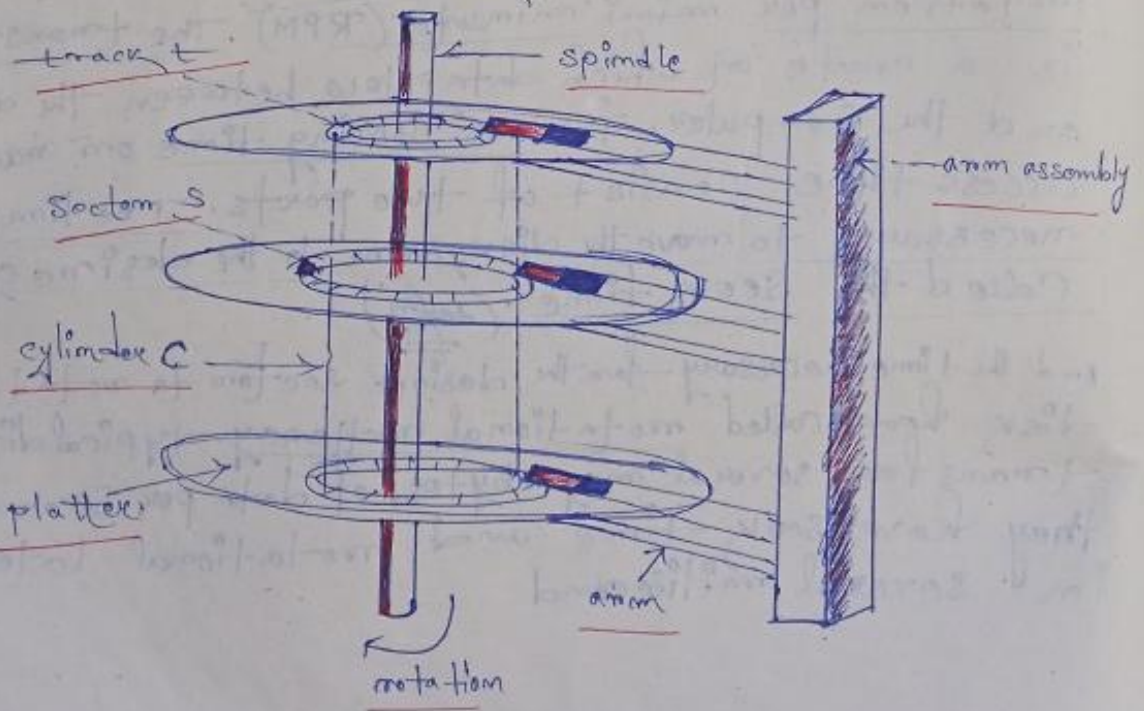
And the time necessary for the desire sector to rotate to the disk head called rotational latency. Typical disk can transfer several mega bytes, of data per second and they have seek time and rotational latency in several milisecond

A disk drive is attracted by a set of wire called an I/O Bus. Several kinds of buses are available including advance technology attachment (ATA). several ATA (SATA), Electronic (eSATA) universal Serial bus (USB) and fibre channel (FC). The data transfers on a bus carried out by special eletronic processors called controllers.

The host controller is the controller at the ~~computer~~ computer end of the bus. A disk controller is build in each disk drive.

Following are the different disk scheduling algorithm —

- FIFO
- SSTF
- SCAN
- LOOK
- C- SCAN
- C- LOOK.



FILE Management

1) Describe swapping and overlaying?

→ In general Computing sence means the process of transferring a block on program code on other data into main memory, replacing what is already stored it is a programming method that allows programmers to be larger than Computers main memory.

swapping :— swapping is a mechanism in which a process can be swapped temporally out of main memory to secondary storage and that memory available to other processes.

2) Explain Indexed allocation and space memory management of files under linked allocation.

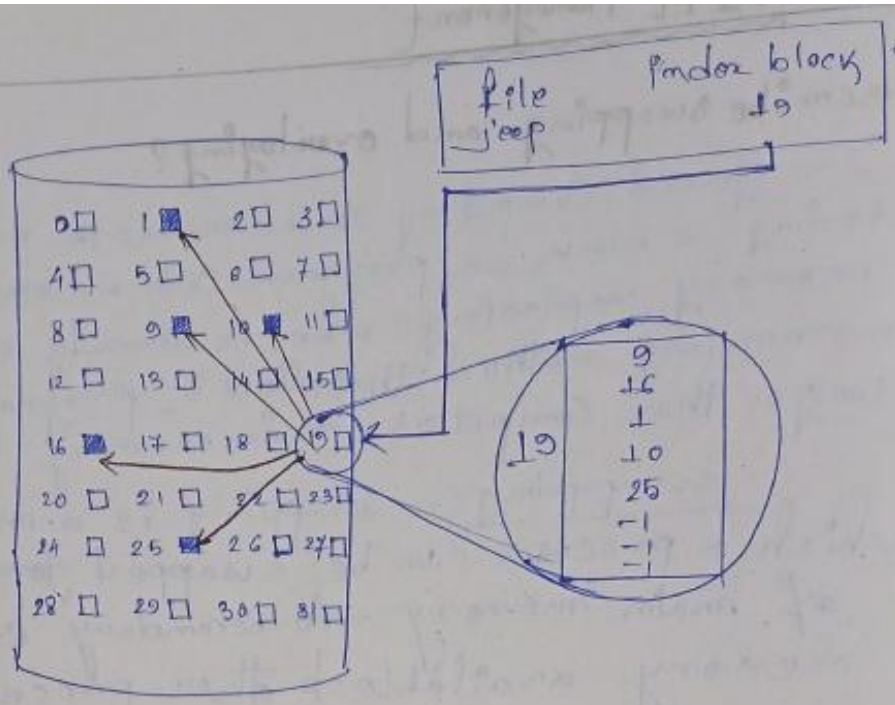
→

■ Indexed Allocation :—

(1) Each file has it's own index block, which is an array of disk-block addresses. The i^{th} entry in the index block points to the i^{th} block of the file. The directory space.

(2) Indexed allocation does suffer from wasted space, however. The pointer overhead of the index block is generally greater than the pointer overhead of linked allocation. Consider a common case in which we have a file of only one or two blocks. with linked allocation, we lose the space of only one pointer per block.

(3)



Free-space management :— (2)

Since disk space is limited, we need to reuse the space from deleted files for new files, if possible (write-once optical disks allow only one write to any given sector, and thus reuse is not physically possible.) To keep track of free disk space, the system maintains a free-space list.

• Bit vector : (2)

Frequently, the free-space list is implemented as a bitmap or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

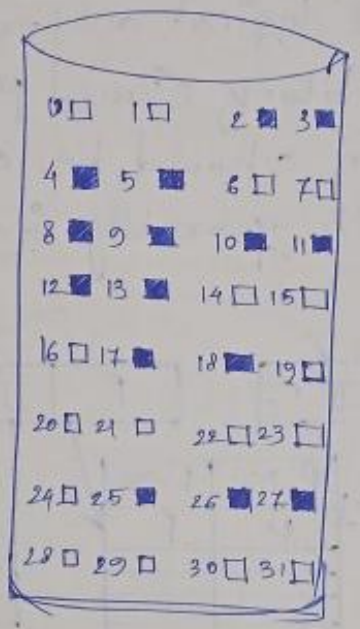
For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, and 27 are free and the rest of the blocks are allocated. The free-space bit map would be,

001111001111100011000000111000000.....

0 → Free to allocate.
1 → Full to allocate.

- Linked List :—

Another approach to free space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory. This first block contains a pointer to the next free disk block, and so on.



3) what is file? Explain Indexed Sequential access.?

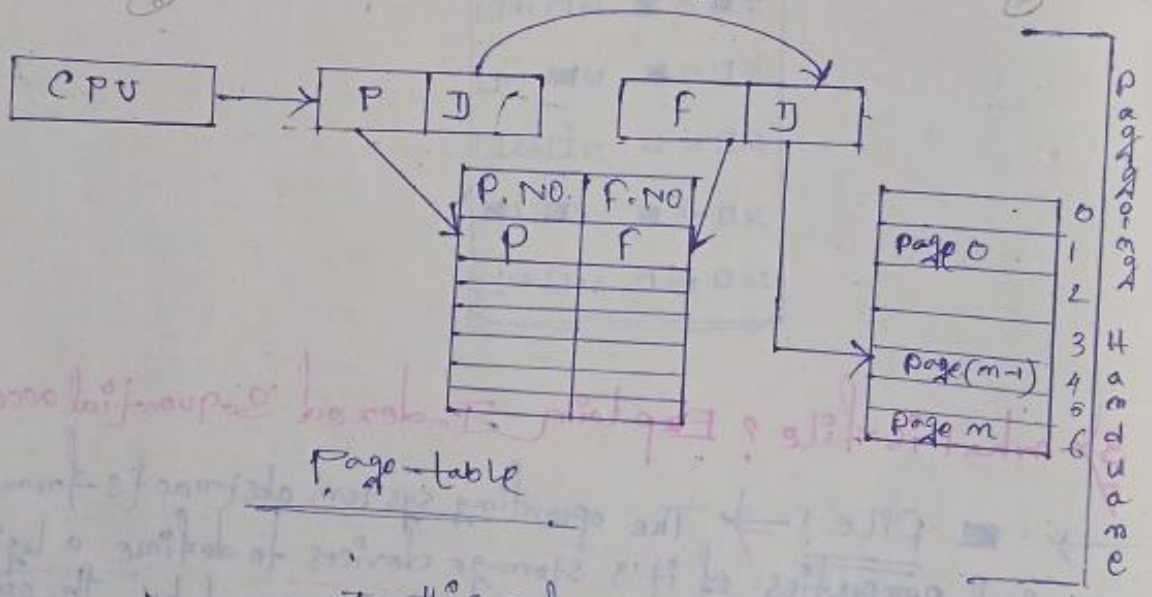
→ ■ File :→ The operating system abstracts from the physical properties of it's storage devices to define a logical storage unit, the file. Files are mapped by the operating system onto physical devices. These storage devices are usually nonvolatile, so the contents are persistent between system reboots.

