SYLLABUS

	TOPIC
1.	**Introduction :** What is an operating system, Simple Batch Systems, Multi- programmed Batch systems, Time- sharing Systems, Personal- computer systems, Personal – Computer Systems, Parallel systems, Distributed Systems, Real - Time Systems. **Memory Management :** Background, Logical versus Physical Address Space, Swapping, Contiguous Allocation, Paging, Segmentation. **Virtual Memory :** Demand Paging, Page Replacement, Page Replacement Algorithms, Performance of Demanding Paging, Allocation of Frames, Thrashing, Other Considerations.
2.	**Processes :** Process Concept, Process Scheduling, Operation on Processes. **CPU Scheduling :** Basic Concepts, Scheduling Criteria, Scheduling Algorithms, Multiple Processor Scheduling. **Process Synchronization :** Background, The Critical- section Problem, Synchronization Hardware. Semaphores, Classical Problems of Synchronization.
3.	**Deadlocks :** System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock.
4.	**Device Management :** Techniques for Device Management, Dedicated Devices, Shared Devices, Virtual Devices; Input or Output Devices, Storage Devices, Buffering, Secondary Storage Structure : Disk Structure, Disk Management, Swap-space Management, Disk Reliability.
5.	**Information Management :** Introduction, A Simple File System, General Model of a File System. Symbolic File System, Basic File System, Access Control Verification, Logical File System, Physical File System. File- system Interface; File Concept, Access Methods, Directory Structure Protection, Consistency Semantics File- System Implementation : File System Structure, Allocation Methods, Free- space Management.

INTRODUCTION / MEMORY MANAGEMENT / VIRTUAL MEMORY

1. ***Write the comparison between Hard Real Time and Soft Real Time.*** **(2023-24)**

Comparison between Hard Real Time and Soft Real Time

Aspect	Hard Real-Time Systems	Soft Real-Time Systems
Timing Guarantee	Strict timing guarantee.	Timing guarantee is not as strict.
Deadline Enforcement	Missing deadlines is not acceptable.	Missing deadlines may be acceptable in certain cases.
Priority of Tasks	High-priority tasks are prioritized over others.	Tasks have relative priorities, but missing deadlines of low-priority tasks may be acceptable.
Response Time	Response time is deterministic and known.	Response time may vary based on system load.
Example	Airbag deployment in a car, Aircraft control systems.	Multimedia streaming, Online gaming.
Tolerance to Overload	Intolerant to overload. System must meet all deadlines.	Tolerant to occasional overload, may miss some deadlines.
Resource Utilization	Resource utilization is typically lower.	Resource utilization can be higher.
Predictability	Highly predictable behavior.	Predictability varies based on system load.
Cost	Implementation may be more costly due to stringent requirements and specialized hardware.	Implementation may be less costly due to less stringent requirements and flexibility in timing constraints.

2. ◆ ***What is the purpose of system call?*** **(2023-24, 2014)**

[A.2]

- ***What do you understand by system calls? List out the various system calls for process management.*** **(2016)**
- ***What is system call in the context of operating system?*** **(2015)**

System calls provide the interface to a running program and the operating system. User program receives operating system services through the set of system calls. Earlier these calls were available in assembly language instructions but nowadays these features are supported through high level languages like C, Pascal etc. which replaces assembly languages for system programming. The use of system calls in C or Pascal programs very much resemble predefined function or subroutine calls. As an example how systems calls are used, let us consider a simple program to copy data from one file to another. In an interactive system, the following system calls will be generated by the operating system.

(1) Prompt messages for inputting two file names and reading it from terminal.
(2) Open source and destination file.
(3) Prompt error messages in case the source file cannot be open because it is protected against access or destination file cannot be created because there is already a file with this name.
(4) Read the source file.
(5) Write in to destination file.
(6) Display status information regarding various read/writes error conditions for example, the program may find that the end of the file has been reached or that there was a hardware failure. The write operation may encounter various errors, depending upon the output device.
(7) Close both files after the entire file is copied.
(8) All interaction between the program and its environment must occur at the result of requests from the program to the operating system.

3. ***Distinguish between batch systems and time sharing systems.*** **(2023-24)**

Difference Sharing S

Aspect	
Mode of Opera	
User Interactio	
Scheduling	
Execution Tim	
Resource Allocation	
System Utiliz	
Response T	
Throughput	
Examples	

4. ◆ *E*
◆ *W*
◆ *D*
n
n
◆ *W*
c
◆ *I*
t
◆

Difference between Batch Systems and Time Sharing Systems

Aspect	Batch Processing Systems	Time-Sharing Systems
Mode of Operation	Non-interactive.	Interactive.
User Interaction	Limited or no user interaction during job execution.	Provides direct interaction with the system for users.
Scheduling	Jobs are processed in batches.	CPU time is shared among multiple users or processes.
Execution Time	Longer execution time for each job.	Shorter time slices allocated to each user/process.
Resource Allocation	Resources are allocated to each job in advance.	Resources are allocated dynamically based on demand.
System Utilization	Typically higher utilization of system resources.	System resources may not be fully utilized.
Response Time	Longer response time for user requests.	Shorter response time due to interactive nature.
Throughput	Typically lower throughput due to sequential execution of jobs.	Higher throughput due to parallel execution of multiple tasks.
Examples	Payroll processing, batch report generation.	Interactive computing, web servers, email servers.

4. ◆ ***Explain Demand Paging.*** **(2023-24)**
   ◆ ***What is demand paging? Explain.*** **(2022-23)**
   ◆ ***Define and explain the concept of virtual memory. Write the advantages of using virtual memory.*** **(2018)**
   ◆ ***Write the advantages and disadvantages of demand paging.*** **(2018)**
   ◆ ***Discuss Paging and Segmentation in terms of logical address to physical address translation technique.*** **(2017)**
   ◆ ***Discuss virtual memory. Which method is used to implement virtual memory technique.*** **(2016)**

- ***What is Virtual Memory? Why are paging and segmentation used in memory management? Draw the block diagram of paging hardware and explain it.*** **(2016)**
- ***What is the concept of demand paging? Explain the following algorithm of first fit, best fit and worst fit allocation with suitable example.*** **(2016)**

Virtual Memory

Virtual memory can be implemented as an extension of paged or segmented memory management scheme, also called demand paging or demand segmentation respectively.

Advantages of Using Virtual Memory

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

(1) User written error handling routines are used only when an error occurred in the data or computation.

(2) Certain options and features of a program may be used rarely.

(3) Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.

(4) The ability to execute a program that is only partially in memory would counter many benefits.

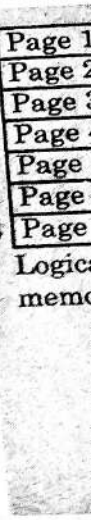
(5) Less number of I/O would be needed to load or swap each user program into memory.

(6) A program would no longer be constrained by the amount of physical memory that is available.

(7) Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

Demand Paging : In demand paging, pages are loaded only on demand, not in advance. It is similar to paging

system with sw
entire program
which are requi



Advantage
(1) Large
(2) More
(3) Uncon
on de

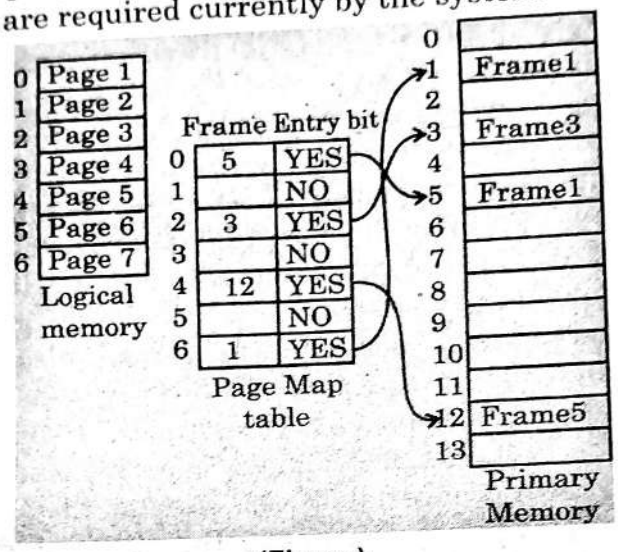
Disadvan
(1) Num
for h
case
(2) Due
add

Paging Manag

Segmen
need of
process.
Logical
address
table co
memory
the log
the loca

P
frames

system with swapping feature. Rather than swapping the entire program in memory, only those pages are swapped which are required currently by the system.



(Figure)

Advantages of Demand Paging :

(1) Large virtual memory.

(2) More efficient use of memory.

(3) Unconstrained multiprogramming. There is no limit on degree of multiprogramming.

Disadvantages of Demand Paging :

(1) Number of tables and amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

(2) Due to the lack of an explicit constraint on jobs address space size.

Paging and Segmentation Used in Memory Management

Segmented Paging : This scheme does away with the need of allocating a single contiguous partition to each process. Physical address space is classified into frames. Logical address space is classified into pages. Every logical address consists of a page number and an offset. The page table contains the base address of each page in physical memory. That base address is added to the offset part of the logical address and that gives the physical address of the location.

Paging itself is a form of dynamic relocation. As frames are discrete blocks, a certain amount of internal

fragmentation is unavoidable. Small page sizes lead to larger page tables (and hence larger memory requirements as page table has to be kept in a fast memory) and large page sizes lead to internal fragmentation. Page size selection has to be supported by hardware. Allocating a process its own separate page table leads us to natural protection without having base and limit registers. There is usually an additional data structure called frame table, which has one entry for each frame, indicating whether the latter is free or allocated and if allocated to which page and process. Paging results in a larger overhead for system calls, and hence the context switch time is also increased.

Implementation of a Page Table : A page table can be implemented as a set of registers if it is small. But it is often not that small. Then we can implement it in the memory with a page table base register that points to the first entry to the symbol table. But this would slow down memory accesses by a factor of 2. Hence we use a set of associative registers or translation look-aside buffers which hold a part of page table which contain frequently referenced memory locations. This is implemented in a fast cache. The rest of the page table is in the memory and a miss at the associative registers would mean a high access time.

Protection Issues to be Satisfied in the Page Table : Associate each entry of a page table with an access bits, such as read write and execute bits and valid/invalid bits which specify access rights and validity of data in a page. A page table length register may also be used which serves the same purpose as the limit register.

Multilevel Paging to Reduce Memory Requirements : An additional level of paging can be used so that only one level of a page table is required to be present on the high speed memory. If we use a 2-level paging structure, the logical address should consist of three parts. An index to the outer page table, another as an index to the inner page which is specified in the page table indicated by the outer level page table and finally an offset is in the page.

The deeper page table is kept in the main memory if needed. Optimizations can be done so that most of the frequently accessed pages are kept in the page table itself and hence, the hit ratio can be improved. Thus, although

the memory
done, memor
Inverted Pa
for each pro
Each field c
which is th
question is
process id,
searched in
found, then
obtain a phy
Page Shar
use the sam
have the op
memory. Th
compilers e
it should n
can easily
have to ha
required th
should ma
possible in
Paged S
scheme w
support.
10*K* array
page size
unit. In
array. Th
storing d
on. A log
and the
Implem
page tab
evaluati
it is not
table ba
This ap
Protec
implem
equally
illegal.

the memory accesses required are doubled, if caching is done, memory access time will be small in most cases.

Inverted Page Tables : Instead of having one page table for each process, we can have one for the whole system. Each field consists of a process id and a page number which is the number of the page that the process in question is using. Each logical address is in the form of < process id, page number, offset >. The first 2 fields are searched in the page table for a match and if none are found, then it is an illegal memory access or else we can obtain a physical address of the page.

Page Sharing and Re-entrant Codes : If a lot of users use the same program which does not modify itself, then we have the option of keeping only one copy of the program in memory. This is particularly useful in case of text editors, compilers etc. But for this the code should be re-entrant, i.e., it should not modify itself during execution. A text editor can easily be implemented in such a way. But for this, we have to have separate page tables for each process as it is required that two logical addresses from different processes should map to the same physical address which is not possible in the case of inverted page tables.

Paged Segmentation : It is a memory management scheme where the user's view of memory is given full support. Example, pages are of fixed size usually, hence, a 10*K* array has to be split into 4 parts if the only available page size is 3*K*. The user's view is to treat the array as a unit. In segmentation, we allocate a 10*K* segment to the array. Thus, a segment may store global variables, other storing data structures which are local to a function and so on. A logical address consists of the number of the segment and the offset.

Implementation of the Segment Table : The issues of page table implementation is also valid for segment table evaluation. If it is small and can be kept in fast registers. If it is not small we can keep it in memory with a segment table base register and a segment table length register. This approach naturally takes care of protection.

Protection and Sharing : Protection is naturally implemented as the data in a segment are considered equally. So a reference which strays outside a segment is illegal. Sharing is also implemented at the segment level

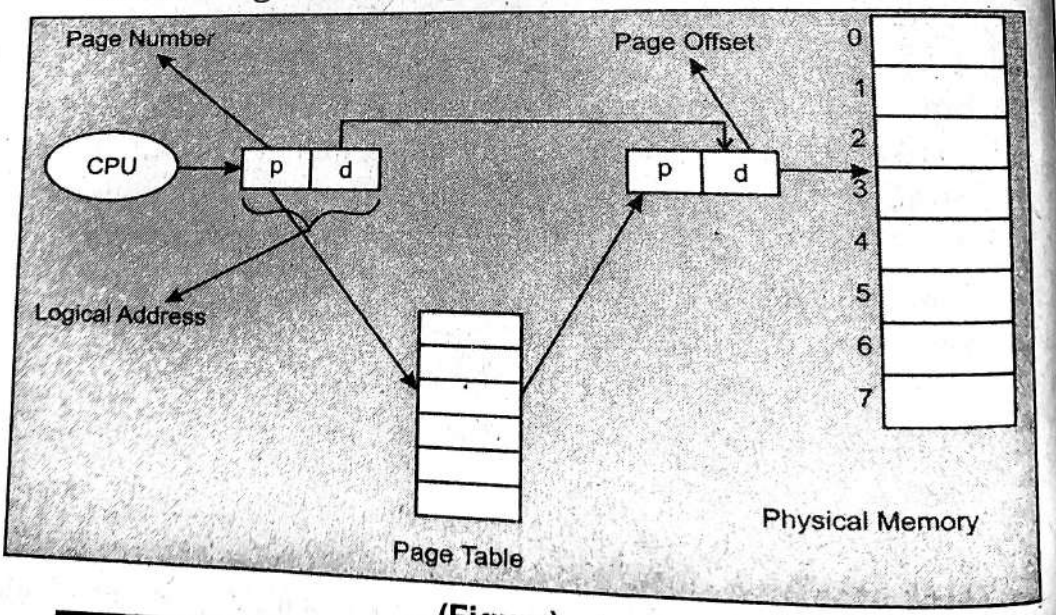
without too many difficulties. A segment can be shared and not a part of a segment. This requires the slightest of modifications but requires some OS support. Two processes can have the same segment in the segment table. No parts of segments can be shared.

Fragmentation : Segmentation due to variable size of segments introduces the problem of external fragmentation. But as segmentation by itself is a dynamic relocation problem, we can use storage compaction.

Paging Hardware : Paging is a memory-management technique that provides the non contiguous address space in main memory. Paging avoids external fragmentation. In this technique physical memory is broken into fixed-sized blocks called frames and logical memory is divided into blocks of the same size called pages. When a process is to be executed, its pages are loaded into any available memory frames from the backing store.

The address generated by CPU is called logical address and it is divided into two parts a page number (p) and a page offset (d). The page number is used as an index into a page table. The page table contains the base address of each page in physical memory. The combination of base address and page offset is used to map the page in physical memory address. Hardware decides the page size.

Block Diagram of Paging Hardware :



(Figure)

Algorithm of First Fit, Best Fit, Worst Fit

First Fit : Allocate the first fit hole that is big enough.

Best Fit : Allocate the smallest hole that is big enough;

must sear
the smalle
Worst Fi
entire list.
Example
size and t
for the fi
start fron
chances b
that seat
them sinc
comfortab
In case t
may be a
large), y
appropri
seat to th
line and
Exampl
but after
for the r
the star
allotted
seats so
job at
keep in
Examp
people
this ti
you ar
them.
when
skip a
the pe
snugly
since
fitting
have f

5. ◆

◆

must search entire list, unless ordered by size. Produces the smallest leftover hole.

Worst Fit : Allocate the largest hole; must also search entire list. Produces the largest leftover hole.

Example of First Fit : You measure/look at each person's size and then go sequentially through all the seats looking for the first seat that the person can sit in. You always start from the first seat as you don't want to take any chances by missing a valid seat; you then allocate them that seat. You don't care if the seat is a bit larger than them since you just want to allocate a seat that they will be comfortable in, even though you are wasting some space. In case there is no seat of the right size available (there may be a few small seats available but the person is extra large), you simply wait till someone seated in an appropriate seat gets up and leaves and then you give that seat to the person waiting, or you look at the next person in line and see if they have a seat available for them.

Example of Worst Fit : You do the same thing as above, but after allotting a seat to someone, you start your search for the next seat from the last seat you allotted rather than the start. This saves you from looking at seats you already allotted at the beginning and forces you to look at the back seats sooner than before, thus you are able to do a faster job at allotting seats to various people. You just have to keep in mind the last seat you allotted to a person.

Example of Best Fit : Again, you are allotting seats to people based on their size in a similar way as above, but this time you take the extra effort to make sure the seat you are allotting someone is the best possible seat for them. You ensure that there is minimum wastage of space when the person sits in his seat. This may require you to skip a large enough seat for one later that is a better fit to the person in question, but you ensure that the person fits snugly into his seat. Also, this may take you more time since you are looking at more seats trying to find the best fitting seat for the current person, even though you may have found a large enough seat already.

5. ◆ ***List two differences between logical and physical addresses.*** **(2023-24)**

◆ ***Differentiate the logical and physical address with an example.*** **(2018)**

Difference Between Logical and Physical Address

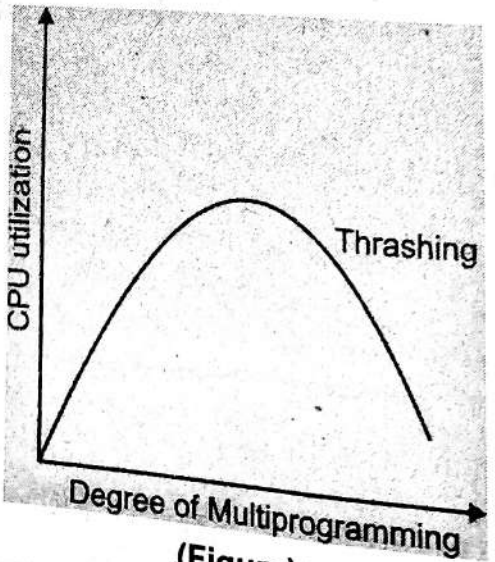
Basis of Distinction	Logical Address in Operating System	Physical Address in Operating System
Definition	The address of something that the central processing system generates.	The actual address of something that the central processing system makes.
Nature	Comes out due to the CPU	Shows as the location of the logical address that is not virtual.
Space	The set of all the logical addresses that the CPU generates with the program reference	The set of all the addresses that get mapped to each logical address.
Variation	Keeps on changing	Always stays the same
Relation	Helps to reach the physical address.	Always stays hidden from the eye of user.
Example	IPv4 is an example of Logical Address	MAC (Medium Access Control) Address is a physical address.

6. ◆ ***Write short note on the Thrashing.*** **(2023-24)**

◆ ***What is thrashing? Explain causes of thrashing in detail. Also suggest its solution.*** **(2019)**

Thrashing

Thrashing is a condition or a situation when the system is spending a major portion of its time in servicing the page faults, but the actual processing done is very negligible.



(Figure)

The ba
allocated too
too frequent
be done by
drastically.
improve th
processes in
multiprogra
in the CPU
higher page
multiprogra

Causes of

It resu

(1) If CP
degre
proces
algori
decre
of mu

(2) CPU
mult

(3) As t
utili

(4) If t
furth
shar

(5) So,
stop
mu

How To
used.

The Wo
looking
This def

Localit
moves fr

A
together
differen

7. ***Write t***
compl

The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no useful work would be done by the CPU and the CPU utilization would fall drastically. The long-term scheduler would then try to improve the CPU utilization by loading some more processes into the memory thereby increasing the degree of multiprogramming. This would result in a further decrease in the CPU utilization triggering a chained reaction of higher page faults followed by an increase in the degree of multiprogramming, called Thrashing.

Causes of Thrashing

It result in severe performance problems.

(1) If CPU utilization is too low then we increase the degree of multiprogramming by introducing a new process to the system. A global page replacement algorithm is used. The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming.

(2) CPU utilization is plotted against the degree of multiprogramming.

(3) As the degree of multiprogramming increases, CPU utilization also increases.

(4) If the degree of multiprogramming is increased further, thrashing sets in and CPU utilization drops sharply.

(5) So, at this point, to increase CPU utilization and to stop thrashing, we must decrease the degree of multiprogramming.

How To Prevent Thrashing : Several techniques are used.

The Working of Set Model (Strategy) : It starts by looking at how many frames a process is actually using. This defines the locality model.

Locality Model : It states that as a process executes, it moves from locality to locality.

A locality is a set of pages that are actively used together. A program is generally composed of several different localities which overlap.

7. *Write the comparison between turnaround time and completion time.* **(2023-24)**

Difference between Turnaround Time and Completion Time

Aspect	Turnaround Time	Completion Time
Definition	The total time taken for a process to complete its execution, including waiting time in the ready queue and actual execution time.	The time at which a process finishes its execution and terminates.
Components	Turnaround time = Completion time - Arrival time.	Completion time is the time at which the process finishes execution, including any waiting time in the ready queue and actual execution time.
Unit of Measurement	Time (e.g., milliseconds, seconds)	Time (e.g., milliseconds, seconds)
Calculation	Turnaround time for a process is calculated by subtracting its arrival time from its completion time.	Completion time is recorded when a process completes its execution.
Significance	Reflects the total time a process spends in the system, including both waiting and execution time.	Indicates when a process finishes its execution and becomes inactive or terminates.
Application	Used to evaluate system performance and efficiency in job or process scheduling.	Used for process scheduling and performance evaluation.

8. *Write short note on the Segmentation.* (2023-24)

Segmentation

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user's view of the process which paging does not provide.

Types of Segmentation in Operating System :

(1) **Virtual Memory Segmentation :** Each process is divided into a number of segments, but the segmentation is not done all at once. This segmentation may or may not take place at the run time of the program.

(2) **Simple Se**
a number
memory
contiguous

Advantages of

(1) Segment
Page tabl
all at onc

(2) The use
similar
program
modules
codes.

(3) The us
paging,
Segmer
segreg

(4) **Flexib**
of flex
size, a
segme
alloca

(5) **Shar**
mem
usef
code

(6) **Pro**
pro
fro
me
an

Disadv
associat
D
segmen
to retri

(1) **F**
l
d

(2)

(2) **Simple Segmentation :** Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

Advantages of Segmentation :

(1) Segment Table consumes less space in comparison to Page table in paging. As a complete module is loaded all at once, segmentation improves CPU utilization.