■ Index Sequential access (Same as Q.No. 2)

Memory Management

1) Explain paging memory management scheme (paging hardware)

→ paging is an efficient memory management scheme, it is not a contigious memory method. In the paging the process are devided into small parts, these is load in the (L^2) main memory. The basic idea of paging is that the physical memory is devided in the fixed size block called frame. The logical address space devided into fixed size block called pages. page size and frame size should be equcal which depends on the OS generally it is 4KB



Page table

In this scheme OS maintain a data structure called page middle. It is use for mapping purpose. The page table specified some useful information, it tells which frames are allocated, which frames are available and how many total pages are there. page table consist of 2 fields ——
(1) Page no. (2) frame no.

Every address generated by the cpu has 2 parts — (1) page no. (2) page offset on displacement. Page no. is the index in the page table. The logical address space i.e. cpu generated address devided into pages. Each page should have a page no. (P) and displacement on offset (D). The pages are loaded into the available free frames (f) into the physical memory.

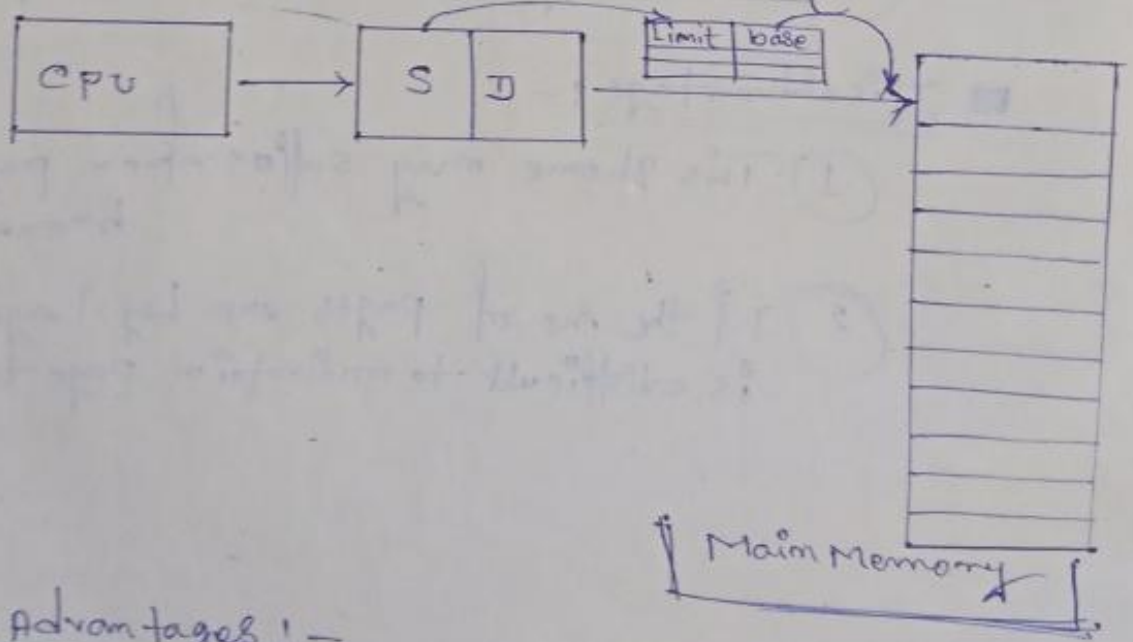
- Advantage :-
  1. It support time shearing system.
  2. It does not affected from external fragmentation.
  3. It supports virtual memory.
  4. shearing of common code is possible.
- Disadvantage :-
  1. This sheme may suffer from page break.
  2. If the no. of pages are tag large, it is difficult to maintain page table.

2) Explain segment memory management scheme?

→ **Segmentation scheme :-**

A segment can be define as a logical grouping of instructions such as a subroutine, an array, a data area etc. every program (Job) is a collection of these segment. segment is a technique for managing these segments.

Each segment has a name and length. The address of the segment specifies both segment name and the offset within the segment. The OS search the entire main memory for free space, to load a segment. This mapping is done by a segment table. Segment table is a table which has a entry for base and limit.



- **Advantages :-**

(1) Eleminate fragmentation by moving segment around. fragmented main memory space can be combined into a single free area.

(2) support virtual memory.
(3) Allow dynamically growing of segment.
(4) Facility for dynamic loading and linking.

3) write the different between paging and segment?

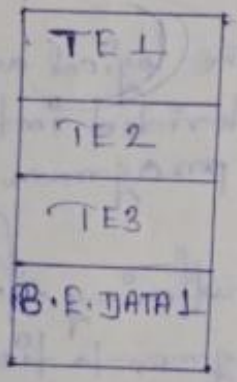
→

Paging	Segment
(1) Main memory partition into frame and blocks.	(1) main memory partition into segmentations.
(2) The logical address space is devided into pages by Compiler or MMU (Memory management unit)	(2) The logical address space is devided into specifies by programm.
(3) suffering from internal fragmentation and Page breake.	(3) suffering from external fragmentation.
(4) OS maintains a free frame list and we need not to search for free frames.	(4) OS maintains the particulars of available memory.
(5) OS maintain the page map table for mapping between page and frame.	(5) OS maintains a segment map table for mapping purpose.
(6) scheme does not support the user view of memory.	(6) It's supports user view of memory.
(7) processor ~~uses~~ uses the page no. and displacement to Calculate absolute address (P.I)	(7) processor uses the segment no. and displacement to Calculate. absolute address (S.I)
(8) Multi level paging is possible.	(8) Multi level segment also possible but not use.

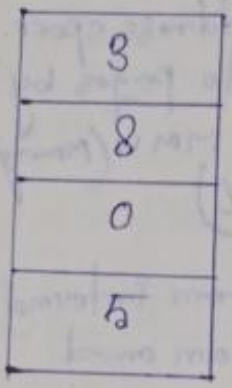
1) what is shared pages?

→ In multiprogramming it is possible to share the common code by no. of processes at a time instead of maintaining the no. of copies of the same code. The logical address is devided into pages. This pages can be sheared by the no. of processes at a time. The page which are sheared by the no. of processes is said to be sheared pages.

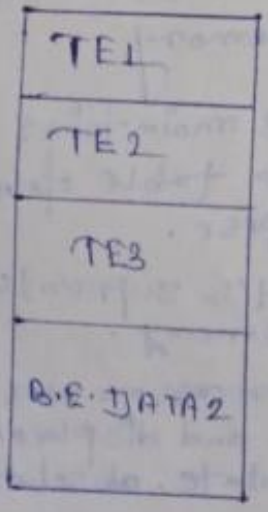
Memory utilization



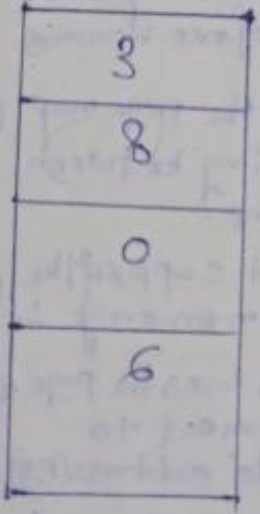
process P1



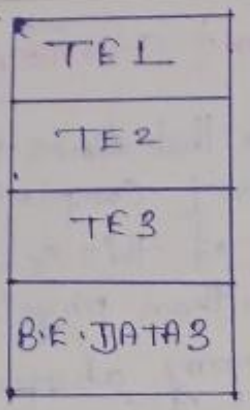
page table for P1



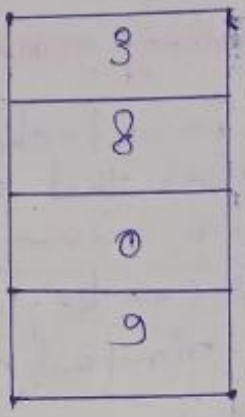
Process P2



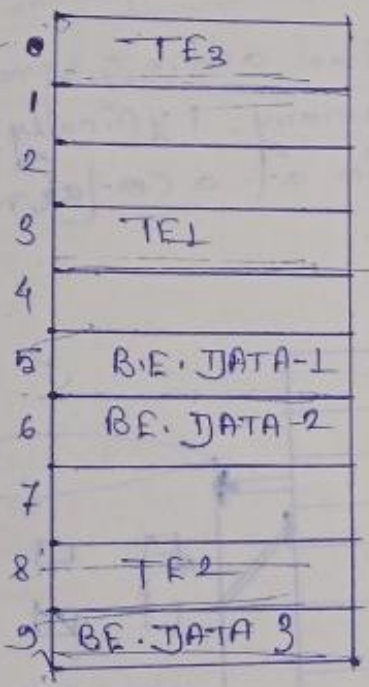
page table for P2



Process of P3



Page table of P3

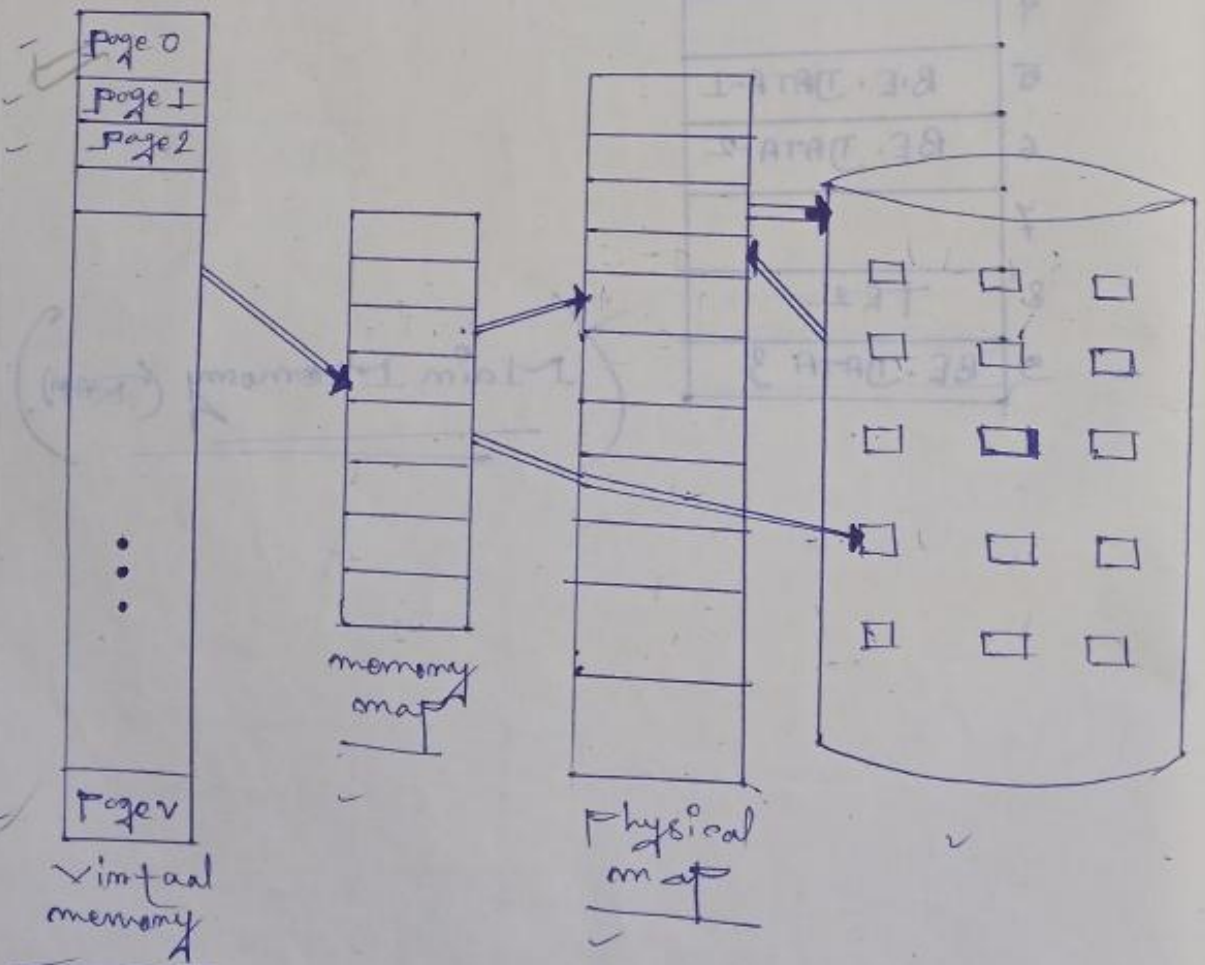


(Main Memory (RAM))

5) Explain virtual memory management scheme?

→ Virtual memory is a technique that allows the execution of processes that are not completely in memory. One major advantage of this scheme is that programs can be larger than physical memory. Further, virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory as viewed by the user from physical memory.

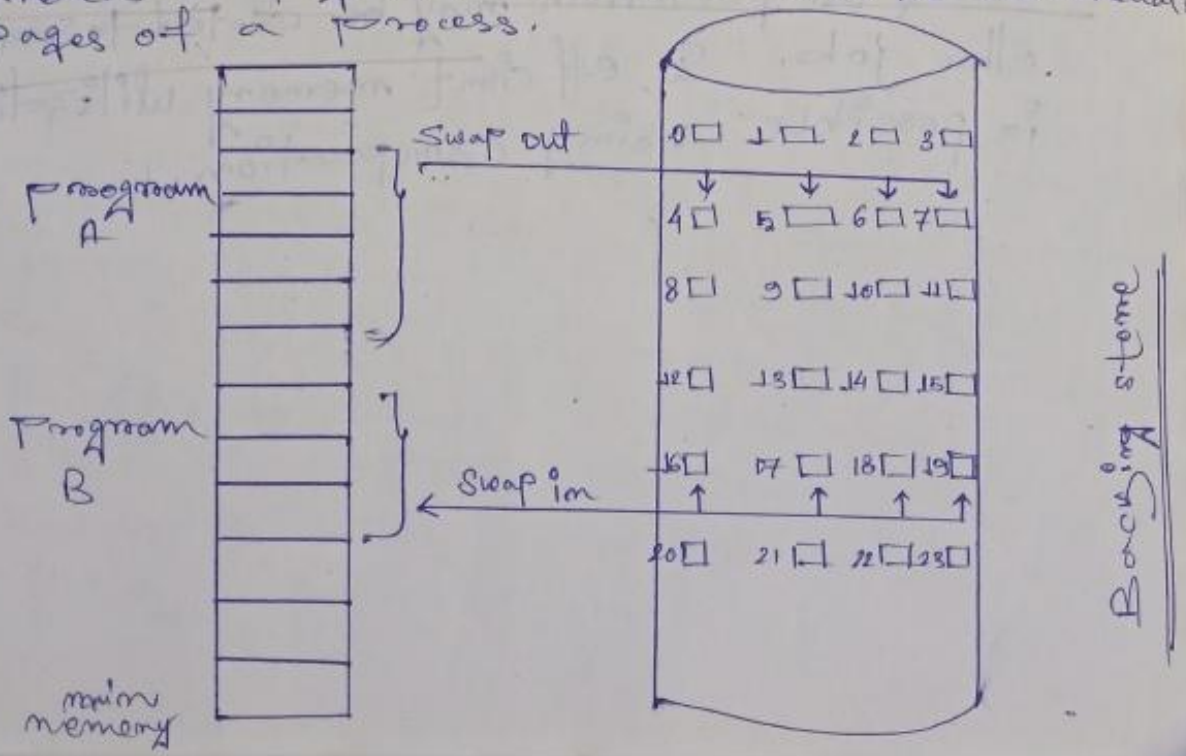
The virtual address space of a process of is stored in memory. Typically, this view is that a process begin at a certain logical address.



6) Explain demand paging memory management scheme?

→ Suppose a program starts with a list of available options from which the user is to select. Loading the entire program into memory results in loading the execution code for all options, regardless of whether or not an option is ultimately selected by the user. An alternative strategy is to load pages only as they are needed. This technique is known as demand paging and is commonly used in virtual memory systems.

Lazy swapper :- A lazy swapper never swaps a page into memory unless that page will be needed. In the context of a demand paging system, use of the term "swapper" is technically incorrect. A swapper manipulates entire processes, whereas a pager is concerned with the individual pages of a process.



7) what is thrashing?

→ In fact, look at any process that does not have "enough" frames. If the process does not have the number of frames it needs to support its pages in active use, it will quickly page fault. At this point it must replace some page. However, since all its pages are in active use, it must replace a page that will be needed again right away. Consequently, it quickly page faults again and again, replacing pages that it must bring back in immediately.

short term scheduler

8) what is compaction? How it is useful in contiguous memory allocation method?

→ Compaction (defragmentation) :→

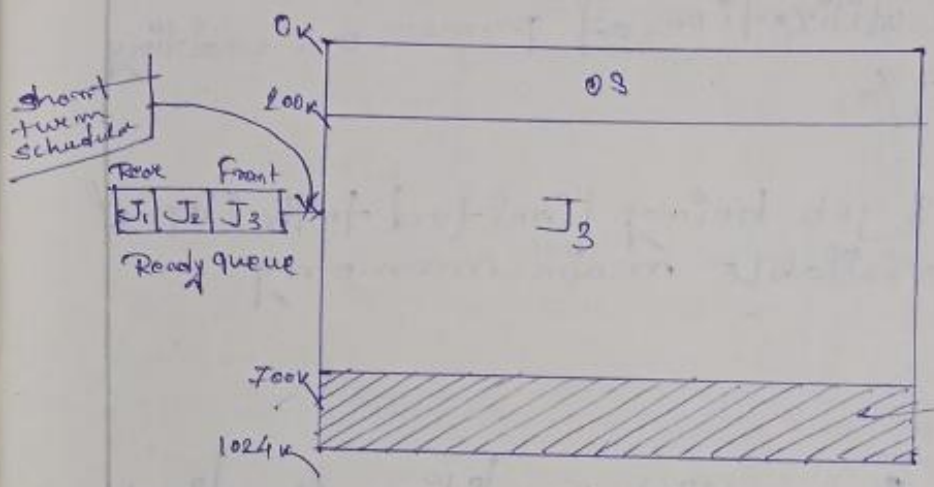
Compaction is a technique of collecting free space together in one block, this block or partition may be alloted to some other job. So, efficient memory utilization is possible using compaction.

Q) Discuss different memory allocation methods?