(2) The user's perception of physical memory is quite similar to segmentation. Users can divide user programs into modules via segmentation. These modules are nothing more than separate processes' codes.

(3) The user specifies the segment size, whereas, in paging, the hardware determines the page size. Segmentation is a method that can be used to segregate data from security operations.

(4) **Flexibility :** Segmentation provides a higher degree of flexibility than paging. Segments can be of variable size, and processes can be designed to have multiple segments, allowing for more fine-grained memory allocation.

(5) **Sharing :** Segmentation allows for sharing of memory segments between processes. This can be useful for inter-process communication or for sharing code libraries.

(6) **Protection :** Segmentation provides a level of protection between segments, preventing one process from accessing or modifying another process's memory segment. This can help increase the security and stability of the system.

Disadvantages of Segmentation : Overhead is associated with keeping a segment table for each activity.

Due to the need for two memory accesses, one for the segment table and the other for main memory, access time to retrieve the instruction increases.

(1) **Fragmentation :** As mentioned, segmentation can lead to external fragmentation as memory becomes divided into smaller segments. This can lead to wasted memory and decreased performance.

(2) **Overhead :** Using a segment table can increase overhead and reduce performance. Each segment

table entry requires additional memory, and accessing the table to retrieve memory locations can increase the time needed for memory operations.

(3) **Complexity :** Segmentation can be more complex to implement and manage than paging. In particular, managing multiple segments per process can be challenging, and the potential for segmentation faults can increase as a result.

9. ◆ ***Define and explain page replacement algorithms.*** **(2023-24)**
◆ ***Discuss the following page replacement algorithm with an example:***
(1) ***Optimal***
(2) ***LRU.*** **(2022-23)**
◆ ***Write the name of various page replacement algorithms. Explain any two methods.*** **(2018)**
◆ ***Why page replacement is needed? Describe.*** **(2015)**

While a used process is executing, a page fault occurs. The operating system determines where the desired page is residing on the disk but then finds that there are no free frames on the free-frame list; all memory in use. The operating system could instead swap out a process, freeing all its frames and reducing the level of multiprogramming we discuss the most common solution page replacement.

Basic Page Replacement

If no frame is free, we find one that is not currently being used and free it. We can free a frame by writing its contents to swap space and changing the page table to indicate that the page is no longer in memory.

(1) Find the location of the desired page on disk.
(2) Find a free frame :
  (a) If there is a free frame, use it.
  (b) If there is no free frame, use a page–replacement algorithm to select a victim frame.
  (c) Write the victim frame to the disk, change the page and frame tables accordingly.
(3) Read the desired page, into the newly free frame; change the page and frame table.
(4) Restart the user process. If no frames are free, two pages transfers are required.

Page replacem
completes the sepa
physical memory.
FIFO : The scheme
Out. There is a queu
to each page is ente
memory. Replacing
inserting the new p
simply do the repla
used regularly will
and then back in m
the page that is res
memory. Whenever
is identified and re
LRU : It is Leas
searches in memo
page was lastly u
time for that pag
order based-upon
used for a while i
the near future.
page, by assumi
past is least like
this no single p
referenced again
OPT : This alg
used for the lon
be referenced i
the lowest poss
frames.
This poli
future knowled
number of pag
IN rows. In ot
most distant f
has the page f
LRU-Approx
computer sys
true LRU p
hardware sup
must used. M
the form of a
set by the

Page replacement is basic to demand paging. It completes the separation between logical memory and physical memory.

FIFO : The scheme works on the basis of First In and First Out. There is a queue and a reference string O. A reference to each page is entered into the queue as it is brought into memory. Replacing the page at the head of the queue and inserting the new page at the tail of the queue can then simply do the replacement of pages. The pages that are used regularly will be swapped out on to secondary storage and then back in many times under this system. It replaces the page that is resident and has spent the longest time in memory. Whenever a page is to be evicted, the oldest page is identified and removed from memory.

LRU : It is Least Recently Used scheme. This scheme searches in memory that how long has been, since each page was lastly used by associating with each page. The time for that pages last use and puts them in ascending order based-upon it. This takes the page that has not been used for a while it is probably not going to be used again in the near future. It replaces the least recently used resident page, by assuming that the page used in the most distant past is least likely to be referenced in the near future. In this no single page is replaced immediately before being referenced again.

OPT : This algorithm replaces the page that will not be used for the longest period of time or removes the page to be referenced in the most distant future. Add guarantees the lowest possible page fault rate for a fixed number of frames.

This policy is difficult to implement, as it requires future knowledge which would not be easily available. The number of page fault may be observed from the respective IN rows. In other words, the page is to be referenced in the most distant future. Since, it requires future knowledge, it has the page fault frequency is about to 28.

LRU-Approximation Page Replacement : Few computer systems provide sufficient hardware support for true LRU page replacement some system provide no hardware support, and other page replacement algorithms must used. Many systems provide some help; however in the form of a reference bit, the reference bit for a page is set by the hardware whenever that page is referenced.

Reference bits are associated with each entry in the page table.

Initially, all bits are cleared (to 0) by the operating system. As a user process executes, the bit associated with each page referenced is set (to 1) by the hardware. After some time, we can determine which pages have been used and which have not been used by examining therefore bits although we do not know the order of use. This information is the basis of for many page replacement algorithms that approximate LRU replacement.

Counting – Based Page Replacement : There are many other algorithms that can be used for page replacement For example we can keep a counter of the number of references that have been made to each page and develop the following two schemes. The Least Frequently Used (LFU) page replacement algorithm requires that the page with the smallest count be replaced. The reason for this selection is that actively used pages should have a large reference count. The Most Frequently Used (MFU) page replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

10. ***Consider the following page reference string :***

1, 2, 3, 4, 2, 1, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

Identify the no. of page faults would occur for the following page replacement algorithms, assuming three frames are initially empty.

(1) ***LRU replacement***

(2) ***Optimal replacement*** **(2023-24)**

Page Reference String

1,2,3,4,2,1,6,2,1,2,3,7,6,3,2,1,2,3,6

Solution :

(1) LRU Replacement :

F3			3	3	3	3	6	6	6	6	6	7	7	7	7	1	1	1
F2		2	2	2	2	1	1	1	1	1	3	3	3	3	2	2	2	2
F1	1	1	1	4	4	4	4	2	2	2	2	2	6	6	6.	6	6	3
Hit					H				H	H				H			H	
Page Fault	Pf	Pf	Pf	Pf		Pf	Pf	Pf			Pf	Pf	Pf		Pf	PF		Pf

Page Fault : 14

(2) Optimal Page

F3			3	4	4
F2		2	2	2	2
F1	1	1	1	1	1
Hit					H
Page Fault	Pf	Pf	Pf	Pf	

Page Fault : 12

11.
- ***Define oper from differ***
- ***Define ope kind of s system.***
- ***What are Operating***
- ***What is necessary***
- ***Write the system.***

Operating Sys

The operat
that is on a com
the computer a
The operating
computer needs
mouse or the k
are on the com
programs that
access to the c

The mo
Microsoft's W
computers ha
recent of whi
source operati

Role of Ope

Most op
below. An ope
which manag
memory, proc

(1) Proces
module

(2) Optimal Page Replacement :

F3			3	4	4	1	6	6	6	6	6	6	6	6	6	1	1	1	6
F2		2	2	2	2	2	2	2	2	2	3	7	7	3	2	2	2	2	2
F1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	H	H
Hit					H			H	H	H			H						
Page Fault	Pf	Pf	Pf	Pf		Pf	Pf				Pf	PF		Pf	Pf	PF			PF

Page Fault : 12

11.
- ***Define operating systems and discuss its role from different perspectives.*** **(2022-23)**
- ***Define operating system. Describe briefly the kind of services provided by an operating system.*** **(2018)**
- ***What are major services provided by an Operating System?*** **(2017)**
- ***What is an operating system? Why is it necessary for a computer system?*** **(2016)**
- ***Write the three main purpose of an operating system.*** **(2014)**

Operating System

The operating system is the most important program that is on a computer. The operating system basically runs the computer and allows other programs to run as well. The operating system does all the basic things that a computer needs to do, such as recognizing inputs from the mouse or the keyboard. It keeps track of where all the files are on the computer. It allocates resources to the various programs that are running and it prevents unauthorized access to the computer.

The most popular operating system today is Microsoft's Windows operating system. Macintosh computers have their own operating system, the most recent of which is called Mac OS X. There are also open source operating systems such as Linux.

Role of Operating System in Computer

Most operating system performs the purposes given below. An operating system is a large collection of software which manages resources of the computer system, such as memory, processor, file system and input / output device.

(1) Process Management : Process management module takes care of allocation creation and deletion

of processes, scheduling of system resources to different processes requesting them, and providing mechanism for synchronization and communication among processes.

(2) **Memory Management :** Memory management module takes care of allocation and de-allocation of memory space to programs in need of these resources.

(3) **File Management :** File management module takes care of file-related activities such as organization, storage, naming, sharing, and protection of files.

(4) **Security :** Security module protects the resources and information of a computer system against distraction and unauthorized access.

(5) **Command Interpretation :** Command interpretation module takes care of interpreting user commands, and directing system resources to process the commands. With this mode of interaction with a system, users are not much concerned about hardware detail of the system.

An operating system acts as an intermediate between the user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

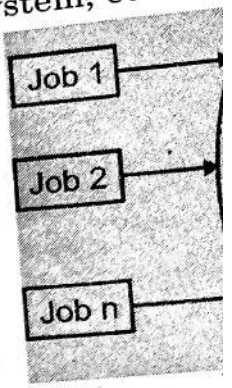
User interacts with the computer through operating system in order to accomplish his task since; it is his primary interface with a computer. It helps users to understand the inner functions of a computer very closely. Many concepts and techniques found in operating system have general applicability in other applications.

12. *Write short note on Simple Batch Systems.*

(2022-23)

In the simple batch operating system, there is no direct communication between the user and the computer. In this, firstly, the user submits a job to the computer operator, and after submitting the job, the computer operator creates a batch of the jobs on an input device. The batch of jobs is created on the basis of the type of

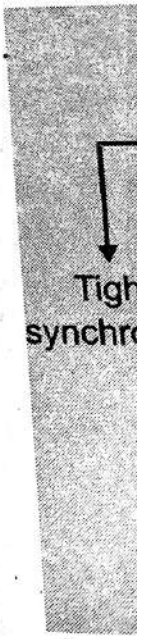
language and n
then a specia
program in a
system, etc.



13. *Write shor*

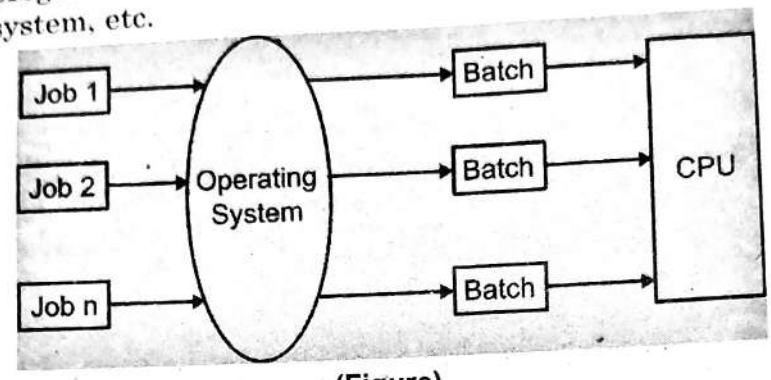
Parallel C

Parall
the executi
segments.
simultaneo
a single c
computers
parallel pr



14. *Explo*

language and needs. After the batch of the job is created, then a special program monitors and manages each program in a batch. Example: Bank Statements, Payroll system, etc.

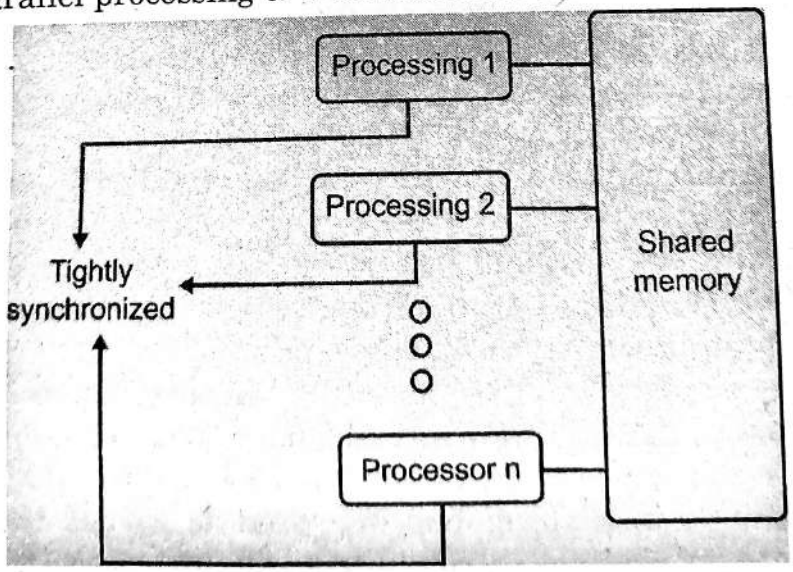


(Figure)

13. *Write short note on Parallel Systems.* (2022-23)

Parallel Operating System

Parallel operating systems are designed to speed up the execution of programs by dividing them into multiple segments. It is used for dealing with multiple processors simultaneously by using computer resources which include a single computer with multiple processors and several computers connected by a network to form a cluster of parallel processing or a combination of both.

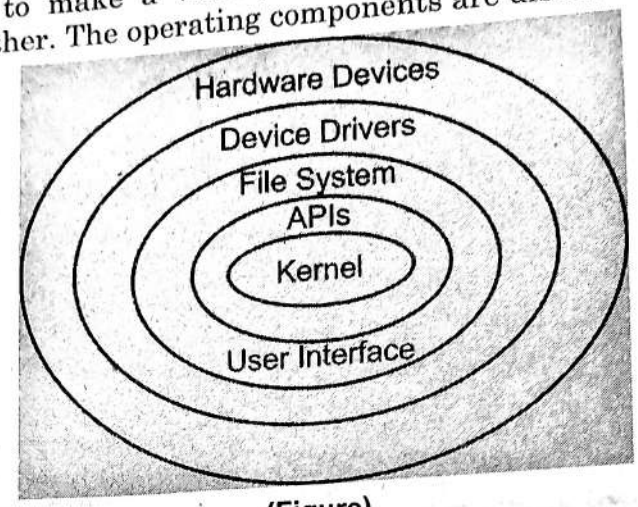


(Figure)

14. *Explain the components of operating system.* (2022-23)

Components of Operating System

The components of an operating system play a key role to make a variety of computer system parts work together. The operating components are discussed below.



(Figure)

Operating-System-Components

Kernel : The kernel in the OS provides the basic level of control on all the computer peripherals. In the operating system, the kernel is an essential component that loads firstly and remains within the main memory. So that memory accessibility can be managed for the programs within the RAM, it creates the programs to get access from the hardware resources. It resets the operating states of the CPU for the best operation at all times.

Process Execution : The OS gives an interface between the hardware as well as an application program so that the program can connect through the hardware device by simply following procedures & principles configured into the OS. The program execution mainly includes a process created through an OS kernel that uses memory space as well as different types of other resources.

Interrupt : In the operating system, interrupts are essential because they give a reliable technique for the OS to communicate & react to their surroundings. An interrupt is nothing but one kind of signal between a device as well as a computer system otherwise from a program in the computer that requires the OS to leave and decide accurately what to do subsequently. Whenever an interrupt signal is received, then the hardware of the computer puts on hold automatically whatever computer

program is runn
computer progra
interrupt.
Memory Mana
nothing but me
memory & move
disk & main m
each & every m
process otherwi
can be allocate
know which p
Whenever me
correspondingl
management
groups like m
application me
Multitasking
independent
system. Mult
execute one o
computers ca
this can be d
program use
Networking
processor
communicat
network m
safety, the p
Preser
different ne
This invol
operating s
to share re
which use
wireless.
Security
allow the
many proc
This syst
technologi
systems g

program is running presently, keeps its status & runs a computer program which is connected previously with the interrupt.

Memory Management : The functionality of an OS is nothing but memory management which manages main memory & moves processes backward and forward between disk & main memory during implementation. This tracks each & every memory position; until it is assigned to some process otherwise it is open. It verifies how much memory can be allocated to processes and also makes a decision to know which process will obtain memory at what time. Whenever memory is unallocated, then it tracks correspondingly to update the status. Memory management work can be divided into three important groups like memory management of hardware, OS and application memory management.

Multitasking : It describes the working of several independent computer programs on a similar computer system. Multitasking in an OS allows an operator to execute one or more computer tasks at a time. Since many computers can perform one or two tasks at a time, usually this can be done with the help of time-sharing, where each program uses the time of a computer to execute.

Networking : Networking can be defined as when the processor interacts with each other through communication lines. The design of communication-network must consider routing, connection methods, safety, the problems of opinion & security.

Presently most of the operating systems maintain different networking techniques, hardware, & applications. This involves that computers that run on different operating systems could be included in a general network to share resources like data, computing, scanners, printers, which uses the connections of either wired otherwise wireless.

Security : If a computer has numerous individuals to allow the immediate process of various processes, then the many processes have to be protected from other activities. This system security mainly depends upon a variety of technologies that work effectively. Current operating systems give an entrée to a number of resources, which are

obtainable to work the software on the system, and to external devices like networks by means of the kernel. The operating system should be capable of distinguishing between demands which have to be allowed for progressing & others that don't need to be processed. Additionally, to permit or prohibit a security version, a computer system with a high level of protection also provides auditing options. So this will allow monitoring the requests from accessibility to resources

User Interface : A GUI or user interface (UI) is the part of an OS that permits an operator to get the information. A user interface based on text displays the text as well as its commands which are typed over a command line with the help of a keyboard.

The OS-based applications mainly provide a specific user interface for efficient communication. The main function of a user interface of an application is to get the inputs from the operator & to provide o/ps to the operator. But, the sorts of inputs received from the user interface as well as the o/p types offered by the user interface may change from application to application. The UI of any application can be classified into two types namely GUI (graphical UI) & CLI (command line user interface).

15. *Explain paging scheme of memory management. What hardware support is needed for its implementation?* **(2022-23)**

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. Additionally, frames will be used to split the main memory. This scheme permits the physical address space of a process to be non – contiguous.

Memory management at the hardware level is concerned with the physical components that store data, most notably the random access memory (RAM) chips and CPU memory caches (L1, L2 and L3).

16. ◆ ***Given memory partitions of 100 KB, 500 KB, 200 KB, 300 KB and 600 KB (in order), how would each of the first-fit, best-fit and worst-fit algorithms place processors of 212 KB, 417 KB, 112 KB and 426 KB (in-order)? Which algorithm makes the most efficient use of memory?*** **(2022-23)**

◆ ***For the following partitions of 100K, 500K, 300K and 600K (in order) place the process of 212K, 417K, 112K and 426K (in order) according to best-fit and first-fit algorithms.*** **(2016)**

First-Fit :

212 *K* is put in 500 *K* partition
417 *K* is put in 600 *K* partition
112 *K* is put in 288 *K* partition (New partition : 500 *K* – 212 *K* = 288 *K*)
426 *K* must wait

Best-Fit :

212 *K* is put in 300 *K* partition
417 *K* is put in 500 *K* partition
112 *K* is put in 200 *K* partition
426 *K* is put in 600 *K* partition

Worst-Fit :

212 *K* is put in 600 *K* partition
417 *K* is put in 500 *K* partition
112 *K* is put in 388 *K* partition (New partition: 600 *K* – 212*K* = 388 *K*)
426 *K* must wait
So Best-Fit turns out to be best.

17. ***How jobs are scheduled in multiple processor system?*** **(2019)**

Multiple-Processor Scheduling

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. In multiple processor scheduling there are cases when the processors are identical i.e. HOMOGENEOUS, in terms of their

functionality, we can use any processor available to run any process in the queue.

One approach is when all the scheduling decisions and I/O processing are handled by a single processor which is called the Master Server and the other processors executes only the user code. This is simple and reduces the need of data sharing. This entire scenario is called Asymmetric Multiprocessing.

A second approach uses Symmetric Multiprocessing where each processor is self scheduling. All processes may be in a common ready queue or each processor may have its own private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.

18. *Differentiate between hard real time system and soft real time system.* **(2019)**

Difference Between Hard Real Time System and Real Time System

Hard Real Time System	Soft Real Time System
A Hard Real-Time System guarantees that critical tasks complete on time. This goal requires that all delays in the system be bounded from the retrieval of the stored data to the time that it takes the operating system to finish any request made of it.	A Soft Real Time System where a critical real-time task gets priority over other tasks and retains that priority until it completes. As in hard real time systems kernel delays need to be bounded.

19. *How the logical address is converted to physical address? Explain.* **(2019)**

Address Translation

Page address is called logical address and represented by page number and the offset.

Logical Address = Page number + page offset

Frame address is called physical address and represented by a frame number and the offset.

Physical Address = Frame number + page offset

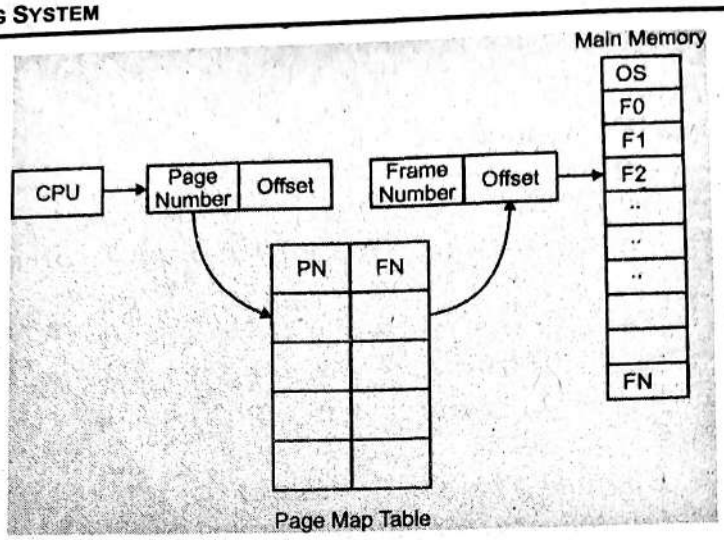
A data structure called page map table is used to keep track of the relation between a page of a process to a frame in physical memory.

Whe
translate
create e
executio
Wh
pages a
Suppose
accomm
paging
runs ou
or unwa
up RA
needed
T
the pro
the m
memo
progra

20. *Exp*

Belo

algor
reduc
Bala
the
incre



(Figure)

When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

20. *Explain Belady's anomaly.* **(2019)**

Belady's Anomaly

In the case of LRU and optimal page replacement algorithms, it is seen that the number of page faults will be reduced if we increase the number of frames. However, Balady found that, In FIFO page replacement algorithm, the number of page faults will get increased with the increment in number of frames.