→ There are mainly 2 type of memory allocation method are present ——

(1) Single partition allocation.
(2) Multiple partition allocation.

(1) Single partition allocation :→



• Single partition allocation

In this memory allocation method the OS resides in the low memory and the maintaining memory is treated as a single partition. This single partition is available for user space. only one job can be loaded in these user space. The short term on the CPU schedular select a job from the ready queue, for execution. The despatcher loads the job into the main memory & connect the CPU to that job. The main memory consisting of only one process a time because the user space treated as a single partition.

• Advantage :-

① simplicity is the main advantage it does not require greate ~~or~~ expertise to understand or use such system.

• Disadvantage :-

① The main disadvantage of these scheme is that memory is not utilize fully.

② Poor utilization of process on waiting for I/o.

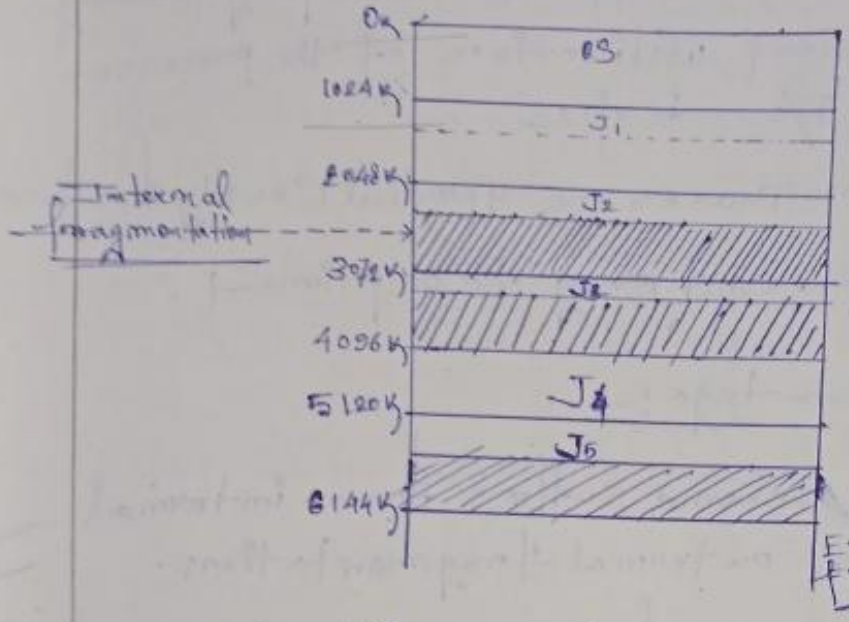
③ users job being limited to the size of the available main memory.

② Multiple partition allocation :-

This scheme can be devided into 3 sub schemes —

(i) Fixed equal multiple partition.
(ii) Fixed variable multiple partition.
(iii) Fixed dynamic multiple partition.

(i) fixed equal multiple partition :-



Job	size
J_1	450KB
J_2	1000KB
J_3	1024KB
J_4	1500KB
J_5	700KB

In this MMS (Memory management scheme) the OS occupies the low memory and the remaining memory is available for user space. This remaining memory is devided into fixed partition. The partition is depend on OS.

Internal and external fragmentation :-

When job1 loading into partition1 (1024-450) = 574KB is wasted. This wastage memory is said to be internal fragmentation. So, memory wastage within a partition is known as internal fragmentation.

The size of job4 is grater than all the partition so if there is any partition which not capable of allocating any job then this partition gets wastage. This wastage partition is known as external fragmentation.

• Advantage :-

(1) This scheme support multiprogramming.
(2) Efficient utilization of the processor and I/O devices.
(3) It requires no special Costly hardware.
(4) simple and easy to compliment.

• Disadvantage :-

(1) This scheme suffers from internal as well as external fragmentation.
(2) The single free area may not be not enough for a partition.
(3) It does not require more memory and single partition method.
(4) A job partition size is limited to the size of the physical memory.

(ii) Fixed variable partition scheme :→

In this scheme the user spaces devided into a number of partition but the partition size are of different lengths. The OS skips a table indecating which partition of memory are available. And which are occupaied. when a process arrived and need memory we search for a partition which is large enough for this process.

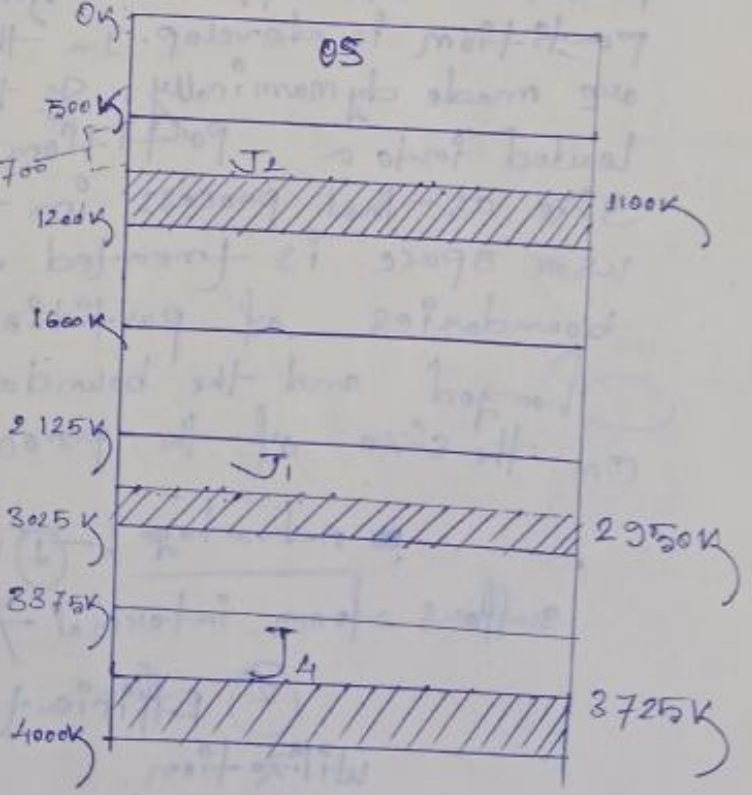
• First fit :- Allocate the partition be. large enough memory to a job. Searching can start either from low or high memory. we can stop searching as soon as we find free partition which is large enough.

• Best fit :- Allocate the smallest partition which is be enough for a job to be store. That means select the partition having least internal fragmentation.

• Worst fit :- Search the entire partition and select the partition which is largest of all available partition. That means select the fragment having big internal fragmentation.

Job	size	arrival time
J_1	825KB	10 m.s
J_2	600KB	5 m.s
J_3	1200KB	20 m.s
J_4	450KB	30. m.s
J_5	650KB	15 m.s

Partition	Size
P_1	700KB
P_2	400 KB
P_3	525 KB
P_4	900 KB
P_5	350KB
P_6	625 KB



■ Advantages :-

(1) Support multiprogramming.
(2) Efficient processor and memory utilization.
(3) simple and easy ~~easy~~ in implement.

■ Disadvantage :-

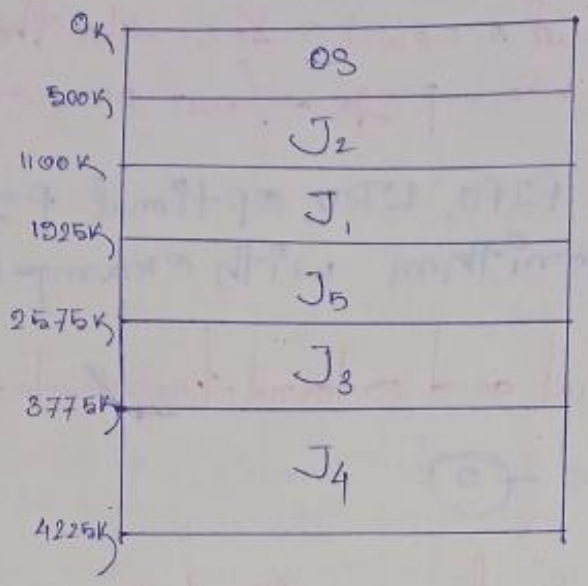
(1) suffers from internal or external fragmentation.
(2) Large external fragmentation is possible.

■ Dynamic partition :-

To eliminate some of the problem with fixed partition and approach known as dynamic partition is develop. In this method the partition are made dynamically. So that each process is loaded into a partition of exactly same size as that process in this scheme the entire user space is treated as bighole. The boundaries of partition are dynamically changed and the boundaries are dependent on the size of the process.

■ Advantage :- (1) This scheme does not suffers from internal fragmentation.
(2) Efficient memory and processor utilization.

■ Disadvantage :- External fragmentation



10) what is first fit, Best fit, and worst fit ?

→ Same as Q. NO. (5).

11) Define the term page, frame, page fault, page table, swap in (roll in), swap out (roll out). ?

→ ■ page, frame, page table :- Same as Q. NO. (6)

■ page fault :- Same as Q. NO. (7)

■ Swap in (roll in) :- when a page is requested by CPU and it is not in main memory then it should be bring from backing store. This is called swap in (Roll in).

■ swapout (roll out) :- when a page is not needed and need to replace then it should be bring back to the backing store ~~from backing store~~. This is called ~~swap~~ Swap out (Rollout).

12) How page-faud occours? Describe the action taken by os when page-fault occours.

→ Describe FIFO, LRU, optimal page replacement algorithm with example.

13) What is internal and external fragmentation?

→ Same as Q.NO (9)

14) What is the role of MMU? (How logical address converted to physical address).