This is the strange behavior shown by FIFO algorithm in some of the cases. This is an Anomaly called as Belady's Anomaly.

21. ◆ ***What do you understand by internal fragmentation? How this can be resolved?*** **(2019)**

◆ ***What is the fragmentation problem? Describe internal and external fragmentation.*** **(2018)**

◆ ***Write short note on Fragmentation.*** **(May 2013)**

◆ ***Discuss external fragmentation and internal fragmentation with respect to main memory allocation.***

Fragmentation

Fragmentation is as a process which are loaded or removed from memory. The free memory space is broken into little pieces; such types of pieces may or may not be of any use to be allocated individually to any process. This may give rise to term memory waste or fragmentation. When a computer program requests blocks of memory from the computer system, the blocks are allocated in chunks. When the computer program is finished with a chunk, it can free the chunk back to the computer. The size and the amount of time a chunk is held by a program varies.

During its lifespan, a computer program can request and free many chunks of memory. When a program is started, the free memory areas are long and contiguous. Over time and with use, the long contiguous regions become fragmented into smaller and smaller contiguous areas. Eventually, it may become impossible for the program to request large chunks of memory.

Internal Fragmentation

Internal fragmentation occurs when memory allocation is based on fixed-size partitions where after a small size application is assigned to a slot the remaining free space of that slot is wasted. External fragmentation occurs when memory is dynamically allocated where after loading and unloading of several slots here and there the free space is being distributed rather than being contiguous.

(1) Intern
memo
fragm
alloca

(2) Inter
partit
less s
space
fragn
space
for
distri

(3) Exte
wher
that
oper
assig
mov
don
defr

(4) Ext
me
a l
wh
par

(5) Int
pa
ba
fra

22. ***For g***
page
algor
7

Given r
7 0 1 2
G
A
page th

7	

(1) Internal Fragmentation occurs when a fixed size memory allocation technique is used. External fragmentation occurs when a dynamic memory allocation technique is used.

(2) Internal fragmentation occurs when a fixed size partition is assigned to a program/file with less size than the partition making the rest of the space in that partition unusable. External fragmentation is due to the lack of enough adjacent space after loading and unloading of programs or files for some time because then all free space is distributed here and there.

(3) External fragmentation can be mined by compaction where the assigned blocks are moved to one side, so that contiguous space is gained. However, this operation takes time and also certain critical assigned areas for example system services cannot be moved safely. We can observe this compaction step done on hard disks when running the disk defragmenter in Windows.

(4) External fragmentation can be prevented by mechanisms such as segmentation and paging. Here a logical contiguous virtual memory space is given while in reality the files/programs are spitted into parts and placed here and there.

(5) Internal fragmentation can be maimed by having partitions of several sizes and assigning a program based on the best fit. However, still internal fragmentation is not fully eliminated.

22. *For given reference string, calculate the number of page faults using optimal page replacement algorithm. Given page size = 3.* **(2019)**

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Given reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Given page size = 3

According to optimal page replacement, replace the page that will not use for the largest period of time.

7	7	7	2	2	2	2	2	7
	0	0	0	0	4	0	0	0
		1	1	3	3	3	1	1

No of page fault = 9 (Ans.)
Fault Rate = 9/20

23. *Differentiate between Asymmetric multiprocessing and Symmetric multiprocessing.* (2019)

Difference Between Asymmetric Multiprocessing and Symmetric Multiprocessing

Basis for Comparison	Symmetric Multiprocessing	Asymmetric Multiprocessing
Basic	Each processor run the tasks in Operating System.	Only . Master processor run the tasks of Operating System.
Process	Processor takes processes from a common ready queue, or there may be a private ready queue for each processor.	Master processor assign processes to the slave processors, or they have some predefined processes.
Architecture	All processor in Symmetric Multiprocessing has the same architecture.	All processor in Asymmetric Multiprocessing may have same or different architecture.
Communication	All processors communicate with another processor by a shared memory.	Processors need not communicate as they are controlled by the master processor.
Failure	If a processor fails, the computing capacity of the system reduces.	If a master processor fails, a slave is turned to the master processor to continue the execution. If a slave processor fails, its task is switched to other processors.
Ease	Symmetric Multiprocessor is complex as all the processors need to be synchronized to maintain the load balance.	Asymmetric Multiprocessor is simple as master processor

24. *What is a hash table? How is it used?* (2019)

Hash Table

A hash table is a data structure that is used to stor keys/value pairs. It uses a hash function to compute

index into an array searched.

25. *Discuss Secon algorithm.*

In the Secon candidate pages robin matter, and consecutive consi replaced is the on matter, has not be

It can be in bit to each me considered (due t this bit is set to when we conside replace the first bits of the other with the "secon during the first all the other pa

Example : Let' 4 0 4 1 4 2 4 3 algorithm proc the pointer.

(1) Initially, all filled with { 0} as non cycle back

(2) **Pass-4 :** F a second so the can page fault

(3) **Pass-5 :** replaced; second ch

(4) **Pass-6 :** pf=4 (No

(5) **Pass-7 :** survived element a

(6) **Pass-8** reference

index into an array in which an element will be inserted or searched.

25. *Discuss Second chance page replacement algorithm.* (2019)

In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and a page that has been accessed between consecutive considerations will not be replaced. The page replaced is the one that, when considered in a round robin matter, has not been accessed since its last consideration.

It can be implemented by adding a "second chance" bit to each memory frame-every time the frame is considered (due to a reference made to the page inside it), this bit is set to 1, which gives the page a second chance, as when we consider the candidate page for replacement, we replace the first one with this bit set to 0 (while zeroing out bits of the other pages we see in the process). Thus, a page with the "second chance" bit set to 1 is never replaced during the first consideration and will only be replaced if all the other pages deserve a second chance too.

Example : Let's say the reference string is 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4 and we have 3 frames. Let's see how the algorithm proceeds by tracking the second chance bit and the pointer.

(1) Initially, all frames are empty so after first 3 passes they will be filled with {0, 4, 1} and the second chance array will be {0, 0, 0} as none has been referenced yet. Also, the pointer will cycle back to 0.

(2) **Pass-4 :** Frame={0, 4, 1}, second_chance = {0, 1, 0} [4 will get a second chance], pointer = 0 (No page needed to be updated so the candidate is still page in frame 0), pf = 3 (No increase in page fault number).

(3) **Pass-5 :** Frame={2, 4, 1}, second_chance= {0, 1, 0} [0 replaced; it's second chance bit was 0, so it didn't get a second chance], pointer=1 (updated), pf=4

(4) **Pass-6 :** Frame={2, 4, 1}, second_chance={0, 1, 0}, pointer=1, pf=4 (No change)

(5) **Pass-7 :** Frame={2, 4, 3}, second_chance= {0, 0, 0} [4 survived but it's second chance bit became 0], pointer=0 (as element at index 2 was finally replaced), pf=5

(6) **Pass-8 :** Frame={2, 4, 3}, second_chance= {0, 1, 0} [4 referenced again], pointer=0, pf=5

(7) **Pass-9** : Frame={2, 4, 3}, second_chance= {1, 1, 0} [2 referenced again], pointer=0, pf=5

(8) **Pass-10** : Frame={2, 4, 3}, second_chance= {1, 1, 0}, pointer=0, pf=5 (no change)

(9) **Pass-11** : Frame={2, 4, 0}, second_chance= {0, 0, 0}, pointer=0, pf=6 (2 and 4 got second chances)

(10) **Pass-12** : Frame={2, 4, 0}, second_chance= {0, 1, 0}, pointer=0, pf=6 (4 will again get a second chance)

(11) **Pass-13** : Frame={1, 4, 0}, second_chance= {0, 1, 0}, pointer=1, pf=7 (pointer updated, pf updated)

(12) **Page-14** : Frame={1, 4, 0}, second_chance= {0, 1, 0}, pointer=1, pf=7 (No change)

(13) **Page-15** : Frame={1, 4, 2}, second_chance= {0, 0, 0}, pointer=0, pf=8 (4 survived again due to 2nd chance!)

(14) **Page-16** : Frame={1, 4, 2}, second_chance= {0, 1, 0}, pointer=0, pf=8 (2nd chance updated)

(15) **Page-17** : Frame={3, 4, 2}, second_chance= {0, 1, 0}, pointer=1, pf=9 (pointer, pf updated)

(16) **Page-18** : Frame={3, 4, 2}, second_chance= {0, 1, 0}, pointer=1, pf=9 (No change)

In this example, second chance algorithm does as well as the LRU method, which is much more expensive to implement in hardware.

26. ◆ ***Differentiate between segmentation and paging.*** **(2019)**

◆ ***Write the comparison between paging and segmentation.*** **(2018)**

Difference Between Paging and Segmentation

Paging	Segmentation
A page is a physical unit of information.	A segment is a logical unit of information.
A page is invisible to the user's program.	A segment is visible to the user's program.
A page is of fixed size e.g. 4Kbytes.	A segment is of varying size.
The page size is determined by the machine architecture.	A segment size is determined by the user.
Fragmentation may occur.	Segmentation eliminates fragmentation.
Page frames on main memory are required.	No frames are required.

27. ***What are*** ***managemen***

Functions of

(1) Memory operating memory main me

(2) Memory memory some pro

(3) It check processe

(4) It decid time.

(5) It trac unalloc status.

28. ◆ ***What*** ***oper*** ***man***

◆ ***What*** ***oper***

Function

(1) Resou

(2) Data

(3) Job (t

(4) Stand comp

The r computer secondary The data and outpu retrieval. schedules, execution

A jo and their

_chance= {1, 1, 0} [2
nd_chance= {1, 1, 0},
nd_chance= {0, 0, 0},
ances)
nd_chance= {0, 1, 0},
ond chance)
nd_chance= {0, 1, 0},
dated)
nd_chance= {0, 1, 0},
d_chance= {0, 0, 0},
2nd chance!)
d_chance= {0, 1, 0},
_chance= {0, 1, 0},
_chance= {0, 1, 0},

orithm does as well
expensive to

nentation and
(2019)
en paging and
(2018)

gmentation

gmentation
is a logical unit of
t is visible to the ram.
is of varying size.
size is determined
on eliminates on.
re required.

27. *What are the main functions of memory management?* (2018)

Functions of Memory Management

(1) Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.

(2) Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.

(3) It checks how much memory is to be allocated to processes.

(4) It decides which process will get memory at what time.

(5) It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

28. ◆ ***What are the basic functions does an operating system perform as a resource manager?*** (2017)

◆ ***What are various management functions of operating system?***

Functions of an Operating System

(1) Resource management
(2) Data management
(3) Job (task) management
(4) Standard means of communication between user and computer.

The resource management function of an OS allocates computer resources such as CPU time, main memory, secondary storage, and input and output devices for use. The data management functions of an OS govern the input and output of the data and their location, storage, and retrieval. The job management function of an OS prepares, schedules, controls, and monitors jobs submitted for execution to ensure the most efficient processing.

A job is a collection of one or more related programs and their data. A job is a collection of one or more related

programs and their data. The OS establishes a standard means of communication between users and their computer systems. It does this by providing a user interface and a standard set of commands that control the hardware.

Typical Day-to-Day Uses of an Operating System

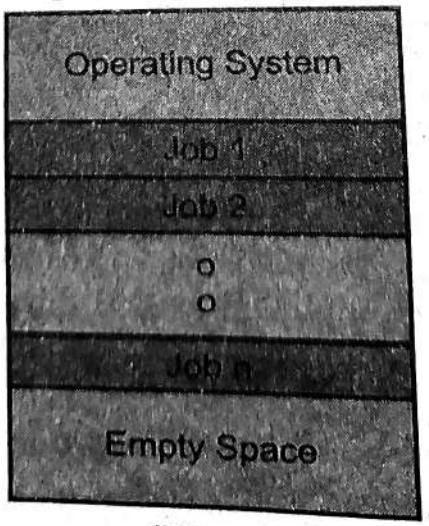
(1) Executing application programs.
(2) Formatting floppy diskettes.
(3) Setting up directories to organize your files.
(4) Displaying a list of files stored on a particular disk.
(5) Verifying that there is enough room on a disk to save a file.
(6) Protecting and backing up your files by copying them to other disks for safekeeping.

29. ◆ ***Write a short note on multiprogramming and time sharing systems.*** **(2016, 2017)**
◆ ***Explain nature of real time operating system.***

Multiprogramming Operating System

Sharing the processor, when two or more programs reside in memory at the same time, is referred as multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

The following figure shows the memory layout for a multiprogramming system.



(Figure)

An OS does the following activities related to multiprogramming.

(1) The ope
at a tim
(2) This se
pool.
(3) The op
of the j
(4) Multip
state
using
the C
proces

Real-Time
system is
have an up
it is desire
in the en
mostly ex
and proce
Examples
simulatio
in milita
system
convenie
concern
process
the rela
processc
process
priority
priority
based p

Time S
operati
with a
comput
a respo

A
simult
action
small
each u
severa

(1) The operating system keeps several jobs in memory at a time.

(2) This set of jobs is a subset of the jobs kept in the job pool.

(3) The operating system picks and begins to execute one of the jobs in the memory.

(4) Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensure that the CPU is never idle, unless there are no jobs to process.

Real-Time Operating System : A real time operating system is that executes programs that are guaranteed to have an upper bound on tasks that they carry out. Usually, it is desired that the upper bound be very small. This used in the environments where a large number of events mostly external to computer systems, must be accepted and processed in a short time or within certain deadline. Examples of such applications are flight control, real time simulations etc. Real time systems are also frequently used in military application. A primary objective of real time system is to provide quick response times. User convenience and resource utilization are of secondary concern to real time system. In the real time system each process is assigned a certain level of priority according to the relative importance of the event it processes. The processor is normally allocated to the highest priority process among those which are ready to execute. Higher priority process usually pre-empted execution of lower priority processes. This form of scheduling called, priority based pre-emptive scheduling.

Time Sharing System : It is a form of multi-programmed operating system which operates in an interactive mode with a quick response time. The user types a request to the computer through keyboard. The computer processes it and a response is displayed on the user's terminal.

A time sharing system allows the many users to simultaneously share the computer resources. Since each action or command in a time shared system take a very small fraction of time, only a little CPU time is needed for each user. The operating system on real-time computer is severally constrained by the timing requirements.

[A.34]

Dedicated computers are special purpose computers that are used to perform only one or more tasks. Often these are real-time computers and include applications such as guided missile and the computers in modern cars that control the fuel injection systems.

30. ◆ ***Under what circumstances does Page fault occur? Describe the actions taken by the operating system when Page fault occurs.*** **(2017)**

◆ ***When do Page Fault occur? Describe the action taken by the operating system when page fault occur.***

◆ ***Explain page fault. List necessary steps to handle page fault. Draw the necessary diagram.***

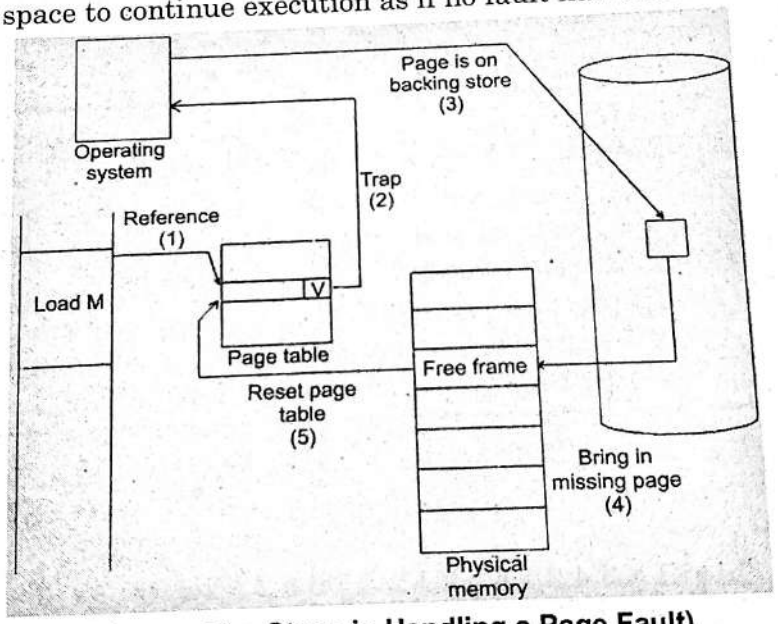
Page Fault

In several systems, when a process is executing with only a few pages in memory and when an instruction is encountered which refers to any instruction or data in some other page which is outside the main memory, a page fault occurs. The operating system discovers that a page fault has occurred and tries to discover which virtual page is needed. Often one of the hardware registers contains this information. If not the operating system must retrieve the program counter fetch the instruction and parse it in software to figure out what it was doing when the fault hit. Once the virtual address that caused the fault is known the operating system checks to see if this address is valid and the protection consistent with the access. If not the process is sent a signal or killed. If the address is valid and no protection fault has occurred the system attempts to acquire a page frame from the list of free frames. If no frames are free the page replace algorithm is run to select a victim. If the page frame selected is dirty the page is scheduled for transfer to the disk and a context switch takes place suspending the faulting process and letting another one run until the disk transfer has completed. In any event the frame is marked as busy to prevent it from being used for another purpose. As soon as the page frame is clean the operating system looks up the disk address

where the
bring it i
process is
one is av
the page
its positi
state. Th
it had w
point to
and the
languag
other
space to

31. ◆

where the needed page is and schedules a disk operation to bring it in. While the page is being loaded the faulting process is still suspended and another user process is run if one is available. When the disk interrupt indicates that the page has arrived the page tables are updated to reflect its position and the frame is marked as being in normal state. The faulting instruction is backed up to the state it had when it began and the program counter is reset to point to that instruction. The faulting process is scheduled and the operating system returns to the assembly language routine that called it. This routine registers and other volatile information and returns to user space to continue execution as if no fault had occurred.



(Figure : The Steps in Handling a Page Fault)

31. ◆ ***Given memory partition of 100 kb, 500 kb, 200 kb, 300 and 600 kb (in order). How would each of the best fit. first fit and worst fit algo place processes of 210 kb, 410 kb, 110 kb and 420 kb (in order)?*** **(2017)**

◆ ***Which Algorithm makes the most efficient use of memory?***

First – Fit :

210K is put in 500K partition

410K is put in 600K partition

110K is put in 288K partition (new partition 290 K = 500K - 210K)
420K must wait

Best – Fit :
210K is put in 300K partition
410K is put in 500K partition
110K is put in 200K partition
420K is put in 600K partition

Worst – Fit :
210K is put in 600K partition
410K is put in 500K partition
110K is put in 388K partition
420K must wait
best-fit turns out to be the best.

32. *When is the need for page replacement occurs? Consider the following reference strings :*
1, 2, 3, 4, 2, 1, 6, 5, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6
How many Page faults will occur for: (2017)
(1) *LRU replacement*
(2) *FIFO replacement*
Assuming 5 frames.
Note : Initially all frames are empty.

LRU : 1 2 3 4 2 1 6 5 2 1 2 3 7 6 3 2 1 2 3 6
. . . .
8 faults

FIFO : 1 2 3 4 2 1 6 5 2 1 2 3 7 6 3 2 1 2 3 6
. . . .
10 faults

33. *Consider the following page reference string :*
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6
How many page faults would occur for the following page replacement algorithm? Assume that are frames are initially empty (Assume frame size is four) : (2016)
(1) *LRU replacement*
(2) *FIFO replacement*
(3) *Optimal replacement*

4 Frames
LRU :
1 2 3 4
✓ ✓ ✓ ✓
10 faults
FIFO :
1 2 3
✓ ✓ ✓
14 faults
Optimal
1 2 3
✓ ✓ ✓
8 faults

34. *Consi*
words
frame
addre
addr

A
becaus
of 8 =
bits.
physi

P

35. ◆
◆

De
(1)

(2)

4 Frames

LRU :

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
✓	✓	✓	✓			✓	✓				✓	✓	✓			✓			

10 faults

FIFO :

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
✓	✓	✓	✓			✓	✓	✓	✓		✓	✓	✓		✓	✓		✓	

14 faults

Optimal :

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
✓	✓	✓	✓			✓	✓					✓				✓			

8 faults

34. ***Consider a logical address space of 8 pages of 1024 words each, mapped on to a physical memory of 32 frames. Find out the number of bits in the logical address and the number of bits on the physical address.*** **(2016)**

Addressing within a 1024-word page requires 10 bits because $1024 = 2^{10}$. Since the logical address space consists of $8 = 2^3$ pages, the logical addresses must be $10 + 3 = 13$ bits. Similarly, since there are $32 = 2^5$ physical pages, physical addresses are $5 + 10 = 15$ bits long.

Physical Address (P = page number bits)

P	P	P	P	P	-	-	-	-	-	-	-	-	-	-

Logical Address (P = page number bits)

P	P	P	-	-	-	-	-	-	-	-	-	-	-

35. ◆ ***What are the various operating system design issues? Explain.*** **(2016)**

◆ ***Discuss the design goals of a distributed system.***

Definition of a Distributed System

(1) A distributed system is a collection of independent computers that appear to its users as a single coherent system :

(a) Multiple connected CPUs working together.

(b) A collection of independent computers that appear to its users as a single coherent system.

(2) **Examples :** Parallel machines, networked machines.

Distributed Systems Models :

(1) **Minicomputer Model (Example, Early Networks) :**
- (a) Each user has local machine.
- (b) Local processing but can fetch remote data (files, databases).

(2) **Workstation Model (Example, Sprite) :** Processing can also migrate.

(3) **Client-Server Model (Example, *V* System, World Wide Web) :**
- (a) User has local workstation.
- (b) Powerful workstations serve as servers (file, print, DB servers).

(4) **Processor Pool Model (Example, Amoeba, Plan 9) :**
- (a) Terminals are *X* terms or diskless terminals.
- (b) Pool of backend processors handle processing.

Design Goals Distributed System :

(1) Access and share remote resources
(2) Interoperability
(3) Portability
(4) Flexibility
(5) Transparency
(6) Scalability

Access and Share Remote Resources : By using the combined processing and storage capacity of many nodes, performance levels can be reached that are beyond the range of centralized machines.

Flexibility :

(1) Failure of a single machine no longer halts everyone.
(2) Generally graceful degradation of the overall system's resources.
(3) Ability to apply fault tolerance for important tasks at a high architectural level.

Transparency in a Distributed System : Different forms of transparency in a distributed system.

Transparency	Description
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource may be replicated
Concurrency	Hide that a resource may be shared by several competitive users

Failure
Persistence

Scalability
(1) Size
(2) Geogr
(3) Admi

Scalabilit

Conce
Centralized s
Centralized c
Centralized a

Decentra
(1) Non state
(2) Mac
(3) Fail
(4) The

(a)

(b)
(c)

36. *What and ti*

Advan Syster
(1) It
(2) It
(3) It

Advan
(1) P
(2) A
(3) R

37. *Supp* 1, 2,
(1)
(2)

Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Scalability :

(1) Size
(2) Geography
(3) Administrative organizations

Scalability Problems in a Distributed System :

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Decentralized Algorithms Characteristics :

(1) None has complete information about the system state.
(2) Machines take decisions on local info.
(3) Failure of one machine doesn't affect the algorithm.
(4) There is no assumption about a global clock.

To solve scalability problems :

(a) Hiding communication latencies : Asynchronous communications (but not only, not always).
(b) Distribution.
(c) Replication (with care for consistency).

36. ***What are the advantages of multiprogramming and time sharing operating system?*** **(2015)**

Advantages of Multiprogramming Operating System

(1) It increases CPU utilization.
(2) It decreases total read time needed to execute a job.
(3) It maximizes the total job throughput of a computer.

Advantages of Timesharing Operating Systems :

(1) Provide advantage of quick response.
(2) Avoids duplication of software.
(3) Reduces CPU idle time.

37. ***Suppose reference strings are as given below :***
1, 2, 4, 3, 6, 7, 2, 3, 4, 3, 1, 3, 4 **(2015)**
(1) ***Explain FIFO replacement with 4 frames.***
(2) ***Explain LRU with 4 frames.***

(1) Given String : 1, 2, 4, 3, 6, 7, 2, 3, 4, 3, 1, 3, 4

FIFO													
0	1	1	1	1	6	6	6	6	6	3	3	3	3.
1		2	2	2	2	7	7	7	7	7	1	1	1
2			4	4	4	4	2	2	2	2	2	2	2
3				3	3	3	3	3	4	4	4	4	4
Page Fault	*	*	*	*	*	*	*		*	*	*		

(Ans.)

Total Fault = 10

(2)

LRU													
0	1	1	1	1	6	7	7	7	7	7	7	7	7
1		2	2	2	2	2	2	2	2	2	1	1	1
2			4	4	4	4	4	4	4	4	4	4	4
3				3	3	3	3	3	3	3	3	3	3
Page Fault	*	*	*	*	*	*					*		

(Ans.)

Total Fault = 7

38. *Given a system with four page frames, the following table indicates page, load time, last reference time, dirty bit and reference bit.* (2014)

Page	Load time	Last reference	Dirt bit	Reference bit
0	167	374	1	1
1	321	321	0	0
2	254	306	1	0
3	154	331	0	1

(1) *Which page will FIFO replace?*

(2) *Which page will LRU replace?*

(1) 3

FIFO selects the page loaded first, page 3 at time 154.

(2) 2

LRU selects the page referenced last page 2 at time 306.

❑❑

PROCE

1. *Distinguish preemptive*

Difference Scheduling

Aspect	
Definition	
Priority Adjustment	
Context Switching	
Response Time	
Real-Time Systems	
Example	

2. *Write th and shor*

3, 4

3	3	3.
1	1	1
2	2	2
4	4	4

*

(Ans.)

7	7	7
1	1	1
4	4	4
3	3	3

*

(Ans.)

imes, the time, last
(2014)

erence bit
1
0
0
1

3 at time

2 at time

PROCESSES / CPU SCHEDULING

1. ***Distinguish between preemptive and non preemptive Scheduling.*** **(2023-24)**

Difference between Preemptive and Non-preemptive Scheduling

Aspect	Preemptive Scheduling	Non-Preemptive Scheduling
Definition	Allows a higher priority task to interrupt the execution of a lower priority task.	Does not allow interruption of the currently executing task until it completes its execution.
Priority Adjustment	Allows for dynamic adjustment of priorities.	Priorities are fixed and cannot be changed during execution.
	Tasks with higher priority can be scheduled to run at any time, preempting lower priority tasks.	Tasks are executed until completion without interruption based on their arrival order and priority.
Context Switching	Frequent context switching may occur, leading to overhead and potentially slower performance.	Context switching occurs less frequently as tasks run to completion, reducing overhead.
Response Time	Generally shorter response times for high priority tasks due to preemption.	May have longer response times as tasks may not be interrupted until they finish execution.
Real-Time Systems	Widely used in real-time systems where timely response to events is critical.	Less commonly used in real-time systems where predictability and determinism are important.
Example	Operating systems with preemptive scheduling (e.g., Linux, Windows).	Batch processing systems, simple embedded systems where responsiveness is not critical.

2. ***Write the difference between long term scheduler and short term scheduler.*** **(2023-24)**

Difference between Long term Scheduler and Short Term Scheduler

Long-Term Scheduler	Short-Term Scheduler
Long-Term Scheduler takes the process from job pool.	Short-Term Scheduler takes the process from ready queue.
Long-Term Scheduler is also known as Job Scheduler.	Short-Term Scheduler is also known as CPU Scheduler.
In Long-Term Scheduler, the programs are setup in the queue and as per the requirement the best one job is selected.	In Short-Term Scheduler no such queue is exist.
It regulates the more DOM (Degree of Multi-programming).	It regulates the less DOM (Degree of Multi-programming)
It regulates the programs which are selected to system for processing.	It ensures which program is suitable or important for processing.
Speed is less than the short-term scheduler.	Speed is very fast as compared to long-term scheduler.
Long-Term Scheduler changes the process state from New to Ready.	Short-Term Scheduler changes the process state from Ready to Running.
Time-sharing operating systems have no long-term scheduler.	It may be minimal in time-sharing system.
It select a good process, mix of I/O bound and CPU bound.	It select a new process for a CPU quite frequently.
Long-Term Scheduler control Multi-Programming	Short-Term Schedule control Multitasking

3. ***Calculate Average waiting time by using shortest Remaining Time First (SRTF) algorithm.***

Process No.	A.T.	B.T.
P1	0	2
P2	1	5
P3	2	1
P4	3	2
P5	3	3
P6	6	6

(2023-24)

Solution :

To cal
Shortest Re
to simulate
waiting tim

Here
solving the

(1) Arran
(2) Start
time.
(3) Execu
until
(pree
(4) Repe
(5) Arri

Executio

At t
At t
At t
At t
At t
At

Waiting

P1 : Wa
Time = (
P2 : Wa
Time =
P3 : Wa
Time =
P4 : W
Time =
P5 : W
Time =
P6 : W
Time =

Avera

A
13 / 6 =
S

Remai
units.

Solution :

To calculate the average waiting time using the Shortest Remaining Time First (SRTF) algorithm, we need to simulate the execution of the processes and track the waiting time for each process.

Here are some steps that we should follow while solving the question :

(1) Arrange the processes based on their arrival time.

(2) Start with the process that has the shortest burst time.

(3) Execute each process for the required burst time or until a shorter burst time process arrives (preemption).

(4) Repeat the process until all processes are completed.

(5) Arrival Time Sorted Queue: P1, P2, P3, P4, P5, P6

Execution Timeline :

At time 0, P1 arrives and executes for 2 units.
At time 2, P3 arrives and executes for 1 unit.
At time 3, P4 arrives and executes for 2 units.
At time 5, P2 arrives and executes for 3 units.
At time 8, P5 arrives and executes for 3 units.
At time 11, P6 arrives and executes for 6 units.

Waiting Time Calculation :

P1 : Waiting Time = 0 (Process starts at arrival) - Arrival Time = 0

P2 : Waiting Time = 5 (Process starts at time 5) - Arrival Time = 4

P3 : Waiting Time = 2 (Process starts at time 2) - Arrival Time = 0

P4 : Waiting Time = 5 (Process starts at time 5) - Arrival Time = 2

P5 : Waiting Time = 5 (Process starts at time 5) - Arrival Time = 2

P6 : Waiting Time = 5 (Process starts at time 11) - Arrival Time = 5

Average Waiting Time Calculation :

Average Waiting Time = (0 + 4 + 0 + 2 + 2 + 5) / 6 = 13 / 6 = 2.167

So, the average waiting time using the Shortest Remaining Time First (SRTF) algorithm is 2.167 time units.

4. ***Given the following queue – 95, 180, 34, 119, 11, 123, 62, 64 with the Red – write head initially at the track 50 and the tail track being at 199. Calculate the Total Head movements by using :***
(a) First Come First Serve (FCFS)
(b) Shortest Seek Time First (SSTF) **(2023-24)**

(1) First Come First Serve (FCFS) :

Now, this algorithm is fairly simple. In the order the request would come, in the same order visit the address on the disk.

Example :

Track Range from 0 to 199 and head initially is rested on 50

95, 180, 34, 119, 11, 123, 62, 64

First the head from 50 goes to 95 then→95→180→34→119→11→123→62→64

Total Head Movement Computation : (THM)

(95 – 50) + (180 – 95) + (180-34) + (119-34) + (119-11) + (123-11) + (123-62) + (64-62)

Method :

Now rather than doing (95 – 50) + (180 – 95) since direction doesn't change we can do (180 – 50). The above step will save a lot of your time.

= (180 – 50) + (180-34) + (119-34) + (119-11) + (123-11) + (123-62) + (64-62) = 130 + 146 + 85 + 108 + 112 + 61 + 2 (THM) = 644 tracks

Assuming a seek rate of 5 milliseconds is given, we compute for the seek time using the formula:

Seek Time = THM * Seek rate = 644 * 5 ms

Seek Time = 3,220 ms

Disk Scheduling Algorithm FCFS

(2) Shortest Seek Time First (SSTF) : Now, even in SSTF to and fro movement is there a lot. Wouldn't it be better if we just moved in one direction and once all requests in that direction are complete changed the direction and moved to 2nd direction.

Note : If in the question the initial direction of movement is given then, Follow that, if the direction is not given then, the whichever is the nearest end that direction is chosen.

For Exa
initially i
So direct
Exampl
Trac
rested on
95, 1
(THM
Seek Tim
Seek Tim

5. ◆ ***Explain critical***
◆ ***What d problem by its so***
◆ ***What is synchr***
◆ ***Discuss proble***
◆ ***What d***

Critical Se

(1) A critic segmen interfe exclusi section their proced design this ru

(2) Beside critica requir

(3) Progre sectior enter t are no partici critica

(4) Bound numbe

For Example : In this question the Read-write head initially is at 50. The nearest b/w 0 or 199 is 0.
So direction of movement will be that.

Example :

Track Range from 0 to 199 and head initially is rested on 50

95, 180, 34, 119, 11, 123, 62, 64

(THM) = (50 – 0) + (180 – 0) = 230

Seek Time = THM * Seek rate = 230 * 5ms

Seek Time = 1150 ms

5. ◆ ***Explain three requirements that a solution to critical-section problem must satisfy.*** **(2022-23)**
◆ ***What do you understand by Critical Section problem? What requirements must be satisfied by its solution?*** **(2019)**
◆ ***What is the role of critical section in process synchronization?*** **(2018)**
◆ ***Discuss the characteristics of a critical section problem.*** **(2016)**
◆ ***What do you understand by critical section?***

Critical Sections

(1) A critical section is part of the program (which is a segment of code) that must be protected from interference. This is usually resolved by mutual exclusion (if one process is executing in its critical section, then no other processes can be executing in their critical sections). A protocol (a standard procedure for regulating data transmission) must be designed that the processes can use to cooperate with this rule.

(2) Besides mutual exclusion, a solution to the critical sections problem must satisfy two other requirements :

(3) Progress - if no process is executing in its critical section and there exists some processes that wish to enter their critical sections, only those processes that are not executing in their remainder section can participate in the decision of which will enter its critical section next

(4) Bounded waiting - there exists a bound on the number of times that other processes are allowed to

enter the critical sections after a process has made a request to enter its critical section and before the request is granted

(5) An example of a critical section problem is the two process solution. It involves three algorithms. The first approach is to let the processes share a common variable initialized to zero or one. This ensures that only one process at a time is in its critical section.

But it does not satisfy the progress requirement because it requires the rotation of processes in executing the critical section. If, say, the first process sets the variable equal to zero, it would not be able to enter its critical section, not even if the second process is in its remainder section.

(6) The first approach does not retain sufficient information about the state of each process. In the second algorithm, we can replace the variable with an array. We can initialize the elements to false. If the flag (a response from a process) changes to true, this indicates that a process is ready to enter its critical section. (NOTE: If by any chance a process were already in its critical section, then the other would wait until it was finished.)

When the process exits its critical section, the flag changes to false. Mutual exclusion is satisfied but the progress requirement still is not. If both flags were set to true, both processes would loop forever in their while statements.

(7) Combining the two algorithms, we derive a solution where the processes share two variables. The initial flag is set to false. To enter the critical section, the first process sets its flag as true, which prompts the other process to enter. If both processes try to enter at once, the variable will be set at the same time, but one of the assignments will be overwritten.

Since one process does not change the value of the variable while executing the while statement, mutual exclusion, progress and bounded waiting are all satisfied.

Solution to the Critical Section Problem

The critical section problem needs a solution to synchronize the different processes. The solution to the

critical secti
conditions:
(1) **Mutual**
only on
any tim
section,
(2) **Progre**
using t
other
process
(3) **Boun**
each
should
sectio

6. ◆ ***Diffe***
preen
◆ ***Expl***
sche

Preempti
prioritized
the proces
Non – Pr
the state
from the

7. ◆ ***Wh***
Wh
◆ ***Wh***
sta
◆ ***Dr***
(P
◆ ***W***
of
◆ ***Di***
pr
◆ ***W***
or

In
progra

critical section problem must satisfy the following conditions:

(1) **Mutual Exclusion :** Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

(2) **Progress :** Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.

(3) **Bounded Waiting :** Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

6. ◆ ***Differentiate between pre-emptive and Non-preemptive Scheduling.*** **(2022-23)**
◆ ***Explain pre – emptive and non – preemptive scheduling.*** **(2015)**

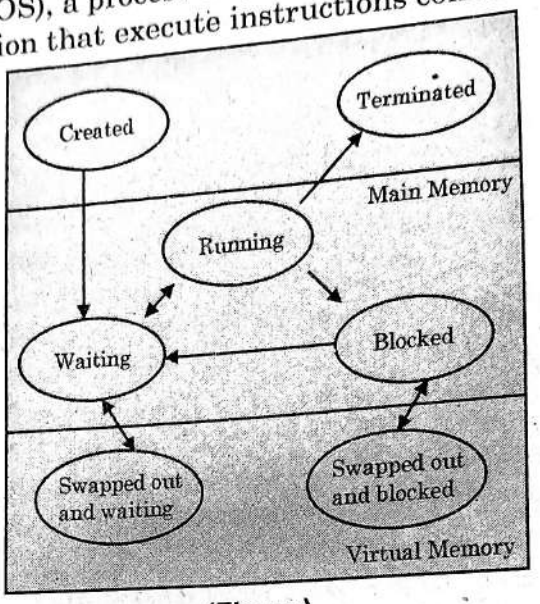
Preemptive Scheduling : The preemptive scheduling is prioritized. The highest priority process should always be the process that is currently utilized.

Non – Preemptive Scheduling : When a process enters the state of running, the state of that process is not deleted from the scheduler until it finishes its service time.

7. ◆ ***What do you mean by PCB? Where is it used? What are its contents? Explain.*** **(2022-23)**
◆ ***What is a process? Draw and explain process state diagram.*** **(2022-23)**
◆ ***Draw and explain Process Control Block (PCB).*** **(2018)**
◆ ***What is a process? What are the different states of a process? What is PCB?*** **(2017)**
◆ ***Discuss the various states of process with process state diagram.*** **(2016)**
◆ ***What is process? What causes process change one state to other? Show schematically.*** **(2015)**

In computing, a process is an instance of a computer program that is being executed. It contains the program

code and its current activity. Depending on the Operating System (OS), a process may be made up of multiple threads of execution that execute instructions concurrently.



(Figure)

A computer program is a passive collection of instructions; a process is the actual execution of those instructions. Several processes may be associated with the same program; for example, opening up several instances of the same program often means more than one process is being executed.

Primary Process States : The following typical process states are possible on computer systems of all kinds. In most of these states, processes are "stored" on main memory.

Created : (Also called New) When a process is first created, it occupies the "created" or "new" state. In this state, the process awaits admission to the "ready" state. This admission will be approved or delayed by a long-term, or admission, scheduler.

Typically in most desktop computer systems, this admission will be approved automatically, however for real-time operating systems this admission may be delayed. In a real time system, admitting too many processes to the "ready" state may lead to over saturation and over contention for the systems resources, leading to an inability to meet process deadlines.

ng on the Operating
p of multiple threads
oncurrently.

ssive collection of
execution of those
associated with the
p several instances
than one process is

typical process
of all kinds. In
tored" on main

process is first
ew" state. In this
he "ready" state.
d by a long-term,

er systems, this
lly, however for
ission may be
ting too many
over saturation
rces, leading to

Ready or Running : (Also called waiting or runnable) A "ready" or "waiting" process has been loaded into main memory and is awaiting execution on a CPU (to be context switched onto the CPU by the dispatcher, or short-term scheduler). There may be many "ready" processes at any one point of the systems execution - for example, in a one processor system, only one process can be executing at any one time, and all other "concurrently executing" processes will be waiting for execution. A ready queue is used in computer scheduling. Modern computers are capable of running many different programs or processes at the same time. However, the CPU is only capable of handling one process at a time. Processes that are ready for the CPU are kept in a queue for "ready" processes. Other processes that are waiting for an event to occur, such as loading information from a hard drive or waiting on an internet connection, are not in the ready queue.

Blocked : A process that is waiting for some event (such as I/O operation completion or a signal).

Terminated : A process may be terminated, either from the "running" state by completing its execution or by explicitly being killed. In either of these cases, the process moves to the "terminated" state. If a process is not removed from memory after entering this state, this state may also be called zombie.

Additional Process States : Two additional states are available for processes in systems that support virtual memory. In both of these states, processes are "stored" on secondary memory (typically a hard disk).

Swapped Out and Waiting : (Also called suspended and waiting). In systems that support virtual memory, a process may be swapped out, that is removed from main memory and placed in virtual memory by the mid-term scheduler. From here the process may be swapped back into the waiting state.

Swapped Out and Blocked : (Also called suspended and blocked). Processes that are blocked may also be swapped out. In this event the process is both swapped out and blocked, and may be swapped back in again under the same circumstances as a swapped out and waiting process (although in this case, the process will move to the blocked state, and may still be waiting for a resource to become

available). PCB contains information associated with each process like :

(1) Process state
(2) Program counter
(3) CPU registers
(4) CPU scheduling information
(5) Memory-management information
(6) Accounting information
(7) I/O status information.

Process Control Box : Each process is represented in the operating by a process control Block (PCB) also called a task control block. A PCB contains many pieces of information associated with a specific process, including these :

Process Id
Process State
Program Priority
Registers
Pointers
List of Open Files
. . .

Process Control Block

A PCB is created when a user creates a process and it is removed from the system when the process is killed. Each user has a PCB. All PCBs are kept in the memory reserved for OS. A PCB contains following fields :

(1) **Process Id :** This is a number allocated by the OS to the process on creation. The OS starts allocating PIDS from number 0. The next PID as 1 and so on. This continuous till $n - 1$.

(2) **Process State :** Different states of a process such as ready, running, suspended etc are kept in coded form in this field.

(3) **Process Priority :** In some cases a few processes are urgently required to be completed than others. This priority can be set externally by the system manager or it can be decided by the operating system internally depending on various parameters. PCP contains value of the priority for the process.

(4) **Register Save Area :** This is needed to save all the CPU registers at the context switch.

(5) **Pointers to the Process's Memory :** It gives direct or indirect addresses of pointers to the locations where the process image resides in the memory.

(6) **Pointers**
other dat
(7) **List of**
close all
terminat
(8) **Accoun**
the usag
time, dis
(9) **Other**
the BFD
(10) **Pointe**
next PC

8. ◆ ***Discus***
schedu
◆ ***Define***
respon
algori
◆ ***What***
the CI

CPU
selects fro
allocates th
may take p
(1) Switc
(2) Switc
(3) Switc
(4) Term
Crite
(a)
(b)
(c)
(d)
(e)

(6) **Pointers to Other Resources :** It gives pointers to other data structures maintained for that process.

(7) **List of Open File :** Operating system can use it to close all open files not closed by a process explicitly on termination.

(8) **Accounting Information :** This gives the account of the usage of resources such as CPU time, connect time, disk input/output used etc. by the process.

(9) **Other Information :** It contains the path name or the BFD number of the current directory.

(10) **Pointers to Other PCBs :** It gives the address of the next PCB within a specific category.

8. ◆ ***Discuss the properties of different CPU scheduling algorithms.*** **(2022-23)**

◆ ***Define throughput, CPU utilization and response time criteria of any CPU scheduling algorithm.*** **(2017)**

◆ ***What are the main criteria used for comparing the CPU scheduling algorithm? Explain.*** **(May 2013, 2014)**

CPU scheduler (also called short-term scheduler) selects from among the processes in ready queue, and allocates the CPU to one of them. CPU scheduling decision may take place when a process is :

(1) Switches from running to waiting state.

(2) Switches from running to ready state.

(3) Switches from waiting to ready.

(4) Terminates.

Criteria for comparing CPU-scheduling algorithms :

(a) **CPU Utilization :** Keep the CPU as busy as possible (0 to 100% efficiency).

(b) **Throughput :** # of processes that complete their execution per time unit.

(c) **Turnaround Time :** Amount of time to execute a particular process.

(d) **Waiting for Memory :** Ready queue + execution + I/O

(e) **Waiting Time :** Amount of time a process waits in the ready queue..

(f) **Response Time :** Amount of time it takes fro when a request was submitted until the fir response is produced.

9. *Consider the following set of processes with the length of the CPU burst given in milliseconds :* (2019)

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

The processes are assumed to have arrived in the order, P_1, P_2, P_3, P_4, P_5, all at time O.