→ The run time mapping from virtual to physical addresses is done by a hardware device called the MMU (memory-management unit). We can choose from many different methods to accomplish such mapping; ~~as we discuss in~~ We illustrate this mapping with a simple MMU scheme that is a generalization of the base-register scheme. The base register is now called a relocation register.

- The Compile-time and load time address-binding methods generate identical logical and physical addresses. However, the execution-time address-binding scheme results in differing logical and physical addresses. In this case, we usually refer to the logical address as a virtual address. We use logical address and virtual address ~~to mean~~ interchangeably in this text. The set of all logical addresses generated by a program is a logical address space. The set of all physical addresses corresponding to these logical address is a physical address space.

15) Define physical and logical address?

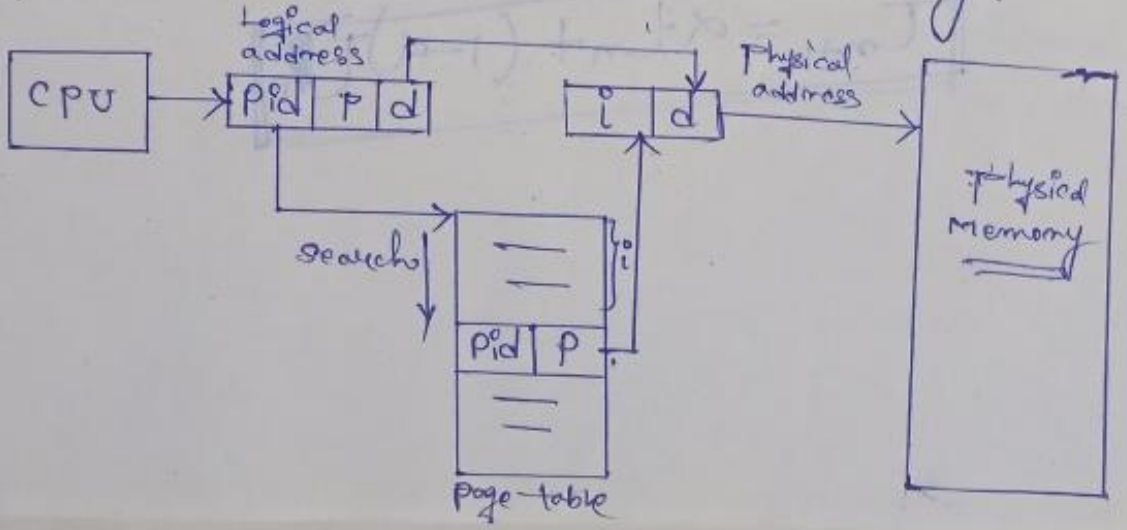
→ Same as Q.No — (14).

16) Discuss the inverted page table mechanism in brief?

→ **Inverted page Table :-**

one of the drawbacks of this method is that each table may consist of millions of entries. These tables may consume large amount of physical memory just to keep track of how other physical memory is being used.

To solve this problem, we can use an inverted page table. An inverted page table has one entry for each real page of memory. Each entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns the page. Thus only one page table is in the system, and it has only one entry for each page of physical memory. Invert page tables often require that an address space identifier be stored in each entry of the page table; since the table usually contains several different address spaces mapping physical memory.



17) what is Hit and Miss ratio?

18) what is exponential averaging? How is it calculated?

→ The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts. We can define the exponential average with the following formula. Let t_n be the length of the nth CPU burst, and let τ_{n+1} be our predicted value for the next CPU burst. Then, for α, $0 \le \alpha \le 1$, define,

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

1) Why are page sizes always in power of 2?

→ We use a binary bits combination of 2 binary bits can be used for identifing memory segment (address). using 2 bit combination, we can represent $2^2 = 4$ different location. To represent K different location, we need 2^K K bits. So, memory size should be 2^K so, that each and every location identify using binary bit combination.

2) What is Belady's anomaly?

→

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Notice, that the number of faults for four frames is greater than the number of faults for three frames. This most unexpected result is known as Belady's anomaly. For some page-replacement algorithms the page-fault rate may increase as the number of allocated frames increases.

21) why are segment and paging sometime Combined into one scheme?

→ Segmentation and paging often combine in order to improve upon each other segmented paging is helpful when the page table becomes very large a large Congious section of the page table ~~into~~ that is unused can be collapsed into a ~~sig~~ single ~~segon~~ segment table with a page table address of 0.

The page segmentation handles the case of having very long segment that require a lot of time for ~~allocating~~ allocation by paging the segments. we reduce wasted memory due to external fragmentation as well as simplify the allocation.

22) Explain FIFO, LRU, OPTIMAL page replacement algorithm?

→ Explain FIFO, LRU, OPTIMAL with a example.

■ FIFO Page replacement :⇒

The simplest page-replacement algorithm is a first in, first out (FIFO) algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. Notice that it is not strictly necessary to record the time when a page is brought in. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When a page is brought into memory we insert it at the tail of the ~~que~~ queue. A

• Give an mathematical example (COPY) •

■ Optimal Page replacement :⇒

one result of the discovery of Belady's anomaly was the search for an optimal page-replacement algorithm — the algorithm that has the lowest page-fault rate of all algorithms and will never suffer from Belady's anomaly. Such an algorithm does exist and has been called OPT or MIN.

• Give an mathematical example (COPY)

■ LRU page replacement :⇒

If the optimal algorithm is not feasible, perhaps an approximation of the optimal algorithm is possible. The key distinction between the FIFO and OPT algorithms is that the FIFO algorithm uses the time when a page was brought into memory, whereas the OPT algorithm uses the time when a page is to be used. If we use the recent past as an

approximation of the near future, then we ca replace the page that has not been used for the longest period of time. This appro is the least recently used (LRU) algorith

23) what is purpose of valid(v) and invalid(i) bit for protection purpose in memory management ?

→ One additional bit is generally attached to each entry in the page table : a valid-invalid bit. When this bit is set to valid the associated page is in the process's logical address space and is thus a legal page. When th bit is set to invalid, the page is not in the process's logical address space. Illegal addresses are trapped by use of the valid-invalid bit.

24) why limit register is called fencing register

→ limit register is also called fencing register because it denotes the limit of the available memo So, whenever we are getting some logical address we have to check within the limit or not if it less than the limit it will be accepted otherwise re

* CPU Scheduling *

1) Define the term cpu scheduling / Job scheduling, throughput, Response time, turned around time.

→ ■ cpu scheduling :- scheduling of this kind is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary comp resources. Thus, its scheduling is central to operating-system design.

■ Throughput :- If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput

■ Response time :- The another measure is the time from the ~~subm~~ submission of a request until the first response in produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response.

■ Turn around time :- From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time.

■ waiting time! — The CPU-scheduling-algorithm does not affect the amount of time during which a process executes on does I/O. It affects only the amount of time that a process spends waiting in the ready queue. waiting time is the sum of the periods spent waiting in the ready queue.

2) what are the criteria of cpu scheduling?

→ same as Q. NO - (1)

3) Compare between preemtive and non preemptive scheduling?

→ Under nonpreemtive scheduling once the CPU has been allocated to a process the process keeps CPU until it releases the CPU either by terminating on by switching to the waiting state.

Cooperative scheduling is the only method that can be used on certain hardware platforms, because it does not require the special hardware needed for preemptive scheduling. unfortunately preemptive scheduling can result in race conditions when data are shared among several processes.

4) Explain fcfs, SJF, preemptive PRIORITY, non preemptive PRIORITY, SSTF/SSTN, ROUND ROBIN (RR) scheduling ?

→ Explain the scheduling algorithm with ~~eac~~ example.

5) Explain the difference in the degree to which the following scheduling algorithms discriminates in favour of processes with short time (i) fcfs (ii) Round Robin (RR) (iii) Multilevel feedback queue.

→ • fcfs (first come first serve) :- fcfs executed those process which arrived first i.e. the process arrived first will be executed first. So, a process having shorted time will not be benifited in fcfs scheduling.

• Round Robin :- Here, each process will get a short period of time (time slice) to executed in CPU. Because of it's ~~non~~ round nature all process will get a fair chance to be executed in CPU. So, some degree of advantage is there ~~for~~ for process will short time.

• mutilevel feedback queue :-

Here, mutlevel queues are maintain with different quantum and each queue follows round robin scheduling aspect the last level queue which follows FCFS schuduling. Here, process with short time may get executed faster because of round robin schuduling.

6) Explain the effect of time quantum in the performance of RR scheduling.

→ In RR scheduling if time Quantum (TQ) very large then round robin schuduling can be look like a FCFS schuduling. If time slice very low then we have to preempt the processes most frequently that will create extra hazard.

7) what is starvation and aging ?