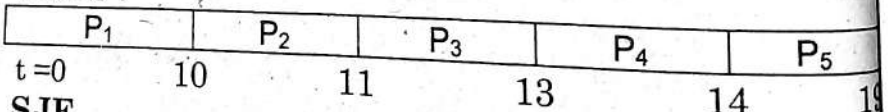
(1) *Draw four Gantt Charts that illustrate the execution of these processes using the following scheduling algorithms : (FCFS, SJF, non – pre – emptive priority (taking smaller priority number implies a higher priority) and RR (quantum = 1).*

(2) *What is the turnaround time of each process for each of the scheduling algorithms in part (i)?*

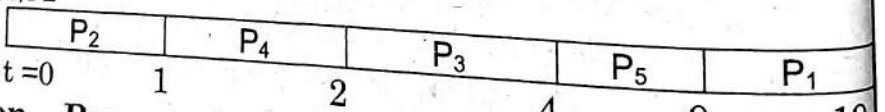
(3) *What is the waiting time of each process for each of the scheduling algorithm in part (i)?*

(4) *Which of the Algorithm in part (i) a results in the minimum average waiting time (over all processes)?*

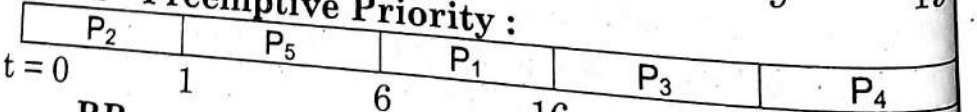
(1) **FCFS :**



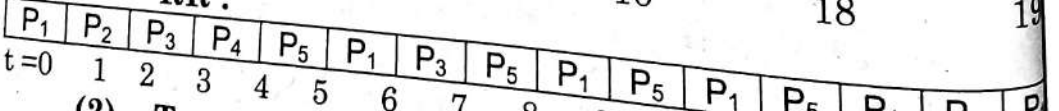
SJF



Non – Preemptive Priority :



RR :



(2) **Turnaround Time (TAT) :**

In case FCFS
TAT of $P_1 = 10$
$P_2 = 11$
$P_3 = 13$
$P_4 = 14$
$P_5 = 19$
In case SJF
TAT of $P_1 = 19$
$P_2 = 1$
$P_3 = 4$
$P_4 = 2$
$P_5 = 9$
In case Non Pre-emptive Priority
TAT of $P_1 = 16$
$P_2 = 1$
$P_3 = 18$
$P_4 = 19$
$P_5 = 6$
In case RR
TAT of $P_1 = 1 + 5 + 3 + 2 + 2 + 6 = 19$
$P_2 = 2$
$P_3 = 3 + 4 = 7$
$P_4 = 4$
$P_5 = 5 + 3 + 2 + 2 + 2 = 14$

(3) Waiting Time (WT) :

WT = TAT – CPU time
In case FCFS
WT of $P_1 = 0$
$P_2 = 10$
$P_3 = 11$
$P_4 = 13$
$P_5 = 14$
In case SJF
WT of $P_1 = 9$
$P_2 = 0$
$P_3 = 2$
$P_4 = 1$
$P_5 = 4$
In case Priority
WT of $P_1 = 6$
$P_2 = 0$

$P_3 = 16$
$P_4 = 18$
$P_5 = 1$

In case RR
WT of $P_1 = 9$
$P_2 = 1$
$P_3 = 5$
$P_4 = 3$
$P_5 = 9$

(Ans.)

(4) Minimum Avg Waiting Time (MAW-T)

In case FCFS

$$\text{AWT} = \frac{0+10+11+13+14}{5} = 9.6$$

$$\text{In case SJF} = \frac{9+0+2+1+4}{5} = 3.2$$

$$\text{In case Priority} = \frac{6+0+16+18+1}{5} = 8.2$$

$$\text{In case RR} = \frac{9+1+5+3+9}{5} = 5.4$$

In case of SJF, the minimum avg waiting time is 3.2

(Ans.)

10. ***What do you understand by synchronization Hardware?*** **(2019)**

Synchronization Hardware

Many systems provide hardware support for critical section code. The critical section problem could be solved easily in a single-processor environment if we could disallow interrupts to occur while a shared variable or resource is being modified.

In this manner, we could be sure that the current sequence of instructions would be allowed to execute in order without pre-emption. Unfortunately, this solution is not feasible in a multiprocessor environment.

Disabling interrupt on a multiprocessor environment can be time consuming as the message is passed to all the processors.

This message transmission lag, delays entry of threads into critical section and the system efficiency decreases.

11. ◆ ***The long – term scheduler controls the degree of multiprogramming. Justify by giving examples. Also discuss the need of mid- term scheduler.*** **(2019)**

◆ ***What are various types of CPU schedulers? Describe.*** **(2015)**

Schedulers are special system software's which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

Schedulers are of three types :

(1) Long Term Scheduler
(2) Short Term Scheduler
(3) Medium Term Scheduler

(1) **Long Term Scheduler :** It is also called job scheduler. Long term scheduler determines which programs are admitted to the system for processing. Job scheduler selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling. The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When process changes the state from new to ready, then there is use of long term scheduler.

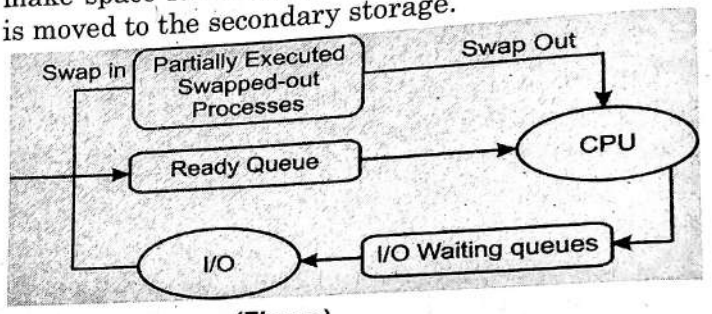
(2) **Short Term Scheduler :** It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.

Short term scheduler also known as dispatcher, execute most frequently and makes the fine grained

decision of which process to execute next. Short term scheduler is faster than long term scheduler.

(3) **Medium Term Scheduler :** Medium term scheduling is part of the swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium term scheduler is in-charge of handling the swapped out-processes.

Running process may become suspended if it makes an I/O request. Suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other process, the suspended process is moved to the secondary storage.



(Figure)

This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

12. ◆ ***What is the Dining – Philosophers problem? Suggest its solution.*** **(2019)**
◆ ***Explain Dining philosopher's problem.*** **(2014)**

Dining Philosophers

There is a dining room containing a circular table with five chairs. At each chair is a plate, and between each plate is a single chopstick. In the middle of the table is a bowl of spaghetti.

Near the room are five philosophers who spend most of their time thinking, but who occasionally get hungry and need to eat so they can think some more. In order to eat, a philosopher must sit at the table, pick up the two chopsticks to the left and right of a plate, then serve and eat the spaghetti on the plate. Thus, each philosopher is represented by the following pseudo code :

process P[i]
while true d
{ THINK;
PICKUP(CH
EAT;
PUTDOWN
}

A philos
philosopher wh
may PICKUP
order, or nor
actions, and,
single CHOPS
design a prot
philosopher, w

13. ***Consider t
arrival tim***

Job
A
B
C
D
E

***Calculate
preemptive***

FCFS :

A	

$t=0$ 5

W. T. of
W. T of
W. T of
W. T of
W. T of

$A\ W.\ T = \frac{0+}{}$

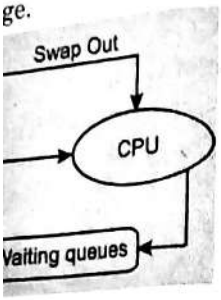
Preemptive

A	C

$t = 0$ 5

W. T of

ecute next. Short term
rm scheduler.
r : Medium term
pping. It removes the
; reduces the degree of
um term scheduler is
ped out-processes.
come suspended if it
nded processes cannot
completion. In this
ess from memory and
the suspended process
ge.



, a the process is
ing may be

ers problem?
(2019)
(2014)

circular table
and between each
le of the table is a

rs who spend most
ally get hungry and
. In order to eat, a
pick up the two
te, then serve and
ach philosopher is
:


```
process P[i]
while true do
{ THINK;
PICKUP(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
EAT;
PUTDOWN(CHOPSTICK[i], CHOPSTICK[i+1 mod 5])
}
```

A philosopher may THINK indefinitely. Every philosopher who EATs will eventually finish. Philosophers may PICKUP and PUTDOWN their chopsticks in either order, or non-deterministically, but these are atomic actions, and, of course, two philosophers cannot use a single CHOPSTICK at the same time. The problem is to design a protocol to satisfy the liveness condition: any philosopher, who tries to EAT, eventually does.

13. ***Consider the following set of jobs with their arrival time, execution time (in minutes) :*** **(2018)**

Job	Arrival Time	Execution Time
A	0	5
B	1	15
C	3	12
D	7	25
E	10	5

Calculate the average waiting time for FCFS, SJF preemptive, SJF non – preemptive.

FCFS :

A	B	C	D	E

$t=0$ 5 20 32 57 62

W. T. of $A=0$

W. T of $B=5-1=4$

W. T of $C=20-3=17$

W. T of $D=32-7=25$

W. T of $E=57-10=47$

$$A\ W.\ T = \frac{0+4+17+25+47}{5} = \frac{93}{5} = 18.6 \text{ min}$$

Preemptive SJF :

A	C	E	C	B	D

$t=0$ 5 10 15 22 37 62

W. T of $A=0$

W. T of $B = 22 - 1 = 21$

W. T of $C = (5-3) + (15-10) = 2+5 = 7$

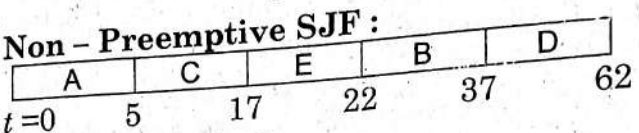
W. T of $D = 37 - 7 = 30$

W. T of $E = 10 - 10 = 0$

$$A.\ W.\ T. = \frac{0+21+7+30+0}{5}$$

$$= \frac{58}{5} = 11.6 \text{ min}$$

Non – Preemptive SJF :



W. T of $A = 0$

W. T of $B = 22 - 1 = 21$

W. T of $C = 5 - 3 = 2$

W. T of $D = 37 - 7 = 30$

W. T of $E = 17 - 10 = 7$

$$A.\ W.\ T = \frac{0+21+2+30+7}{5}$$

$$= \frac{60}{5} = 12 \text{ min}$$

14. *What is semaphore? Describe.* **(2017)**

A semaphore is a protected variable or abstract data type which constitutes the classic method for restricting access to shared resources such as shared memory in a parallel programming environment.

Example : Weak, Busy-Wait Semaphore :

(1) The simplest way to implement semaphores.

(2) Useful when critical sections last for a short time, or we have lots of CPUs.

(3) S initialized to positive value (to allow someone in at the beginning).

(4) S is an integer variable that, apart from initialization, can only be accessed through two atomic and mutually exclusive operations :

```
wait(s) :
while (s.value != 0);
s.value--;
```

```
signal(s):
s.value++;
```

All happen automatically i.e. wrap pre and post protocols.

15. ***Draw resource allocation graphs for three processes P_1, P_2, P_3 and P_4 resources R_1, R_2, R_3, R_4 with one, two, three, four instances respectively.*** **(2017)**

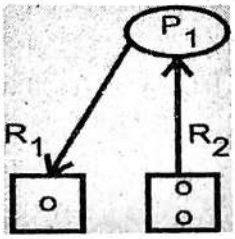
Process States Status :

(1) ***Process P_1 is holding an instance of resource type R_2 and waiting for an instance of R_1.***

(2) ***Process P_2 is holding an instance of R_1 and waiting for R_3.***

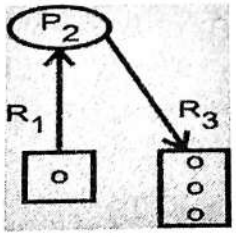
(3) ***Process P_3 in holding an instance of R_3.***

(1)



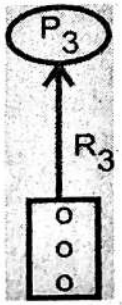
(Figure)

(2)



(Figure)

(3)



(Figure)

16. ◆ ***Discuss swapping.*** **(2014, 2017)**

◆ ***What is the significance of swapping technique in operating system?*** **(2016)**

◆ ***What is Swapping and how can it be performed?***

Swapping

A process needs to be in memory to be executed. A process however can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution as a multiprogramming environment with a round robin CPU-scheduling algorithm. After expiring the time quantum, the memory manager starts to swap out the process that just finished and to swap in another process to the memory space that has been freed. In the meantime, the CPU scheduler will allocate a time slice to some other process in memory. A variant of this swapping policy is used for priority based scheduling algorithms.

If higher priority process arrives and wants service, the memory manager swaps out the lower priority processes, so that it can load and execute the higher-priority process. When the higher-priority process finishes, the lower-priority process can be swapped back in the continued. This is called as roll-out and roll-in. Swapping requires a backing store. The backing store is commonly a fast disk and large enough to accommodate copies of all memory images for all users and directly accessed, for the purpose system maintains a ready queue consisting of all processes whose memory images are on the backing store or in memory to ready to run. When the CPU scheduler decides to execute a process, it calls the dispatcher to see whether the next process in the queue is in memory. If not, and there is no free memory region, the dispatcher swaps out a process currently in memory and swaps in the desired process. It then reloads registers as normal and transfers control to the selected process. Context switching takes place very highly.

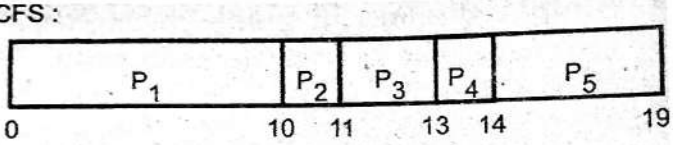
In swapping, the total transfer time is directly proportional to the amount of memory swapped. Swapping is constrained by other factors also as processes to be swapped should be surely idle. Generally, swap space is allocated as a chunk of disk, separate from the file system, so that its use as fast as possible.

17. *Draw Gantt chart for the following situation :* **(2017)**

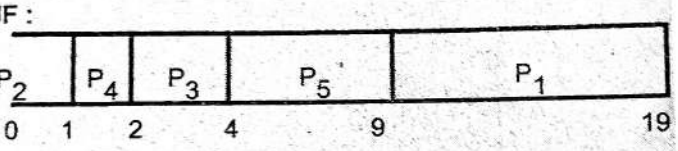
Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes are answered to have arrived in order P1, P2, P3, P4, P5 all at time = 0. Find average waiting time using FCFS, SJF, non – preemptive priority. (Small priority number means highest priority).

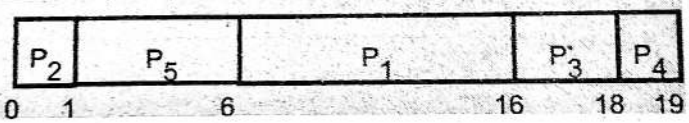
For FCFS :



For SJF :



For Non Premeptive priority Scheduling :



For FCFS :

WT for P_1 = 0 ms WT for P_2 = 10 ms WT for P_3 = 11 ms

WT for P_4 = 13 ms WT for P_5 = 14 ms

Average WT = (0 + 10 + 11 + 13 + 14) / 5 = 48 / 5 = 9.6 ms

For SJF :

WT for P_1 = 9 ms WT for P_2 = 0 ms WT for P_3 = 2 ms

WT for P_4 = 1 ms WT for P_5 = 4 ms

Average WT = (9 + 0 + 2 + 1 + 4) / 5 = 16 / 5 = 3.2 ms

For Non Preemptive Priority :

WT for P_1 = 6 ms WT for P_2 = 0 ms WT for P_3 = 16 ms

WT for P_4 = 18 ms WT for P_5 = 1 ms

Average WT = (6 + 0 + 16 + 18 + 1) / 5 = 41 / 5 = 8.2 ms

[B.22]

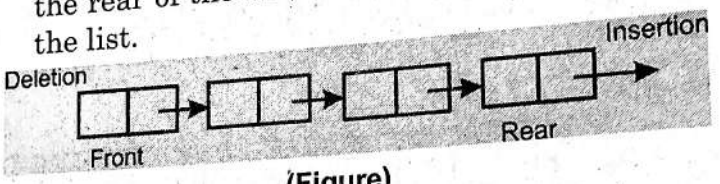
18. *Define various types of queues with the help of queuing diagram.* **(2016)**

A queue is a type of abstract data type that can be implemented as a linear or circular list. A queue has a front and a rear.

Queue can be of four types :

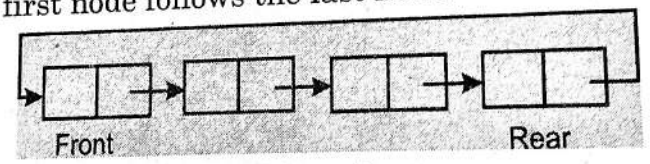
(1) Simple Queue
(2) Circular Queue
(3) Priority Queue
(4) Dequeue (Double Ended queue)

(1) **Simple Queue :** In simple queue insertion occurs at the rear of the list, and deletion occurs at the front of the list.



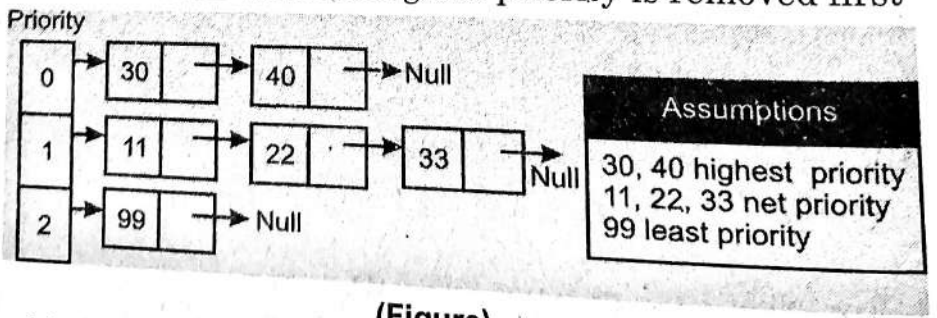
(Figure)

(2) **Circular Queue :** A circular queue is a queue in which all nodes are treated as circular such that the first node follows the last node.



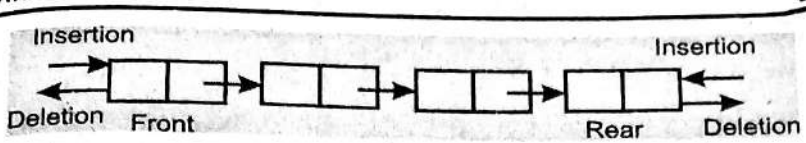
(Figure)

(3) **Priority Queue :** A priority queue is a queue that contains items that have some preset priority. When an element has to be removed from a priority queue, the item with the highest priority is removed first



(Figure)

(4) **Dequeue (Double Ended Queue) :** In dequeue(double ended queue) insertion and deletion occur at both the ends i.e. front and rear of the queue.



(Figure)

19. *What is context switch? Explain.* **(2016)**

Context Switch

A context switch occurs when a computer's CPU switches from one process or thread to a different process or thread.

Context switching allows for one CPU to handle numerous processes or threads without the need for additional processors. Any operating system that allows for multitasking relies heavily on the use of context switching to allow different processes to run at the same time. Typically, there are three situations that a context switch is necessary, as discussed below.

(1) **Multitasking :** When the CPU needs to switch processes in and out of memory, so that more than one process can be running.

(2) **Kernel/User Switch :** When switching between user mode to kernel mode, it may be used (but isn't always necessary).

(3) **Interrupts :** When the CPU is interrupted to return data from a disk read.

20. ◆ ***Consider the following set of processes with the length of CPU burst time given in ms with priority :***

Process	Burst Time	Priority
P1	6	4
P2	8	2
P3	7	1
P4	3	3

For FCFS, SJF and priority algorithm :

(1) ***Draw the Gantt chart.***

(2) ***Find the average waiting time and turnaround time.*** **(2016)**

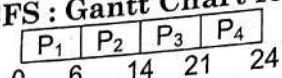
◆ ***Consider the following set of processes with the length of the CPU burst given in milli seconds :***

[B.24]

Process	Burst time
P1	6
P2	8
P3	7
P4	3

(1) *Draw the Gantt chart.*
(2) *Calculate average turnaround time and waiting time.*
(3) *Explain whether SJF is optimal.*

FCFS : Gantt Chart for FCFS is :



Average Waiting Time (AWT) and Turnaround Time (TAT) :

Process	Burst Time	Waiting Time (WT)	Turnaround Time (TAT)
P_1	6	0	6 (0 + 6)
P_2	8	6	14 (6 + 8)
P_3	7	14	21 (14 + 7)
P_4	3	21	24 (12 + 3)

Average waiting time is $= \frac{(0+6+14+21)}{4} = \frac{41}{4}$

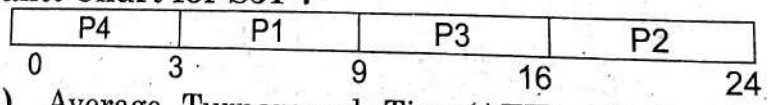
AWT = 10.25 unit **(Ans.)**

Average Turn Around Time $= \frac{(6+14+21+24)}{4}$

$= 16.25$ **(Ans.)**

SJF : Gantt Chart for SJF, Average Waiting Time (AWT) and Turnaround Time (TWT) :

Gantt Chart for SJF :

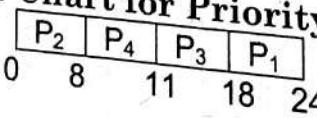


(1) Average Turnaround Time(ATT)= (3+9+16+24) / 4 = 52/4= 13.

Average Waiting time (AWT) = (0+3+9+16) / 4 = 28/4 =7.

(2) SJF (Shortest Job First) will be optimal because it gives us the optimal waiting time as well as turnaround time as compared to FIFO and Round robin scheduling algorithms.

Priority : Gantt Chart for Priority is :



Proc
P
P
P
P

Average

AWT =

Process
P_1
P_2
P_3
P_4

Averag

TAT =

21. *What is*
example

Proce
resources
access to
chance of
demands
cooperatin
producer
concurren
generates
to consu
consumer
be certai
when the
an item
not acces
tries to
produce
only pa
ensure
A
The pro
letter in

Process	Burst Time	Waiting Time
P_1	6	18
P_2	8	0
P_3	7	11
P_4	3	8

Average Waiting Time = $\frac{(18+0+11+8)}{4}$

AWT = 9.25 **(Ans.)**

Process	Burst Time	Turnaround Time (TAT)
P_1	6	18+ 6 = 24
P_2	8	8 + 0 = 8
P_3	7	11 + 7 = 18
P_4	3	8 + 3 = 11

Average Turnaround Time = $\frac{(24+8+18+11)}{4}$

TAT = 15.25 **(Ans.)**

21. *What is process synchronization? Show with the example of Bounded buffer problem.* (2015)

Process Synchronization means sharing system resources by processes in such a way that, concurrent access to shared data is handled thereby minimizing the chance of inconsistent data. Maintaining data consistency demands mechanisms to ensure synchronized execution of cooperating processes. Two processes, one called the producer and the other called the consumer, run concurrently and share a common buffer. The producer generates items that it must pass to the consumer, who is to consume them. The producer passes items to the consumer through the buffer. However, the producer must be certain that it does not deposit an item into the buffer when the buffer is full, and the consumer must not extract an item from an empty buffer. The two processes also must not access the buffer at the same time, for if the consumer tries to extract an item from the slot into which the producer is depositing an item, the consumer might get only part of the item. Any solution to this problem must ensure none of the above three events occur.

A practical example of this problem is electronic mail. The process you use to send the mail must not insert the letter into a full mailbox (otherwise the recipient will never

see it); similarly, the recipient must not read a letter from an empty mailbox (or he might obtain something meaningless but that looks like a letter). Also, the sender (producer) must not deposit a letter in the mailbox at the same time the recipient extracts a letter from the mailbox; otherwise, the state of the mailbox will be uncertain. Because the buffer has a maximum size, this problem is often called the bounded buffer problem. A (less common) variant of it is the unbounded buffer problem, which assumes the buffer is infinite. This eliminates the problem of the producer having to worry about the buffer filling up, but the other two concerns must be dealt with.

❑❑

d a letter from
in something
so, the sender
nailbox at the
n the mailbox;
be uncertain.
his problem is
(less common)
oblem, which
es the problem
ffer filling up,
n.