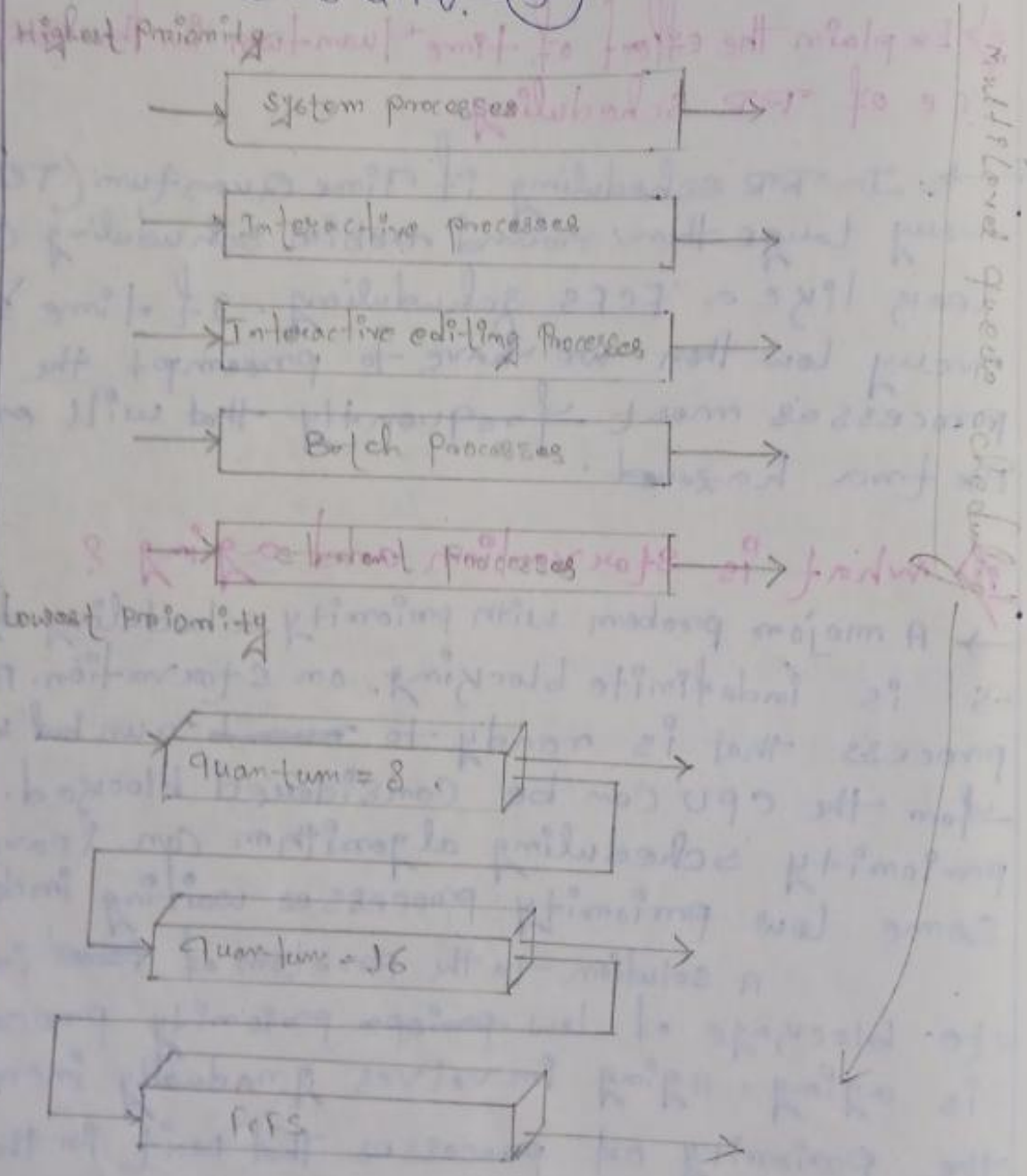
→ A major problem with priority scheduling algorithms is indefinite blocking, or starvation. A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low priority processes waiting indefinitely. A solution to the problem of indefinite blockage of low priority processes is aging. Aging involves gradually increasing the priority of processes that wait in the system for a long time.

8) Explain multilevel Round Robin scheduling?

→ Same as Q.NO. (5)

9) Explain multilevel feedback Queue Scheduling? what is the drawbacks?

→ Same as Q.NO. (5)



10) what do you mean by foreground and background process?

→ Another class of scheduling algorithms has been created for situations in which processes are easily classified into different groups. For example, a common division is made between foreground processes and background processes. These two types of processes have different response-time requirements and and so may have different scheduling needs. In addition, foreground processes may have priority over background processes.

11) "Fairness is one of the objectives of CPU Scheduling" — Critically Comment on this.

→ Discuss about FCFS, SJF, RR, priority Scheduling algorithm with a example.

Process Management

1) what is process scheduling / thread scheduling ?

2) Discuss different thread models

(or)

3) Compare between process and thread.

(or)

4) What is the different between Blocked and sleep state in thread life cycle?

(or)

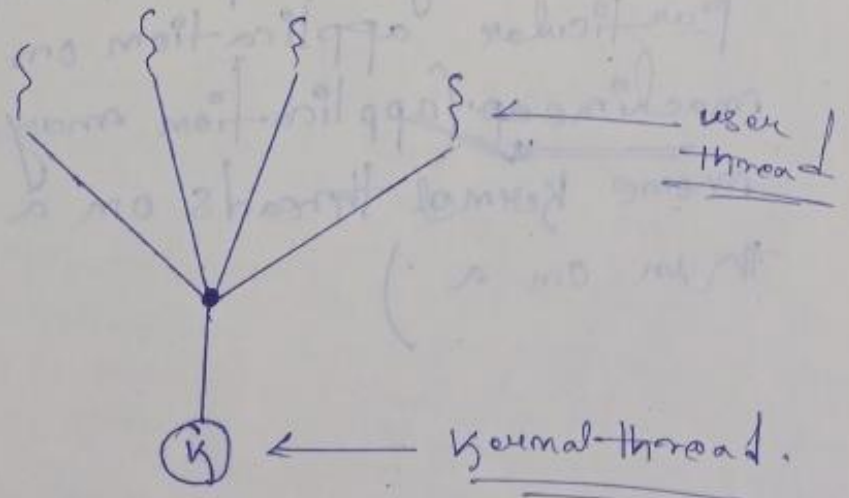
5) "Thread improves the responsiveness of a system" — critically Comment on this.

(or)

6) Discuss about threading?

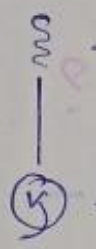
→ • Many-to-one Model :—

The many-to one model maps many user-level threads to one kernel thread. Thread management is done by the thread library in user space, so it is efficient. However, the entire process will block if a thread makes a blocking system call.

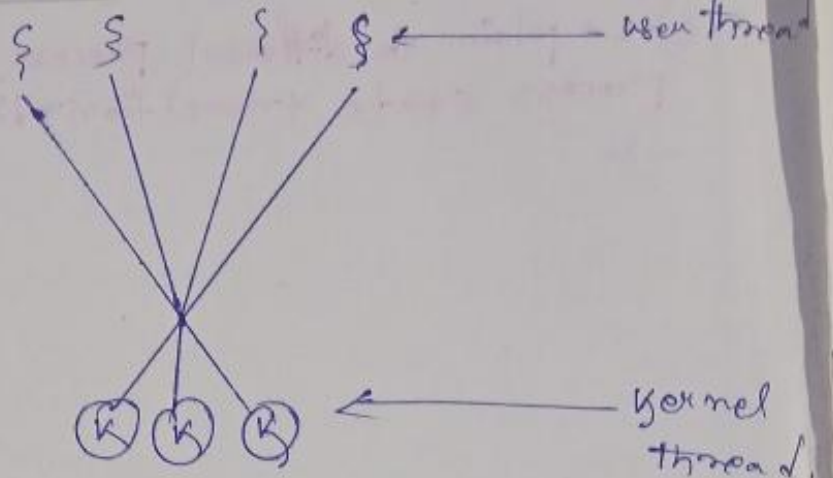


- One-to-one model :-

The one-to-one model maps each user thread to a kernel thread. It provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call. It also allows multiple threads to run in parallel on multiprocessors.



- Many-to-Many model :- The many-to-many model multiplexes many user level thread to a smaller or equal number of kernel threads. The number of kernel threads may be specific to either a particular application or a particular machine (application may be allocated more kernel threads on a multiprocessor than on a)

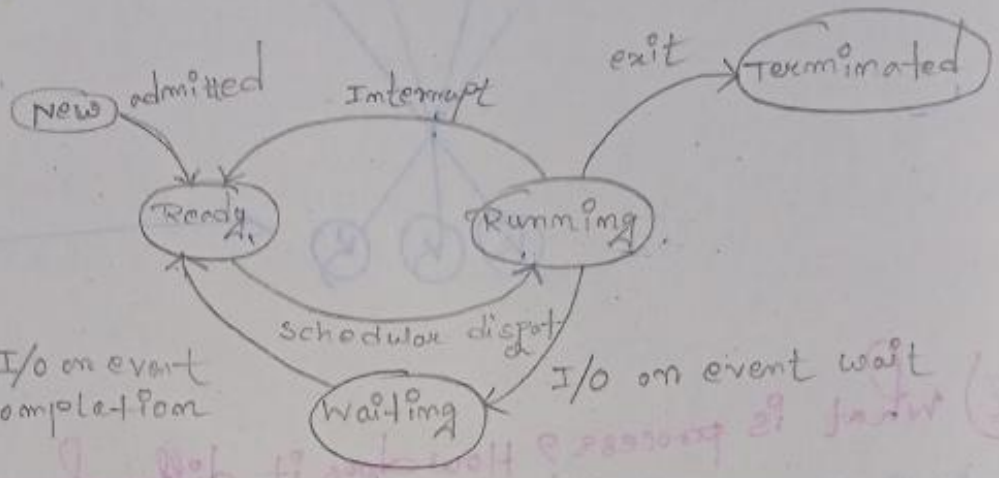


7) What is process? How does it differ from program?

→ A process is a program in execution. A process is more than the program code which is sometimes known as the text section. It also includes the current activity as represented by the value of program counter and the contents of the processor's registers.

process	program
(i) A process is a active program of a portion	(i) collection of instruction make a program.
(ii) A process may have different states like new, ready, waiting, running, and terminated	(ii) A program do not have such states.
(iii) A process may ~~fath~~ farther devided into threads.	(iii) A ~~g~~ program may be devided into several function.
(iv) If multiple cpu runs several processes then it is called multiple processor statement.	(iv) If a single cpu runs different program then ~~its~~ this is called program statement.