❑❑

DEADLOCKS

1. *Define Starvation in deadlock.* **(2023-24)**

Starvation happens when a low priority program requests a system resource but cannot run because a higher priority program has been employing that resource for a long time. When a process is ready to start executing, it waits for the CPU to allocate the necessary resources. However, because other processes continue to block the required resources, the process must wait indefinitely.

In most priority scheduling algorithms, the problem of starvation arises. The resource is frequently assigned to the higher priority process in a priority scheduling method, which helps to prevent the lower priority process from obtaining the requested resource.

Starvation is an issue that can be solved through aging. Aging raises the priority of a procedure that has been waiting for resources for a long period. It also helps to prevent a low-priority procedure from waiting indefinitely for resources.

Causes of Starvation : There are some common causes of starvation as follows :

(1) Starvation may occur if there aren't enough resources to provide to every process as needed.

(2) Starvation can occur if a process is never given the resources it needs for execution due to faulty resource allocation decisions.

(3) If higher priority operations constantly monopolize the processor, a lower priority process may have to wait indefinitely.

Solution of Handling Starvation :

(1) The resource allocation priority scheme should contain concepts such as aging, in which the priority of a process increases the longer it waits. It prevents starvation.

(2) An independent manager may be used for the allocation of resources. This resource manager distributes resources properly and tries to prevent starvation.

(3) Random process selection for resource allocation or processor allocation should be avoided since it promotes starvation.

2. ◆ ***Write short note on the Deadlock detection.*** **(2023-24)**
◆ ***Discuss Banker's Algorithm.*** **(2019)**
◆ ***Explain the terms mutual exclusion, hold and wait, preemption and circular wait in deadlocks with examples.*** **(2018)**
◆ ***What is deadlock? What are the methods to handle deadlock?*** **(2017)**
◆ ***What is Deadlock? Explain the various characteristics of deadlock in detail. Discuss deadlock prevention schemes.*** **(2016)**
◆ ***What is deadlock?*** **(2015)**
◆ ***Describe Banker's Algorithm for safe allocation.*** **(2015)**

Deadlock and Deadlock Prevention Schemes

A deadlock occurs when two or more processes are waiting indefinitely for an event (e.g. a resource to become available) that can be caused only by one of the waiting processes. In order for a process to execute, it needs resources. If the resources are not available at that time, the process enters a wait state. It may happen that the waiting process will never leave the wait state, because the resources it needs to execute are being held by another process. This situation is an example of a deadlock. There are three methods for dealing with deadlocks :

(1) Use some protocol to ensure that the system will never enter a deadlock state.

(2) Allow the system to enter deadlock state and then recover. Ignore the problem and pretend that deadlocks never occur in the system. This method is implemented by most operating systems, such as UNIX.

How a Deadlock Occurs : For a deadlock to occur, four conditions must hold simultaneously. One is the mutual exclusion condition, under which processes perform exclusive access to resources. A second is the wait condition. A process that has requested a resource will wait for the request to meet it, continuing to hold all the other

resources it has acquired. A third condition is that of no preemption. No resource can be preempted from a process that has acquired it but not released it. The fourth is the circular wait condition. There exists a circular chain of processes, each of which is waiting for a resource held by its predecessor in the chain.

Methods for Handling Deadlocks : One of three fundamental policies can be selected to cope with the problem of deadlock. The policy of prevention is based upon a judgment that deadlock is so costly that it is best to spend extra system resources to prevent the possibility of deadlock arising under any circumstances. The policy of avoidance is to ensure that deadlock, will not occur for the particular set of processes and requests being run at the moment.

The policy of detection and subsequent recovery is based on the judgment that deadlock occurs infrequently enough that it would cost more to prevent or avoid it than to test for its occurrence and effect recovery when it is found. Prevention can be thought of as prohibiting the existence of unsafe states, avoidance as prohibiting entry into unsafe states, and recovery as prohibiting permanent residence in unsafe states.

Deadlock Prevention : A mechanism to enforce prevention need only to insure that some one of the four conditions necessary for deadlock cannot arise. The mutual exclusion condition is suppressed by permitting unrestricted sharing resources. Although, this is quite convenient for such resources as re-enterable code segments or the use of a disk drive to access different data sets, it is unreasonable for shared variables in critical regions or for a reel of magnetic tape.

The wait for condition can be suppressed by preallocation. A process must request all of its resources initially and cannot proceed until all resources have been allocated. Consequently, the total resources required by the concurrent processes cannot exceed the capacity of the system. Although preallocation is often used, it entails a substantial cost in the inefficient resource utilization. Each process must wait until all of its resources are available, even one not needed until late in its execution. The resources allocated may not all even be used by the process, if it requests resources needed only in exceptional

circumstances. The resources that are not used until late are denied to other processes, causing them to wait. Another disadvantage is the difficulty of formulating resource requests for a process whose requirements cannot be predicted before execution starts.

The no preemption condition is suppressed by allowing the operating system to take resources away from a process. This is unpractical if the state of the process can be easily saved for later restoration. For readily preemptable resources, the cost of preemption is mainly the time required. Two different disciplines may be applied. One is to require a job that requests more of a resource to prevent from itself first of whatever amount of that resource it already holds. Although this may often help, it is not a total suppression of the any preemption condition. The other discipline, total suppression, takes away all resources from a job that becomes blocked by requesting any further resource.

The circular wait condition can be suppressed by preventing the formation of a circle. This is accompanied by hierarchic allocation. Resources are grouped into a hierarchy of levels. Once a process has acquired resources at one level, it can request further resources only at a higher level. It can release resources at a given level only after releasing all resources acquired at every higher level. After a process has acquired and released resources at a given level, it can again request resources at that same level. Preallocation can be viewed as a special case of hierarchic allocation that has but one level. Hierarchic allocation is a bit more costly to perform than preallocation, but it can reduce the waste associated with full preallocation. The reduction may not develop, however, if the order in which a process needs resources is different from that in which the levels are ordered. If a plotter is needed early in execution and a tape drive only towards the end, but the tape drive is at a lower level than the plotter, the tape drive will have to be allocated early and remain idle until needed.

Deadlock Avoidance : If none of the four conditions is suppressed, entry to an unsafe state can still be forbidden, if the system has knowledge of the sequence of requests associated with each of the concurrent processes. We can

prove that if th
exists at least o
follow there fro
Consequently, i
resource alloca
to an unsafe st
be granted. To
search over
methods have
rather efficie
controlled all
of requests
advance. If,
each type is
the allocati
determine
the total
sequence o
the total
solution
algorithm

Banker

Let
Request
resource
process
(1) If
Ot
ex
(2) If
O
m
(3) T
r
f

trans
reso
mus
stat

prove that if the computation is in any safe state, there exists at least one sequence of states that a trajectory can follow there from without ever entering an unsafe state. Consequently, it is sufficient to test whether a requested resource allocation will immediately admit a computation to an unsafe state. If so, the request is denied. If not, it can be granted. To test whether a state is safe or not requires a search over sequences of future process actions. Some methods have been developed to perform these searches rather efficiently. The foregoing approach is an example of controlled allocation. It is often the case that the sequence of requests associated with each process is not known in advance. If, however, the total demand for resources of each type is known in advance, it is still possible to control the allocation to avoid unsafe states. What is done is to determine for each request whether there exists, within the total demands of all processes, any subsequent sequence of requests that can lead to an unsafe state given the total amount of resource available. The classical solution to this problem is known as the Banker's algorithm.

Banker's Algorithms for Deadlock Avoidance

Let Request$_i$ be the request vector for process p_i . If Request$_i$ [j] = k , then process p_i wants k instances of resource type r_j. When a request for resources is made by process p_i, the following actions are taken :

(1) If Request $_i$ = < Need$_i$ then proceed to step 2. Otherwise, we have an error, since the process has exceeded its maximum claim.

(2) If Request $_i$ = < Available$_i$ then proceed to step 3. Otherwise, the resources are not available, and p_i must wait.

(3) The system pretends to have allocated the request resources to process p_i by modifying the state as follows :

(a) Available : = Available – Request$_i$;

(b) Allocation : = Allocation$_i$ – Request$_i$;

(c) Need$_i$: = Need $_i$ – Request$_i$;

If the resulting resource allocation state is safe, the transaction is completed and process p_i is allocated its resources. However, if the new state is unsafe, then p_i must wait for Request $_i$ and the old resource allocation state is restored.

A major cost of deadlock avoidance by controlled allocation is nevertheless the time required to execute the avoidance algorithm, because it must be invoked for every request. Moreover, the avoidance algorithm runs particularly slowly when the system is near deadlock.

Recovery from Deadlock : Once deadlock has been detected, recovery must be affected. This will certainly involve restarting one or more processes. A deadlocked process is blocked at some point in its execution. It must be restored to a condition from which it can resume execution. For most processes, this is possible only from the beginning, and some temporary force may need to be overcome even then. If the process has made an occasional record of its state, a snapshot of its execution, in order to permit resumption from a specific point, a checkpoint, it is possible to resume without repeating all of the computation prior to the preemption. Checkpoints are not normally provided for the purpose of assistance in recovery from error, particularly in long production jobs and in real-time systems.

The simplest recovery method is to terminate all processes, and restart the operating system afresh. A less drastic method is to terminate all deadlocked processes, whose users will eventually reintroduce them. A third method is to terminate deadlocked processes one at a time, invoking the detection algorithm after each termination, until the deadlock has been dissolved. A fourth method, applicable if checkpoints have been provided, is to restart the deadlocked processes from checkpoints and hope that the deadlock will not occur.

Characteristics of Deadlock

Deadlock : This is a situation where a group of processes are permanently blocked as a result of each process having acquired a subset of the resources needed for its completion and waiting for release of the remaining resources held by other in the same group thus making it impossible for any of the processes to proceed.

Suppose that we have two processes A_1 and A_2. These are interleaved in such a way that at some point process A_1 is granted the use of the printer and A_2 manages to size a disk drive. Now both the processes are deadlocked as neither of them can acquire the balance of

the resources that it needs to make further progress. A deadlock situation can arise if the following four conditions hold simultaneously in a system :

(1) **Mutual Exclusion :** At least one resource must be held in non-sharable mode; that is only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

(2) **Hold and Wait :** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

(3) **No Preemption :** Resources cannot be preempted that is a resource can be released only voluntarily by the process holding it, after that process has completed its task.

(4) **Circular Wait :** A set $\{P_0, P_1, ..., P_n\}$ of waiting process must exist such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2,....P_{n-1} is waiting for a resource that is held by P_n and P_n is waiting for a resource that is held by P_0.

Dead Lock Detection

(1) **Single Instance of Each Resource Type :** If all resources have only a single instance, then we can define a deadlock detection algorithm that uses a variant of the resource allocation graph, called a wait for graph. We obtain this graph from the resource-allocation graph by removing the resource nodes and collapsing the appropriate edges.

(2) **Several Instance of a Resource Type :** The wait-for graph scheme is not applicable to a resource allocation system with multiple instances of each resource type. We turn how to a dead lock detection algorithm that is applicable to such a system.

   (a) **Variable :** A vector of length m indicates the number of available resources of each type.

   (b) **Allocation :** An $n \times m$ matrix defines the number of resources of each type currently allocated to each process.

(c) **Request :** An $n \times m$ matrix indicates the current request of each process if request $[i]$ $[j]$ equals k then process P_i is requesting k more instances of resource type R_j.

The $\leq$ relation between two vectors is defined. We again treat the rows in the matrices Allocation and Request as vectors; we refer them as Allocation$_i$ and Request$_i$, respectively. The detection algorithm described here simply investigates every possible allocation sequence for the process that remains to be completed.

(1) Let work and finish be vectors of length m and n, respectively. Initialize work = Available. For $i = 0$, $1....n - 1$, if Allocation $i \neq 0$, then Finish $[i]$ = false; otherwise finish$[i]$ = true.

(2) Find an index i such that both
   (a) Finish$[i]$ = false
   (b) Request$_i \leq$ 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 $[i]$ = = false, for some i, $0 \leq i < n$, then the system is in a deadlock state. Moreover, if Finish $[i]$ = = false, then process P_i is deadlocked.

3. ◆ ***Explain the resource allocation graph.*** **(2022-23)**
   ◆ ***Write a short note on Resource Allocation graph.*** **(2018)**

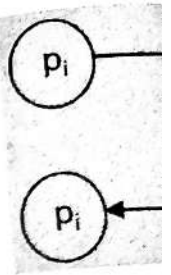
Deadlock

A set of processes is in a deadlock state, if every process in the set is waiting for an event (release) that can only be caused by some other process in the same set.

Resource Allocation Graphs

Resource allocation graphs are drawn in order to see the allocation relations of processes and resources easily. In these graphs, processes are represented by circles and resources are represented by boxes. Resource boxes have some number of dots inside indicating available number of that resource that is number of instances.

(1) If the r
then th
instance
(2) If the r
a deadl
(3) If ther
resourc
deadlo



4. ◆ ***Expl***
dea
◆ ***Who***
for
◆ ***List***
dea

The
deadlock
preventi

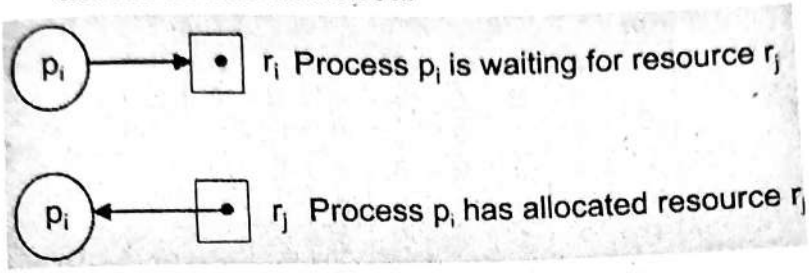
Necess

A
conditio
(1) M
he
a
re
re
(2) H
o
r
p
(3) N
r
t
c

(1) If the resource allocation graph contains no cycles then there is no deadlock in the system at that instance.

(2) If the resource allocation graph contains a cycle then a deadlock may exist.

(3) If there is a cycle, and the cycle involves only resources which have a single instance, then a deadlock has occurred.



(Figure)

4. ◆ ***Explain four necessary conditions for deadlock.*** **(2022-23)**
◆ ***What are four necessary conditions responsible for deadlocks?*** **(2019)**
◆ ***List four necessary conditions for occurrence of deadlock.*** **(2015)**

The processes never finish their execution in a deadlock state, and system resources are tied up, preventing other jobs from starting.

Necessary Conditions

A deadlock condition can arise if the following four conditions hold simultaneously in a system.

(1) **Mutual Exclusion :** At least one resource must be held in a non-sharable mode; i.e. Only one process at a time can use the resource. Another process requesting for the same must be delayed until the release of the resource.

(2) **Hold and Wait :** A process must be holding at least one resource and waiting to acquire additional resource that are currently being held by other processes.

(3) **No Preemption :** Resources cannot be preempted means a resource can be released only voluntarily by the process holding it, after that process has completed its task.

(4) **Circular Wait :** A set of waiting processes must exist such that P_0 is waiting for a resource that is held by P_1, and further more until P_n, and P_n is waiting for a resource held by P_0.

These four conditions are necessary for occurring deadlock.

5. ◆ ***Consider the following snapshot of the system :***

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Answer the following questions using the Banker's algorithm :

(1) ***What is the content of the Matrix Need?***

(2) ***Is the system in a safe state?*** **(2022-23)**

◆ ***Consider the following snapshot at time t_0 of the system and answer the following questions using Banker's algorithm :*** **(2016)**

Process	Allocation	Max.	Available
	X Y Z	X Y Z	X Y Z
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
iP_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

(1) ***Compute the need matrix.***

(2) ***Is the system in a safe state?***

(3) ***If the request from P_1 arrives for (102), can the request be granted immediately?***

(1) Need Matrix :

Need $[i, j]$ = Max $[i, j]$ – Allocation $[i, j]$

So, the content of need matrix is :

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

(2) System in a Safe State : Applying the safety algorithm on the given system,

m = 3, n = 5 Step 1 of Safety Algo

Work = Available

Work =

3	3	2

Finish =

0	1	2	3	4
False	False	False	False	False

Step 2

For i = 0

$Need_0$ = 7, 4, 3

×

7, 4, 3 3, 3, 2

Finish [0] is false and $need_1$ > work

But Need ≤ Work

So P_0 must wait **Step 2**

For i = 1

$Need_1$ = 1, 2, 2

✓

1, 2, 2 3, 3, 2

Finish [1] is false and $need_1$ < work

So P_1 must be kept in safe sequence **Step 3**

3, 3, 2 2, 0, 0

Work = Work + $Allocation_1$

Work =

A	B	C
5	3	2

Finish =

0	1	2	3	4
False	True	False	False	False

Step 2

For i = 2

$Need_2$ = 6, 0, 0 6,0, 0 5, 3, 2

Finish [2] is false and $need_2$ > work

So P_2 must wait **Step 2**

For i = 3

$Need_3$ = 0, 1, 1

✓

0, 1, 1 5, 3, 2

Finish [3] = false and $need_3$ < work

So P_3 must be kept in safe sequence **Step 3**

5, 3, 2 2, 1, 1

Work = Work + $Allocation_3$

Work =

A	B	C
7	4	3

Finish = | 0 | 1 | 2 | 3 | 4 |
| --- | --- | --- | --- | --- |
| False | True | False | True | False |

Step 2

For i = 4
$Need_4$ = 4, 3, 1
4, 3, 7, 4, 3
1

Finish [4] is false and $need_4$ < work
So P_4 must be kept in safe sequence

Step 3

Work = 7, 4, 3 Work + $Allocation_4$ 0, 0, 2
Work = | A | B | C |
| --- | --- | --- |
| 7 | 4 | 5 |

Finish = | 0 | 1 | 2 | 3 | 4 |
| --- | --- | --- | --- | --- |
| False | True | False | True | True |

Step 2

For i = 0
$Need_0$ = 7, 4, 3
✓
7, 4, 3, 3, 2
3

Finish [0] is false and need < work
So P_0 must be kept in safe sequence

Step 3

Work = 7, 4, 5 Work + $Allocation_0$ 0, 1, 0
Work = | A | B | C |
| --- | --- | --- |
| 7 | 5 | 5 |

Finish = | 0 | 1 | 2 | 3 | 4 |
| --- | --- | --- | --- | --- |
| True | True | False | True | True |

Step 2

For i = 2
$Need_2$ = 6, 0, 0
✓
6, 0, 7, 5, 5
0

Finish [2] is false and $Need_2$ < Work
So P_2 must be kept in safe sequence

Step 3

Work = 7, 5, 5 Work + $Allocation_2$ 3, 0, 2
Work = | A | B | C |
| --- | --- | --- |
| 10 | 5 | 7 |

Finish = | 0 | 1 | 2 | 3 | 4 |
| --- | --- | --- | --- | --- |
| True | True | True | True | True |

Finish [i] = true for $0 \leq i \leq n$
Hence the system is in safe state
The safe sequence is P_1, P_3, P_4, P_0, P_2

Step 4

(3) P_1 Arrives for (1, 0, 2), can the Request be Granted Immediately :

	A	B	C
$Request_1$	1,	0,	2

To decide whether the request is granted we use Resource Request algorithm

Step 1

$Request_1 \leq Need_1$

1, 0, 2 ≤ 1, 2, 2 ✓

Step 2

$Request_1 \leq Available$

1, 0, 2 ≤ 3, 3, 2 ✓

Step 3

$Available = Available - Request_1$
$Allocation_1 = Allocation_1 + Request_1$
$Need_1 = Need_1 - Request_1$

Process	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	4	3	**2**	**3**	**0**
P_1	**2**	**0**	**0**	**0**	**2**	**0**			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	0	0	2	4	3	1			

We must determine whether this new system state is safe. To do so, we again execute safety algorithm on the above data structures.

Step 1 of Safety Algo

m = 3, n = 5

Work = Available

Work = | 2 | 3 | 0 |

Finish =

0	1	2	3	4
False	False	False	False	False

Step 2

For i = 0

$Need_0$ = 7, 4, 3

×

7, 4, 3 ≤ 2, 3, 0

Finish [0] is false and $Need_0 > Work$

So P_0 must wait But Need ≤ Work

Step 2

For i = 1

$Need_1$ = 0, 2, 0

✓

0, 2, 0 ≤ 2, 3, 0

Finish [1] is false and $Need_1 < Work$

So P_1 must be kept in safe sequence

Step 3

2, 3, 0 3, 0, 2

Work = Work + Allocation$_1$

Work =

A	B	C
5	3	2

Finish =

0	1	2	3	4
False	True	False	False	False

Step 2

For i = 2
Need$_2$ = 6, 0, 0
×
6, 0, 0 5, 3, 2

Finish [2] is false and Need$_2$ > Work
So P_2 must wait

Step 2

For i = 3
Need$_3$ = 0, 1, 1
✓
0, 1, 1 5, 3, 2

Finish [3] is false and need$_3$ < work
So P_3 must be kept in safe sequence

Step 3

5, 3, 2 2, 1, 1

Work = Work + Allocation$_3$

Work =

A	B	C
7	4	3

Finish =

0	1	2	3	4
False	True	False	True	False

Step 2

For i = 4
Need$_0$ = 4, 3, 1
✓
4, 3, 1 7, 4, 3

Finish [4] is false and need$_4$ < work
So P_4 must be kept in safe sequence

Step 3

7, 4, 3 0, 0, 2

Work = Work + Allocation$_4$

Work =

A	B	C
7	4	5

Finish =

0	1	2	3	4
False	True	False	True	True

Step 2

For i = 0
Need$_0$ = 7, 4, 3
✓
7, 4, 3 7, 4, 5

Finish [0] is false and need < work
So P_0 must be kept in safe sequence

Step 3

7, 4, 5 0, 1, 0

Work = Work + Allocation$_0$

Work =

Finish =

For i = 2
Need$_2$ = 6

Finish [2] is fa
So P_2 must b

Work =
Work =

Finish =

Finish [i] = t
Hence the
The Safe
Hen
immediately g

6. ***Consider th***

	Alloc	
	A	B
P_0	0	0
P_1	1	0
P_2	1	3
P_3	0	6
P_4	0	0

answer t
Algorithm
(1) ***Who***
(2) ***Is th***
(3) ***If***
(0,

(1) Ne

Work = A B C

7	5	5

Finish =

0	1	2	3	4
True	True	False	True	True

For i = 2 **Step 2**

$Need_2$ = 6, 0, 0

✓

6, 0, 0 7, 5, 5

Finish [2] is false and $need_2$ < work

So P_2 must be kept in safe sequence

Step 3

7, 5, 5 3, 0, 2

Work = Work + $Allocation_1$

Work = A B C

10	5	7

Finish =

0	1	2	3	4
True	True	True	True	True

Step 4

Finish [i] = true for $0 \leq i \leq n$

Hence the system is in safe state

The Safe sequence is P_1, P_3, P_4, P_0, P_2

Hence the new system state is safe, so we can immediately grant the request for process P_1.

6. *Consider the following snapshot of a system :* (May 2013, 2014, 2017, 2019)

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

answer the following questions using the Banker's Algorithm :

(1) ***What is the content of the matrix need?***

(2) ***Is the system in a safe-state?***

(3) ***If a request from process P_1 arrives for (0, 4, 2, 0) can be request immediately***

(1) Need = Maximum - Allocation

	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2

P_3	0	0	2	0
P_4	0	6	4	2

(2) P_0 runs to completion

Max Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	1	7	5	0
P_2	2	3	5	6
P_3	0	6	5	2
P_4	0	6	5	6

Allocation Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	1	0	0	0
P_2	1	3	5	4
P_3	0	6	3	2
P_4	0	0	1	4

P_2 runs to Completion

Max Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	1	7	5	0
P_2	2	3	5	6
P_3	0	6	5	2
P_4	0	6	5	6

Allocation Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	1	0	0	0
P_2	0	0	0	0
P_3	0	6	3	3
P_4	0	0	1	4

P_1 runs to completion

Max Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	0	0	0	0
P_2	0	0	0	0
P_3	0	6	5	2
P_4	0	6	5	6

Allocation Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	0	0	0	0
P_2	0	0	0	0
P_3	0	6	3	2
P_4	0	0	1	4

P_3 runs to completion

Max Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	0	0	0	0
P_2	0	0	0	0
P_3	0	0	0	0
P_4	0	6	5	6

Allocation Matrix

	A	B	C	D
P_0	0	0	0	0
P_1	0	0	0	0
P_2	0	0	0	0
P_3	0	0	0	0
P_4	0	0	1	4

P_4 runs to completion the system is in a safe state.

Safe Sequence : $<P_0\ P_2\ P_1\ P_3\ P_4>$ and the current availability : [2 1 1 8]

(3) If request of P_2 is granted, availability will become [1 1 0 0] and row 2 in C will be (1 4 2 0). This is a safe state and the request of P_2 can be granted.

7. *Discuss when a resource allocation graph contains a cycle but no deadlock.* **(2019)**

(1) If grap
(2) If grap
(a) If
(b)

8. *Consider*
state :

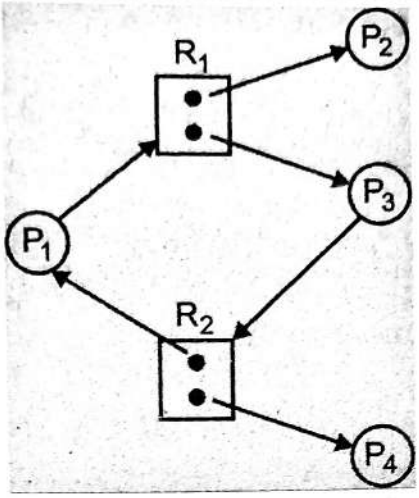
Proces
P_1
P_2
P_3

Answe
algori
(1)
(2)

(1)

(2)

Graph With a Cycle But No Deadlock



(Figure)

(1) If graph contains no cycles ⇒no deadlock

(2) If graph contains a cycle ⇒

(a) If only one instance per resource type, then deadlock.

(b) If several instances per resource type, possibility of deadlock.

8. ***Consider the following current resource allocation state :*** **(2018)**

Process	Allocation			Maximum			Available		
	A	B	C	A	B	C	A	B	C
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	3			

Answer the following questions using Banker's algorithm :

(1) ***What is the content of the Need Matrix?***

(2) ***Is the system in safe state?***

(1) Need Matrix = Maximum – Allocation

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} A & B & C \\ 1 & 4 & 5 \\ 2 & 3 & 0 \\ 2 & 2 & -1 \end{bmatrix}$$

(2) For Process P_1

Need = (1 4 5)

Available = (7 7 10)

Here Need < Available
Condition in true
Process P_1 execute
So Available = (7 7 10) + (2 2 3) = (9 9 13)
For Process P_2
Need = (2 3 0)
Available = (9 9 13)
Here Need < Available
Condition is true
Process P_2 execute
So Available = (9 9 13) + (2 0 3) = (11 9 15)
For Process P_3
Need = (2 2 −1)
Available = (11 9 15)
Here Need< Available
Condition is true
Process P_3 execute
and Available = (11 9 15) + (1 2 4) = (12 11 19)
Safety sequence = $< P_1, P_2, P_3 >$

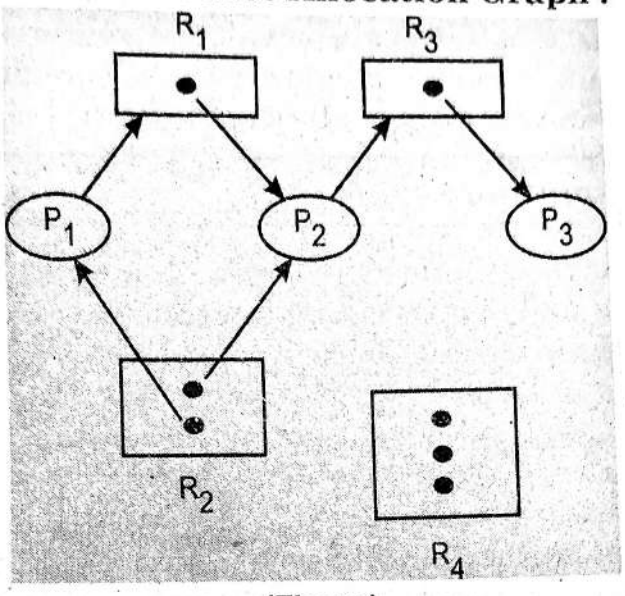
9. *Describe resource allocation graph with multiple instance of resources with a suitable example.* (2015)

Resource allocation graph or wait for graph is a directed graph. Using this graph deadlocks can be explained easily. 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 $P = \{P_1, P_2, ..., P_n\}$, the set consisting of all the active processes in the system, and $R = \{R_1, R_2, ... R_m\}$, the set consisting of all resource types in the system.

Request edge – directed edge $P_1 \rightarrow R_j$
Assignment edge – directed edge $R_j \rightarrow P_i$
Resource type with 4 instances
P_i requests instance of R_j
P_i is holding an instance of R_j
P_i

P_i
R_j
R_j

Example of a Resource Allocation Graph :



(Figure)

10. ◆ ***Compare deadlock prevention and deadlock avoidance.*** **(2015)**

◆ ***Differentiate clearly between Deadlock Avoidance and Prevention. Which one is generally preferred? Why?*** **(May 2013)**

Difference

Deadlock Prevention	Deadlock Avoidance
Preventing deadlocks by constraining how requests for resources can be made in the system and how they are handled.	The system dynamically considers every request and decides whether it is safe to grant it at this point.
The goal is to ensure that at least one of the necessary conditions for deadlock can never hold.	The system requires additional apriority information regarding the overall potential use of each resource for each process.

Deadlock prevention is more preventive and conservative. Conservative is in the way that, it would decrease the power of the system by not allowing mutual exclusion.

11. ***What is meant by a non-shareable resource? Explain clearly.*** **(2014)**

A non-preemptable or non-shareable resource is one that cannot be taken away from process (without causing ill effect). For example, CD resources are not preemptable at an arbitrary moment. Reallocating resources can resolve deadlocks that involve preemptable resources. Deadlocks that involve non-preemptable resources are difficult to deal with. Resources already allocated to a process.

12. *Write safe and unsafe state.* **(2014)**

A state is considered safe if it is possible for all processes to finish executing (terminate). Since the system cannot know when a process will terminate, or how many resources it will have requested by then, the system assumes that all processes will eventually attempt to acquire their stated maximum resources and terminate soon afterward.

This is a reasonable assumption in most cases since the system is not particularly concerned with how long each process runs (at least not from a deadlock avoidance perspective). Also, if a process terminates without acquiring its maximum resources, it only makes it easier on the system. A safe state is considered to be the decision maker if it is going to process ready queue. Safe state ensures the security. Given that assumption, the algorithm determines if a state is safe by trying to find a hypothetical set of requests by the processes that would allow each to acquire its maximum resources and then terminate (returning its resources to the system). Any state where no such set exists is an unsafe state.

❑❑

DEVICE MANAGEMENT

1. ◆ ***What are the various disk-scheduling algorithms? Explain.*** **(2023-24)**
   ◆ ***Write the name of disk scheduling algorithms. Write the method and explain the working of any three methods.*** **(2018)**
   ◆ ***What is disk scheduling? List various types of disk scheduling and explain any two of them.*** **(2016)**
   ◆ ***Write short note on Disk Scheduling.*** **(2014)**

When a process needs input/output to or from the disk, it issues a system call to the operating system as whether the operation is input or output, what is the disk address for the transfer; what is the memory address for the transfer what the number of bytes to be transferred is. If the desired disk drive and controller are available, the request can be serviced immediately.

If the drive or controller is busy, any new requests for service will be placed on the queue of pending requests for that drive. For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus when one request is completed, the operating system chooses which pending request to service next. The process of choosing is known as disk scheduling and can be performed by many ways as :

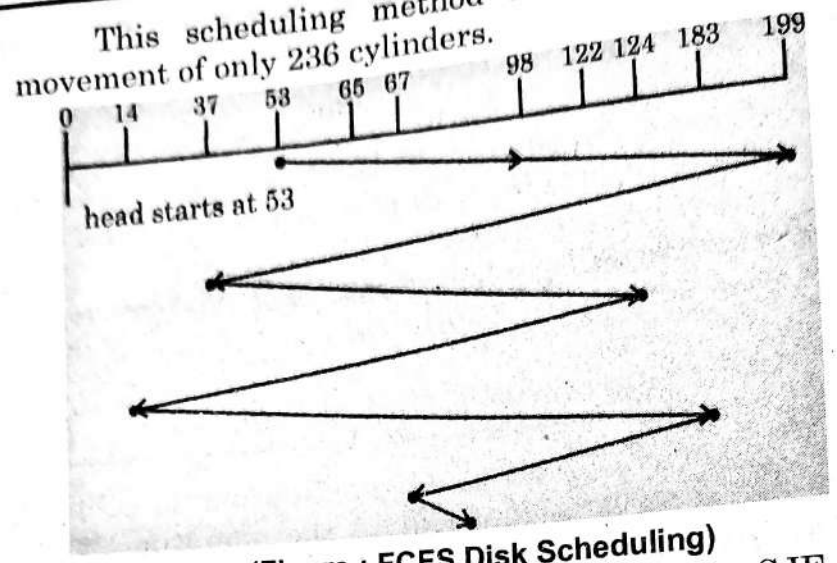
(1) FCFS
(2) SSTF
(3) SCAN
(4) C-SCAN
(5) LOOK

FCFS Scheduling : The simplest one, and based on First Come First Served basis. But it is not as much fast, for example a disk queue with requests for input/output to blocks on cylinders.

Queue = 98, 183, 37, 122, 14, 124, 65, 67.

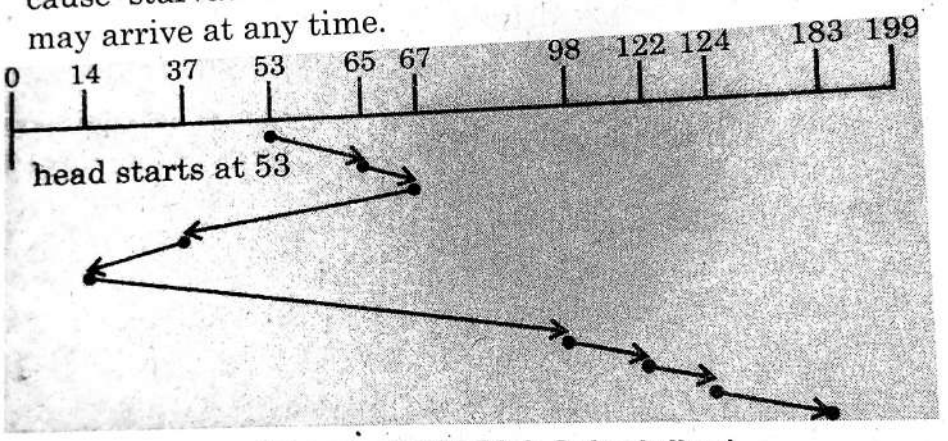
If the disk head is initially at cylinder 53, it will first move from 53 to 98 then 183, 37 then further till 67 as sequentially. For a total head movement of 640 cylinders.

This scheduling method results in a total head movement of only 236 cylinders.



(Figure : FCFS Disk Scheduling)

SSTF Scheduling : It is most likely to the SJF. It may cause starvation of some requests. In this, the requests may arrive at any time.



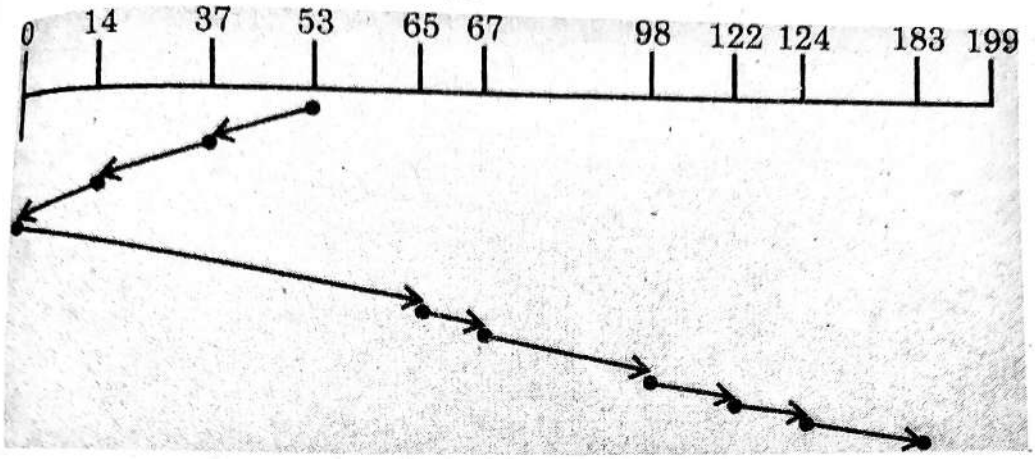
(Figure : SSTF Disk Scheduling)

Let suppose, the queue having two requests for cylinders 14 and 186 and while servicing the request from 14, a new request near 14 arrives and serviced next, making the request at 186 wait and while it being serviced, another request close to 14 could arrive. This is not optimal scheduling.

Queue = 98, 183, 37, 122, 14, 124, 65, 67
Head starts at = 53

SCAN Scheduling : In this type of scheduling, the disk arm starts at one end towards the other end; servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed and servicing continues. The head continually scans back and forth across the disk.

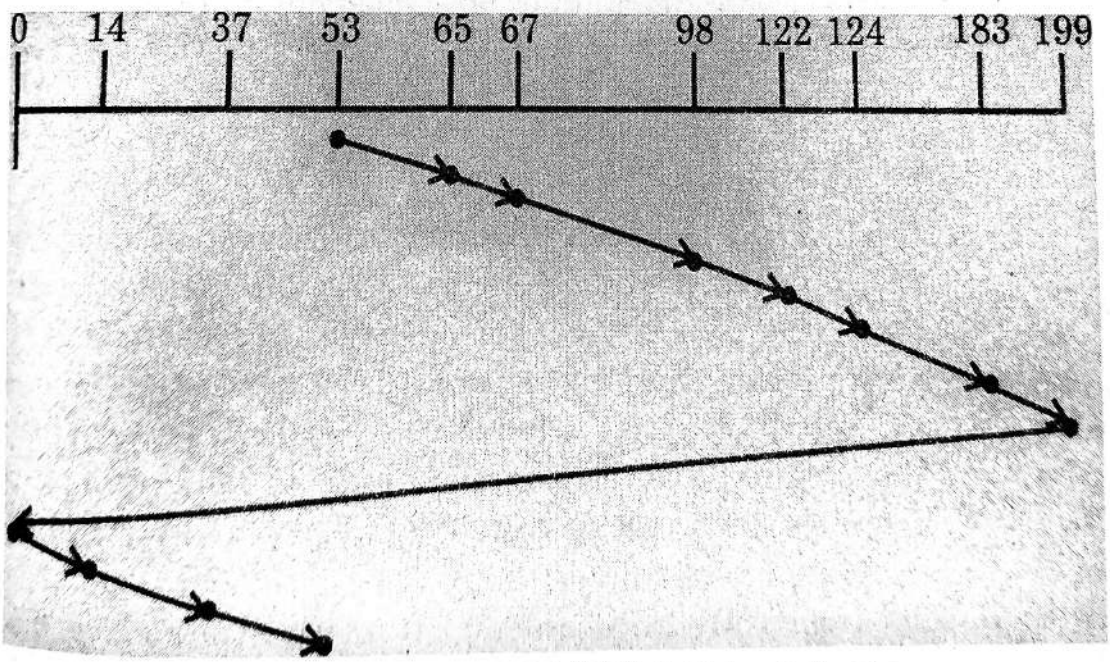
Queue = 98, 183, 37, 122, 14, 124, 65, 67
Head starts at = 53



(Figure : SCAN Scheduling)

C-SCAN Scheduling : It provides a more uniform wait time. In this the head moves from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on the return trip.

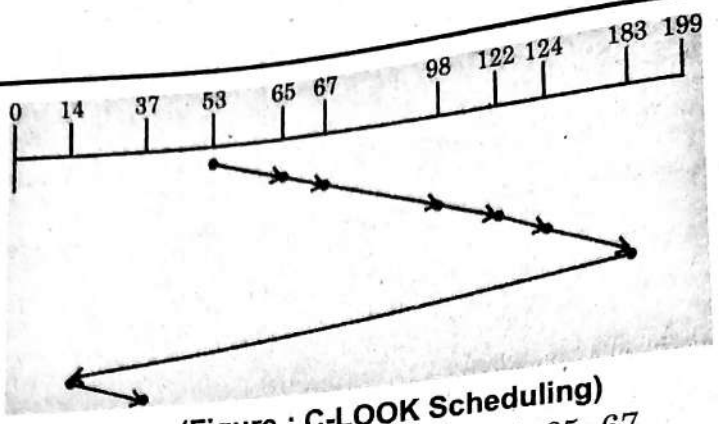
Queue = 98, 183, 37, 122, 14, 124, 65, 67
Head starts at = 53



(Figure : C-SCAN Disk Scheduling)

LOOK Scheduling : In this type of scheduling the arm goes only as far as the final request in each direction. Then it reverses direction immediately without going all the way to the end of the disk. The request is looked for a request before continuing to move in a particular direction.

[D.4]



(Figure : C-LOOK Scheduling)

Queue = 98, 183, 37, 122, 14, 124, 65, 67

Head starts at = 53

2. ◆ ***Discuss dedicated devices and shared devices.*** **(2022-23)**

◆ ***Discuss shared devices and virtual devices for device management.*** **(2018)**

◆ ***What are techniques for Device management?***

◆ ***What are dedicated devices?***

Techniques for Device Management

Three major techniques are used for managing and allocating devices :

(1) Dedicated,

(2) Shared, and

(3) Virtual.

(1) **Dedicated Devices :** A dedicated device is allocated to a job for the job's entire duration. Some devices lend themselves to this form of allocation. It is difficult, for example, to share a card reader, tape or printer. If several users were to use the printer at the same time, would they cut up their appropriate output and paste it together? Unfortunately, dedicated assignment may be inefficient if the job does not fully and continually utilize the device. The other techniques, shared and virtual, are usually preferred whenever they are applicable.

(2) **Shared Devices :** Some devices such as disks, drums and other Direct Access Storage Devices (DASD) may be shared concurrently by several processes. Several processes can read from a single disk at essential the

same time. The management of a shared of a shared device can become quite complicated, particularly if utmost efficiency is desired. For example, if two processes simultaneously request a read from disk A, some mechanism must be employed to determine which request should be handled first. This may be done partially by software (the I/O schedule and traffic controller) or entirely by hardware (as in some computers with very sophisticated channels and control units).

Policy for establishing which process request is to be satisfied first might be based on (1) a priority list or (2) the objective of achieving improved system output (for example, by choosing whichever request is nearest to the current position of the read heads of the disk).

(3) **Virtual Devices :** Some devices that would normally have to be dedicated (e.g., card readers) may be converted into shared devices through into shared devices through techniques such as spooling. For example, a spooling program can read and copy all card input onto a disk at high speed. Later, when a process tries to read a card, the spooling program intercepts the request and converts it to a read from the disk. Since a disk may be easily shared by several users, we have converted a dedicated device to a shared device, changing one card reader into many "virtual" card readers. This technique is equally applicable to a large number of peripheral devices, such as teletypes, printers and most dedicated slow input/ output devices.

(4) **Generalized Strategies :** Some professionals have attempted to generalize device management even more than is done here. For example, the call side of spooling is similar to the file system and buffering bears striking similarity to spooling. We could generalize even more and view memory, processors and I/O devices as simple devices to which the same theory should apply. At present we realize that there are practical limitations to these generalizations,

although at some time in the future more general theories may emerge that are applicable to all devices and that have direct practical applications.

3. *Discuss the Disk – space management methods.* **(2019)**

Disk-Space Management

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly two approaches by using which, the free blocks in the disk are managed.

(1) **Bit Vector :** In this approach, the free space list is implemented as a bit map vector. It contains the number of bits where each bit represents each block.

If the block is empty then the bit is 1 otherwise it is 0. Initially all the blocks are empty therefore each bit in the bit map vector contains 1.

(2) **Linked List :** It is another approach for free space management. This approach suggests linking together all the free blocks and keeping a pointer in the cache which points to the first free block.

Therefore, all the free blocks on the disks will be linked together with a pointer. Whenever a block gets allocated, its previous free block will be linked to its next free block.

4. *Suppose the head of moving disk is currently serving at request at track 60. If the queue of request is kept in FIFO order, what is the total head movement to satisfy these requests for the following disk scheduling algorithms :* **(2019)**

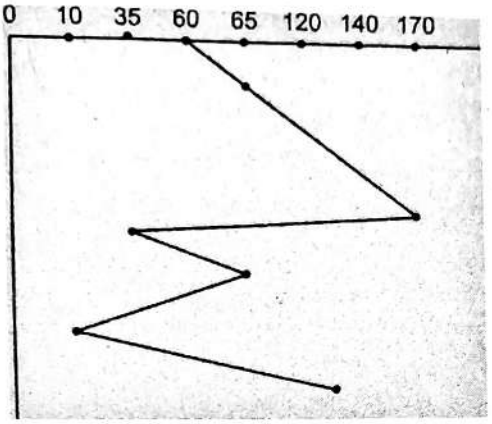
(1) *FCFS*
(2) *SSTF*
(3) *SCAN*
(4) *C – SCAN*

Disk request come into the disk drives for cylinder 65, 170, 35, 120, 10, 140 in that order.