8) Explain the different process state using suitable process state transition diagram.



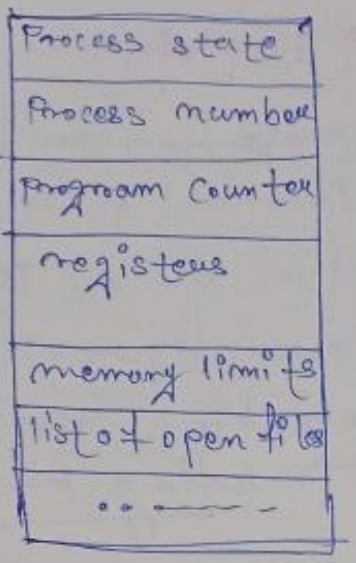
→ As a process executes, it changes state. The state of a process is defined at in part by the current activity of that process. A process may be in one of the following —

- New – The process is being created.
- Running – Instruction are being executed.
- Waiting – The process is waiting for some event to occur (such as an I/O completion on reception of a signal)
- Ready – The process is waiting to be assigned to a processor.
- Terminated – The process has finished execution.

9) what is PCB/TCB ? Explain.

→ Each process is represented in the operating system by a process control block (PCB) - also called a task control block (TCB). A PCB is shown in the lower figure. It contains many pieces of information associated with a specific process, including these:-

- process state :- The state may be new, ready, running, waiting, halted and so on.
- program counter :- The counter indicates the address of the next instruction to be executed for this process.
- CPU register :- The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued.
- CPU-scheduling information :- This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- Memory-management information :- This information may include such item as the value of the base and limit registers and the page tables or the segment tables, depending on the memory system used by the OS.



Process control block (PCB)

10) What is PJB ? Explain.

→ PJB is same as PCB (process Control Block). [See the Q. NO. 9]

11) What is context switching?

→ switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as context switching

12) How do I/O bound and CPU bound program differ?

→ It is important that the long-term scheduler make a careful selection. In general most processes can be described as either I/O bound or CPU bound. An I/O-bound process is one that spends more of its time doing I/O than it spends doing computations. A CPU-bound process, in contrast, generates I/O requests infrequently, using more of it's time doing computations.

13) what are the different between user level thread and Kernel supported thread? under what circumstances is one type is better than the other?

→ For user threads, are by the Kernel, for Kernel threads. user threads are supported above the Kernel and are managed without Kernel support, whereas Kernel threads are supported and managed directly by the operating system.

14) Define long term(job) scheduler, short term (CPU) Scheduler, medium term scheduler. why is it called so?

→ ▨ Long term scheduler :- The long term scheduler or job scheduler, selects processes from this pool and loads them into memory for execution.

▨ short term scheduler :- The short term scheduler or CPU scheduler, selects from among the processes that are ready to execute and allocates the CPU to one of them.

▨ medium term scheduler :- Medium-term scheduler is ~~diagrammed~~ the idea behind a medium-term scheduler is that sometimes it can be advantageous to remove a process from memory and thus reduce the degree of multi-programming.

15) what is the function of dispatcher?

→ • The process could issue an I/O request and then be placed in an I/O queue.

• The process could create a new child process and wait for the child's termination.

• The process could be removed forcibly from the CPU as a result of an interrupt, and be put back in the ready queue.

16) what is JCL (Job Control Language)?

→ Job control language (JCL) is name for scripting languages used on IBM mainframe operating systems & to instruct the system on how to run a batch job or start a subsystem. Mainframe batch operating systems had some form of job control language, whether called that or not; their syntax was totally different from IBM versions, but they usually provided similar capabilities. Interactive signal — include "command language" — command files can be run non-interactively, but these usually do not provide as robust an environment for running unattended jobs as JCL. On some computer systems, the job control language and the interactive command language may be different.

In both DOS and OS JCL the first 'card' must be the job card, which:

- Identifies the job.
- usually provide information to enable the Computer Services department to bill the appropriate user department.
- Defines how the job as a whole is to be run, its priority relative to other jobs in the queue.

17

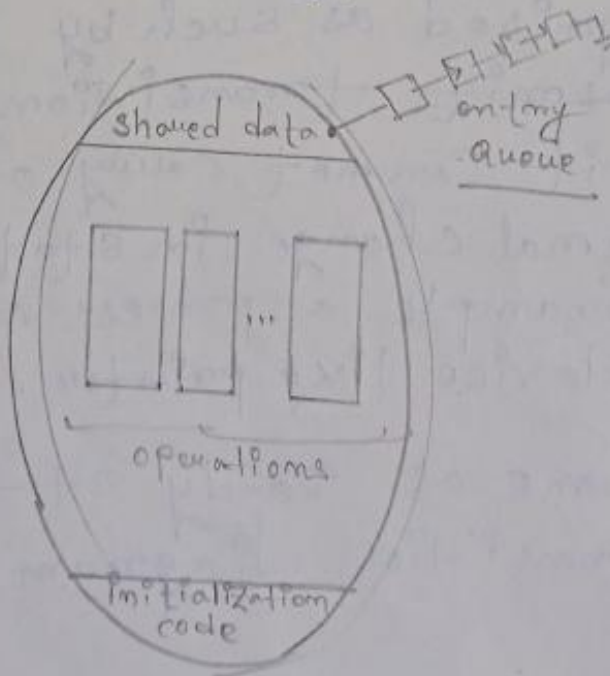
On unix and unix like computer OS a Zombie process is a process that has Complited execution but still has an entry in the process table. It is the process in the terminated state.

Block/suspence:- In a multitasking computer system process may occupy a variety of states these disting state may not be recognised as such by kernel. A process transmit transitions to a block state when it cannot carry on without an external change in state or even occowing. For example a process may wait for I/O device like printer.

- Same as Ready state of process transition diagram.

(18) Explain the use of monitor in thread synchronization ?

→ An abstract data-type — or ADT - encapsulates data with a set of functions to operate on that data that are independent of any specific implementation of the ADT. A monitor type is an ADT that includes a set of programmer-defined operations that are provided with mutual exclusion within the monitor. The monitor type also declares the variables whose values define the state of an instance of that type, along with the bodies of functions that operate on those variables. The representation of a monitor type cannot be used directly by the various processes. Thus a function defined within a monitor can access only those variables declared locally within the monitor and it's formal parameters. Similarly, the local variable of a monitor can be accessed by only the local functions.



10) Multithreading means multiple thread are running at the same time so, we can improve the cpu performance by introducing multithreading Concept.

In case of swapping each process has to swap in the memory when it is not need this procedure is called swap out. If a new process bring into ram using swap in methodlogy. swapping takes huge number of time for doing it's job whereas multithreading is much more efficient than swapping procedure.

10) Multithreading means multiple thread are running at the same time so we can improve the CPU performance by introducting multithreading, Concept.

In case of swapping each process has to swap in the memory when it is not need this procedure is called swap out. If a new process being into ram using Swap in methodology. swapping takes huge number of time for doing it's job whereas multithreading is much more efficient than swapping procedure.

(1) Why does Computer start in Kernel mode when power is first turned on?

(2) What characteristic is common to traps, interrupts and subroutine Call?

(1) Why does Computer start in Kernel mode when power is first turned on?

(2) What characteristic is common to traps, interrupts and subroutine Call?

(3) what is distributed os and network os? what is the difference between them? (Advantage & Disadvantage)

→ A network os is basically that is designated primarily to support work station, personal Computer, older terminal, that are connected on a LAN.

whereas in distributed system the os is spreaded over a collection of independent network communicating, physically seperated Computational note. They handle the job which are by multiple cpu.

(4) (5) (11) same

- list the different functions performed by os?
- what is os? list the different functions performed by multiuser os?

(6)

- "os as resource manager" – Explain. / "operating system is like a manager in an organization – justify.

→ ■ User interface :– Almost all os have a user interface (UI). This interface can take several forms. One is a Common-line interface (CLI).

■ program execution :– The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally.

■ I/O operations :– A running program may require I/O, which may involve a file or an I/O device. For specific devices, special functions may be desired

- File-system manipulation :- The file system is of particular interest. obviously, programs need to read and write files and directories They also need to create and delete them by name, search for given file, and list file information.

- Communication :- There are many circumstances in which one process needs to exchange information with another process. Such communication may occure between processes that are executing on the same computer or between processes that are executing on different computer systems tied together by a computer network.

- Error detection :- The operating system needs to be detecting and correcting errors constantly. Errors may occure in the CPU and memory hardware, in I/O devices.

- Resources allocation :- when there are multiple users or multiple jobs running at the same time resources must be allocated to each of them. The operating system manages many different type of resources.

- Accounting :- we want to keep track of which users how much and what kinds of computer resources.

- Protection & Security :- The owners of the information stored in a multiuser or networked computer system may want to control use of that information.

(6) what is interrupt Control I/O — Explain.

7) Compare multiprogramming OS, multiprocessing os, multitasking os, time sharing os, batch processing os, ~~real~~ real time os.

→ (1) Multiprocessing :⇒ It's refers to processing of multiple processes at same time by multiple CPUs. If one of the cpu gets break down another will take over the load.

(2) Multiprogramming :⇒ multiprogramming keeps several program in main memory at same time execute ~~them~~ them concurrently ~~utilizing~~ utilizing in single cpu.

(3) multitasking :⇒ It has the same meaning of multiprogramming but in a more general have refer to having multiple program processes, task running at the same time.

(4) Time sharing OS :— Time sharing os is a technique which enable many people located at various terminal to used a particular computer system at the same time. Time sharing or multitasking is a logical extention of multiprogramming processors time which is share among multiple user symultaniously is turned as time sharing.

(8) Define deadline with respect to soft realtime

(9) what is spooling (how it is improve efficiency and buffering) ?

→ It is a technique that mainly use in printer where spooler is use to select a print from the pool of printing jobs. This spooling may be perform in FCFS basic that means those printing jobs will be executed first which where appear first.

Exp:— printer spooling.

al time.

(10) what is the different between OS main-frame and personal Computer.

ciency)

be

f

t.

(12) (13) (14) (15) (16) (17) (18) OS Question Answer Sheet.

Dead lock Recovery strategy:-

There are different strategy to recover the dead lock. The two condition are as follows-

(i) Process termination.

(ii) Resource preemption.

Process termination:- The process termination is specifies in two different ways:-

(i) **Abond all the deadlock cycle:-** In this method we close all deadlock processes to make the system deadlock free.

(ii) **Abond one process at a time until the dead lock is eleminate:-** To make a system deadlock free we have terminate the process that involve in dead. After then we terminate the next process until the deadlock is eleminate from the system.

Resource preemption:- The resource preemption is specifies in 3 different ways:-

(i) **Selecting a victim:-** In this method we select the resource or the process who creates the deadlock condition will have to terminate from the system.

(ii) **Rollback:-** If the system is in dead lock state and to eleminate the dead lock condition we have to rollback the condition until the system to be deadlock free. Here rollback means undo the present work.

(iii) Starvation :- In this method when a process creates the dead lock situation we must terminate the process. After then if the same process create the dead lock situation we terminate these process. In this way we terminate the process multiple time so the process doesn't accomplished the given task.

Reader writer problem :⇒

```
Semaphore mutex = 1; // Controls access to the reader Count
Semaphore db = 1; // Controls access to the database
int reader_Count; // The number of reading processes
                     accessing the data.

Reader()
{
    while(TRUE){          // Loop forever
        down(&mutex);     // gain access to reader_Count
        reader_Count = reader_Count + 1; // Increment the
                                            reader_Count
        if(reader_Count == 1)
            down(&db);    // if this is the first process
                             read the database,
                          // a down on db is executed to prevent access
                             to the process.
                          // database by a writing process.
        up(&mutex);       // allow other process to access
                             reader_Count
        read_db();        // read the database.
        down(&mutex);     // gain access to reader_Count
        if(reader_Count == 0)
            up(&db);      // if there are no more process
                             to access the data
        up(&mutex);       // allow other process to acces
                             reader_Count use_data();
                          // use the data read from the database(non-critical
```

```
}
writer()
{
  while (TRUE){   // loop forever
    create_data(); // create data to enter into
                   database (non-critical)
lock down(&db)   //gain access to the database
    write_db();  // write information to
                    the database.
unlock up(&db);  //release exclusive access to the
                                    database.
  }
```

Dining Philosopher..... !

```
#include<pthread.h>
#include<semaphore.h>
#include<stdio.h>
#define N 5
#define THINKING 2
#define HUNGRY 1
#define EATING 0
#define LEFT (phnum+4)%N
#define RIGHT (phnum+1)%N

int state[N];
int phil[N] = {0,1,2,3,4};
sem_t mutex;
sem_t S[N];
void test(int phnum)
{
  if(state[phnum] == HUNGRY
    && state[LEFT] != EATING
    && state[RIGHT] != EATING) {
      // state that eating
    state[phnum] = EATING;
  sleep(2);
printf("philosopher %d takes fork %d and %d\n
        phnum+1, LEFT+1, right+1);
printf("philosopher %d is Eating\n", phnum+1);
```

```
// sem_post(&S[phnum]) has no effect
// during take fork
// used to wake up hungry philosophers
// during put fork
    sem_post(&S[phnum]);
    }
}

// take up chopsticks
void take_fork(int phnum)
{
    sem_wait(&mutex);
    // state that hungry
    state[phnum] = HUNGRY;
printf("Philosopher %d is Hungry\n", phnum+1);
    // eat if neighbours are not eating test(phnum);
    sem_post(&mutex);
    // if unable to eat wait to be signalled
    sem_wait(&S[phnum]);
    sleep(1);
}

// put down chopsticks
```

function call

```
void put_fork(int phnum)
{
  sem_wait(& mutex);
  // state that thinking
  state[phnum] = THINKING;
printf("philosopher %d putting fork %d and %d
                    down\n",
      phnum+1, LEFT+1, right+1);

printf("philosopher %d is thinking\n", phnum+1);
   test(LEFT);
   test(RIGHT);
 sem_post(& mutex);
}
void* philosopher(void * num)
{
  while(1) {
   int * i = num;
    sleep(1);
   take_fork(* i);
    sleep(0);
   put_fork(* i);
  }
}
```

```
int main()
{
  int i;
  pthread_t thread_id[N];
  //initialize the semaphores.
  sem_init(&mutex, 0, 1);
  for(i=0; i<N; i++)
    sem_init(&S[i], 0, 0);
    for(i=0; i<N; i++) {
    // create philosopher processes
      pthread_create(&thread_id[i], NULL
        philosopher, &phil[i]);
    printf("philosopher %d is thinking\n",
                                  i+1);
    }
    for(i=0; i<N; i++)
      pthread_join(thread_id[i], NULL);
    }
```

Bounded Buffer problem (Producer Consumer problem)

The bounded buffer problem present a general structure of the scheme. we assume that the pool consist of n buffers, each capable of holding one item. The mutex semaphore provide mutual exclution for access to the buffer pool and this is initially the value 1. the empty & full semaphore count the number of empty and full buffers. The semaphore empty is initially to the value n the semaphore full is initialize to the value 0.

```
do {
   ...
   // produce an item in next p
   ...
   wait (empty);
   wait (mutex);
   ...
   // add next p to buffer.
   ...
   signal (mutex);
   signal (Full);
} while (TRUE);
```

The structure of the producer process.

```
do {
  wait (full);
  wait (mutex);
    ...
// remove an item from buffer to nextc
    ...
  signal (mutex);
  signal (empty);
    ...
// consume the item in nextc.
    ...
} while (TRUE)
```

The Structure of the Consumer process

Descard problem :⇒

The descard's algorithm first probably correct solution to the critical section problem.

```
main()
{
  int Threads_NO = 1;
   start Threads();
}
Thread 1()
{
  do{
    // Entry section, wait until thread no. is 1
  while(Thread NO == 2)
   // critical section
  // Exit section
  // Give access to the other threads
  Thread NO = 2
  // remainder section
  } while(completed == False)
}
```

```
Thread 2( )
{
    do {
        // Entry section, wait until thread no. is 2
        while (Thread No. == 1);
        // critical Section.
        // Exit section
        // Give access to the other threads
        Thread No = 1
        // remainder section
    } while (compleated == False)
}
```

(1) what is shell in unix?

→ The shell is the command interpreter in an OS it is program that execute other program it provide a computer interface to the unix system. so that user can roun different commands with some input data.

• shell types :- There are two major type of shell.

• see shell

(2) what is shellscript ?

→ It is a list of commands which are listed in the order of execution a good shell script will have comments preceded # sign

(3) what is Kernel in unix ?

→ The computer programs that is allocate the system resources and co-ordinate all the details of the computers internal is called the OS or Kernel. It is the heart of the OS. It interacts the hardware and most of the task like memory management task scheduling and file management.

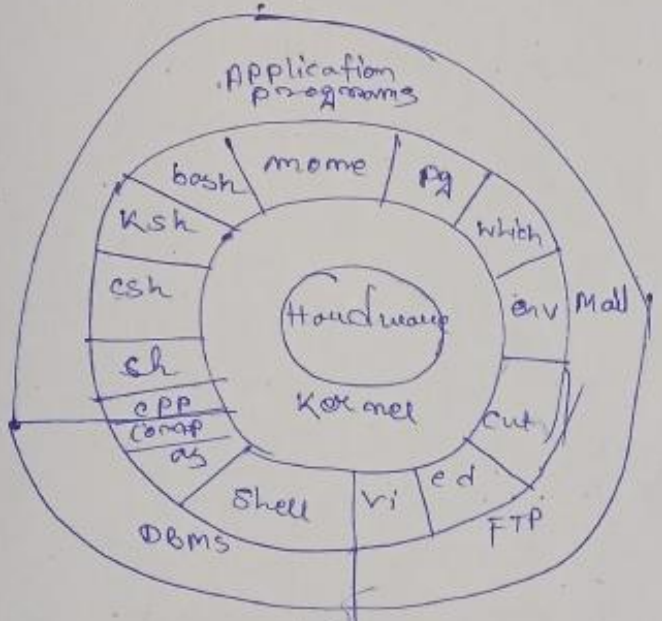
(4) Briefly discuss different file structure in unix ?

→ In unix, there are 3 basic types of files.

• ordinary files :- An ordinary file is a file on the system that content data, text or program instruction.

• Directories :- It's store both special and ordinary files for users familier with windows, unix directories are equivantant to folder.

• special files :- Some special files provide access to hardware such as harddrive, CD, Rom drive, modem etc.



Layer Architecture on Kernel Shell relationship in unix

$16K = 2^4 \cdot K$ ∴ where $K = 1024$

$= 2^4 \cdot 2^{10}$

$= 2^{14}$