Given Queue
65, 170, 35, 120, 10, 140

Start Request at track 60

(1) In case FCFS :

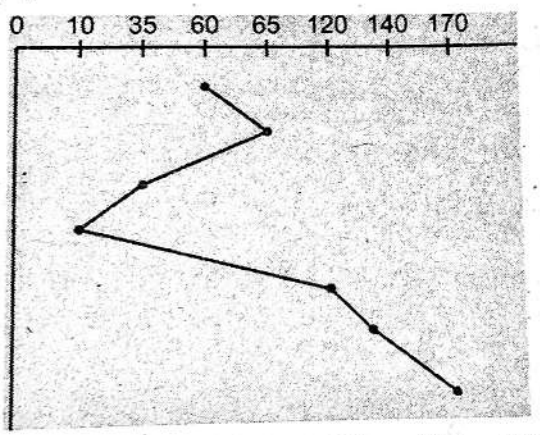


Total Seek Time = (170 – 60) + (170 – 35) + (65 – 35) + (65 – 10) + (140 – 10)

= 110 + 135 + 30 + 55 + 130

= 460 **(Ans.)**

(2) SSTF :

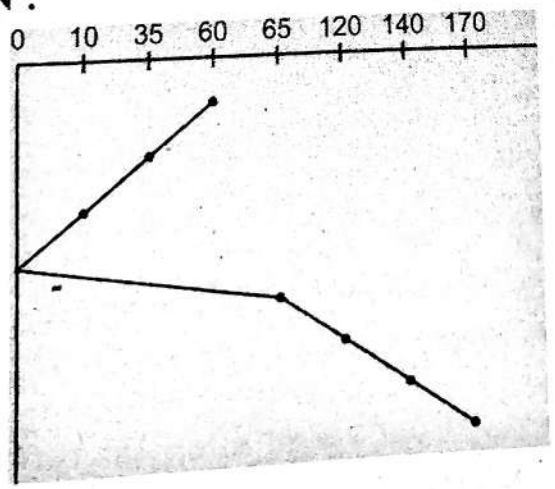


Total Seek Time = (65 – 60) + (65 – 10) + (170 – 10)

= 5 + 55 + 160

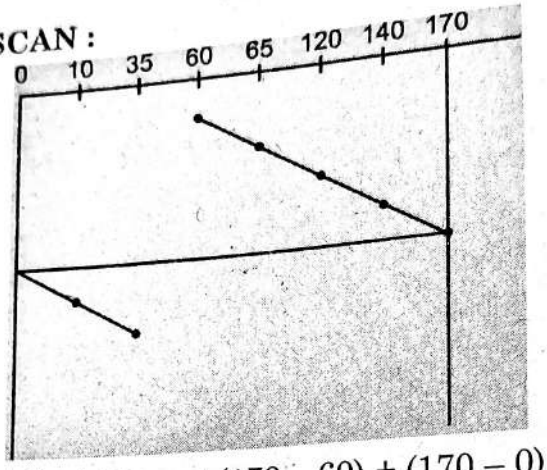
= 220 **(Ans.)**

(3) SCAN :



Total Seek Time = (60 – 0) + (170 – 0)
= 60 + 170
= 230 (Ans.)

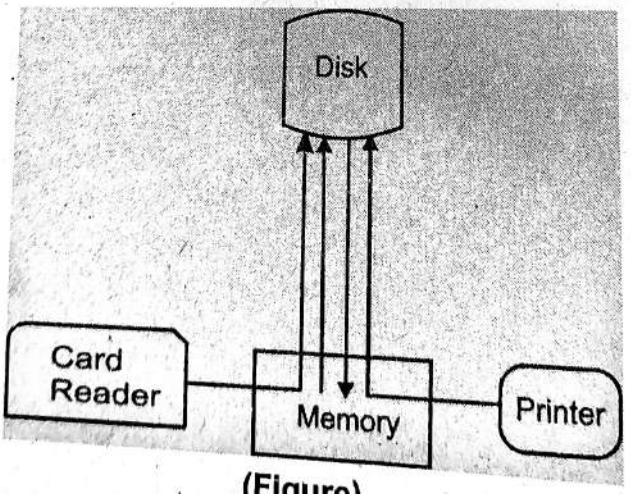
(4) **C-SCAN :**



Total Seek Time = (170 – 60) + (170 – 0) + (35 – 0)
= 110 + 170 + 35
= 315 (Ans.)

5. *What is Spooling? Draw a diagram and explain.* **(2018)**

Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices. An operating system does the following activities related to distributed environment :



(Figure)

(1) Handles I/O device data spooling as devices have different data access rates.

(2) Maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.

(3) Maintains

6. ◆ *Define*

◆ *Show*

◆ *Discuss*

◆ Write

◆ Give

(3) Maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashion. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task.

6. ◆ ***Define seek time and rotational latency of disk.*** **(2017)**

◆ ***Show disk structure pictorially. Find the total capacity of a disk based on the disk parameters.*** **(2017)**

◆ ***Discuss Disk (Secondary storage structure in detail). What is seeking time and latency time?*** **(2015)**

◆ ***Write short note on Disk structure.*** **(2014)**

◆ ***Give the detailed discussion of Disk Structure.***

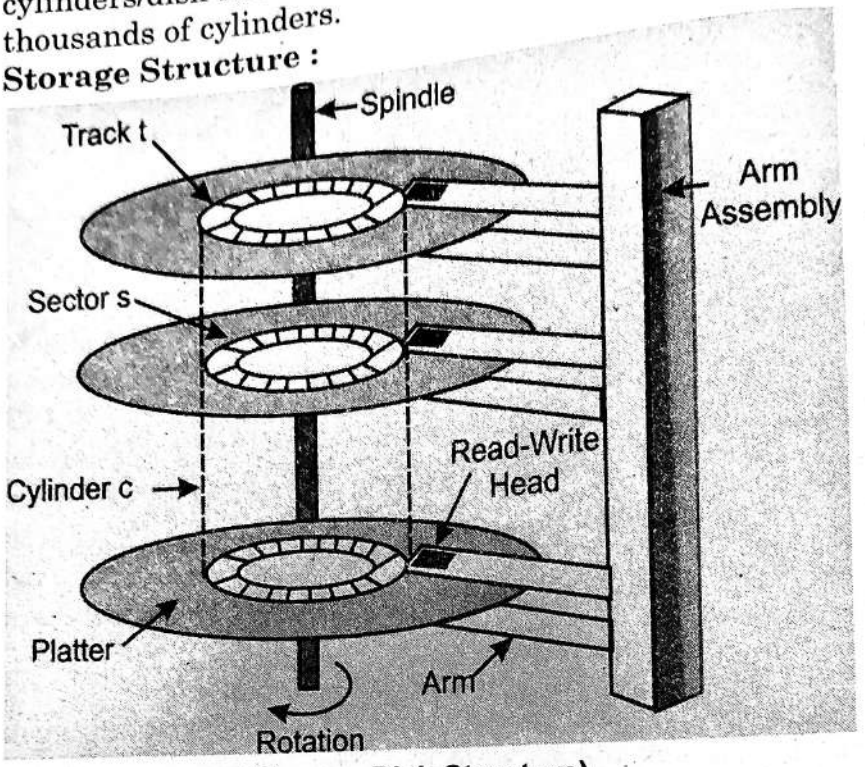
The disks provide the medium for secondary storage. We use disk storage system for transferring information from one system to another, and for storing quantities of data so large that they are impractical as disk storage. The disk drives are addresses as large one-dimensional array of logical blocks; these blocks are the smallest unit of transfer as about the size of 512 bytes. This one dimensional array of logical blocks is mapped on to the sectors of the disk sequentially starts from the zeroth sector.

The disk is the most reliable system for storing data. It is divided among the track and sectors in cylinder. The mapping process is performed between these tracks for accessing the data. By using this mapping, we can convert a logical block number into disk address that consists of a cylinder number, a track number and a sector number with in that track, but through this mapping we cannot explore the defective sectors and the constant numbers of sectors per block can also not find out. As we move from outer zone to inner, the numbers of sectors/track decreases. The drive increase its rotation speed as the head moves from outer to inner tracks to keep the same rate of data movement under the head.

The numbers of sectors per track has been increasing as disk technology improves and also the numbers of

cylinders/disk has been increasing large disks have tens of thousands of cylinders.

Storage Structure :



(Figure : Disk Structure)

Seek Time : Seek time is the time taken for a hard disk controller to locate a specific piece of stored data. Other delays include transfer time (data rate) and rotational delay (latency). When anything is read or written to a disc drive, the read/write head of the disc needs to move to the right position. The actual physical positioning of the read/write head of the disc is called seeking. The amount of time that it takes the read/write head of the disc to move from on part of the disk to another is called the seek time. The seek time can differ for a given disc due to the varying distance from the start point to where the read/write head has been instructed to go. Because of these variables, seek time is generally measured as an average seek time.

Latency Time : Latency is a time interval between the stimulation and response, or, from a more general point of view, as a time delay between the cause and the effect of some physical change in the system being observed.

Formula for Finding the Total Capacity :

Total capacity of the disk = number of recoding surface × tracks / surface × sectors / track × bytes / sector

7. *For a disk*
cylinder :
90, 50,
In tho
cylinder 5
with prope
disk schedu
(1) *FCFS*
(2) *SCAN*
(3) *C – SC*
Assume tot

(1) FCFS :

200
190
180
170
160
150
140
130
120
110
100
90
8
7
6

(2) SC

7. ***For a disk queue with request for I/O to blocks on cylinder :*** **(2017)**

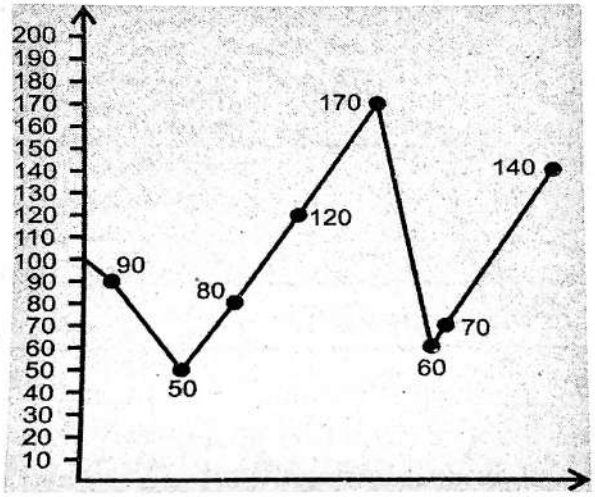
90, 50, 80, 120, 170, 60, 70, 140

In that order. If the disk head is initially at cylinder 55, find the total no. of head movement with proper schematic diagram for the following disk scheduling :

(1) ***FCFS***
(2) ***SCAN***
(3) ***C – SCAN***

Assume total no. of cylinder is 200.

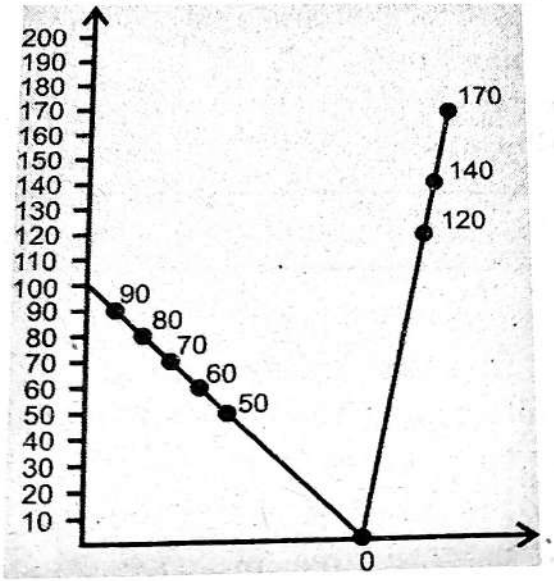
(1) FCFS : (90, 50, 80, 120, 170, 60, 70, 140)



(Figure)

= 10 + 40 + 30 + 40 + 50 + 110 + 10 + 70 = 360

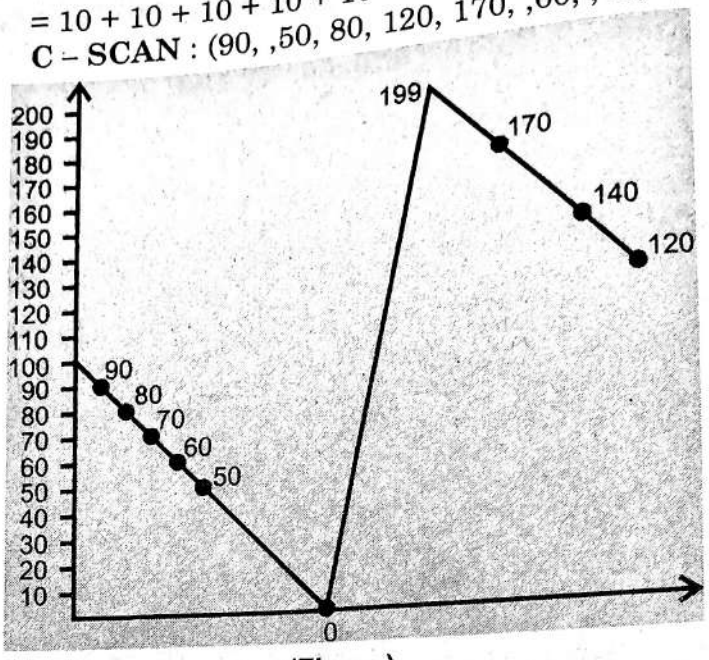
(2) SCAN : (90, ,50, 80, 120, 170, ,60, ,70, 140)



(Figure)

$= 10 + 10 + 10 + 10 + 10 + 120 + 20 + 30 = 220$

(3) C – SCAN : (90, ,50, 80, 120, 170, ,60, ,70, 140)



(Figure)

$= 10 + 10 + 10 + 10 + 10 + 199 + 29 + 30 + 20 = 328$

8. *What is device management?* **(2015)**

OS manages device communication via their respective drivers. Operating System does the following activities for device management.

(1) Keeps tracks of all devices.

(2) Program responsible for this task is known as the I/O controller.

(3) Decides which process gets the device when and for how much time.

(4) Allocates the device in the efficient way.

(5) De-allocates devices.

9. *Explain Directory structure.* **(2014)**

An organizational unit, or container, used to organize folders and files into a hierarchical structure. Directories contain bookkeeping information about files that are, figuratively speaking, beneath them in the hierarchy. You can think of a directory as a file cabinet that contains folders that contain files. Many graphical user interfaces use the term folder instead of directory.

Computer manuals often describe directories and file structures in terms of an inverted tree. The files and directories at any level are contained in the directory above them. To access a file, you may need to specify the names of all the directories above it. You do this by specifying a path. The top most directory in any file is called the root directory. A directory that is below another directory is called a subdirectory. A directory above a subdirectory is called the parent directory. Under DOS and Windows, the root directory is a back slash (\).

10. ***A disk has 19,456 cylinders, 16 heads and 63 sectors per track. The disk spins at 5400 rpm. Seek time between adjacent tracks is 2 μs. Assuming the read/write head is already positioned at track 0, how long does it take to read the entire disk?*** **(2014)**

Time to read the entire disk = $19456 \times 16 \times 63 / 5400 \times 2$
$= 1815.89\ \mu s$ **(Ans.)**

11. ***What is meant by 'interleaving' of disk blocks?*** **(May 2013)**

Interleaving is a process or methodology to make a system more efficient, fast and reliable by arranging data in a non contiguous manner. There are many uses for interleaving at the system level, including :

(1) **Storage :** As hard disks and other storage devices are used to store user and system data, there is always a need to arrange the stored data in an appropriate way.

(2) **Error Correction :** Errors in data communication and memory can be corrected through interleaving.

(3) Multi-Dimensional Data Structures.

12. ***What is spooling? Describe the following disk scheduling diagrammatically for a disk queue with request for I/O to block on cylinder in the order :***
95, 170, 37, 120, 14, 180, 65, 69 **(May 2015)**

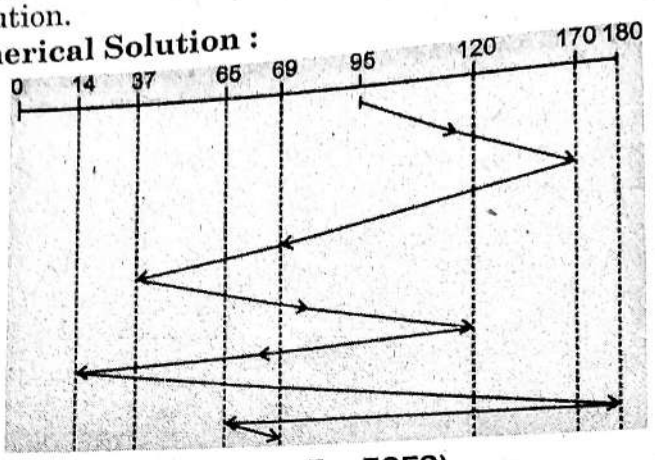
(1) ***For FCFS***

(2) ***SSTF***

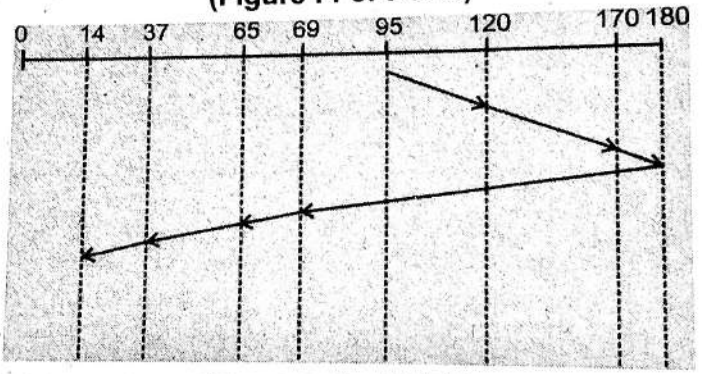
Spooling

Spooling is a process in which data is temporarily held to be used and executed by a device, program or the system. Data is sent to and stored in memory or other volatile storage until the program or computer requests it for execution.

Numerical Solution :



(Figure : For FCFS)



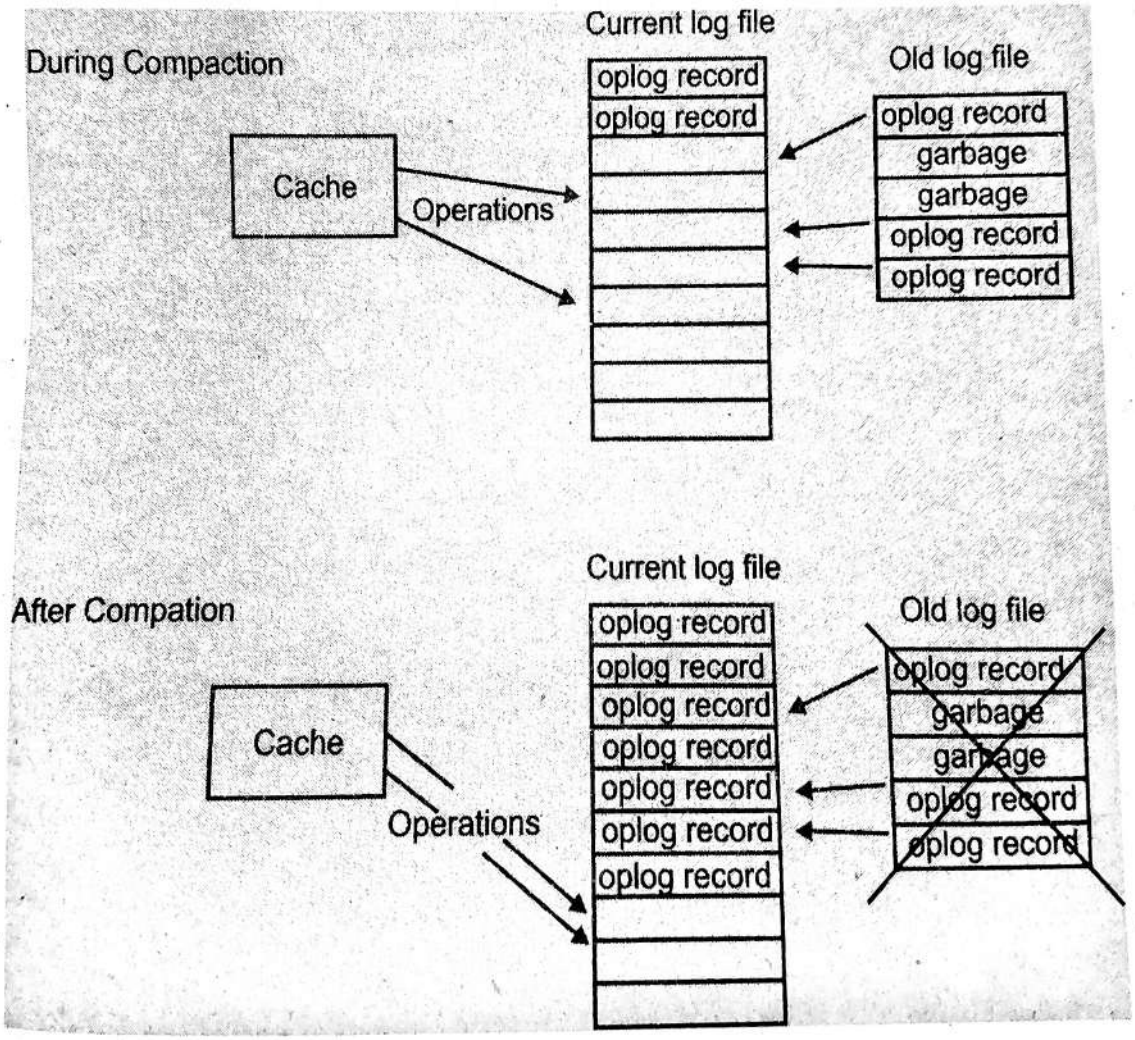
(Figure : For SSTF)

13. *What is meant by 'Compaction' of disk? Why is it required to be done?* **(May 2013)**

When a cache operation is added to a disk store, any preexisting operation record for the same entry becomes obsolete, and marks it as garbage. For example, when we create an entry, the create operation is added to the store. If we update the entry later, the update operation is added and the create operation becomes garbage. we does not remove garbage records as it goes, but it tracks the percentage of garbage in each operation log, and provides mechanisms for removing garbage to compact your log files.

We compacts an old operation log by copying all non-garbage records into the current log and discarding the old files. As with logging, operation logs are rolled as needed during compaction to stay within the max operation log setting. We can configure the system to automatically compact any closed operation log when its garbage content reaches a certain percentage. We can also manually request compaction for online and offline disk stores. For the online disk store, the current operation log is not available for compaction, no matter how much garbage it contains.

Log File Compaction for the Online Disk Store



(Figure)

Offline compaction runs essentially in the same way, but without the incoming cache operations. Also, because there is no current open log, the compaction creates a new one to get started.

14. *Write short note on Buffering.*

Buffering

(1) Buffering is used in direct and indirect communication. Messages exchanged by communicating reside in a temporary queue.

(2) Buffering is implemented in three ways :

(a) Zero capacity

(b) Bounded capacity

(c) Unbounded capacity

(a) **Zero Capacity :** Maximum length of the queue is 0 (zero). Communication link cannot have any messages waiting in it. In this case, the sender must block until the recent receives message.

(b) **Bounded Capacity :** In bounded capacity, the queue has a finite length i.e. '*n*'. Almost '*n*' messages can reside in it. If the queue is full, it discards the message and sender is blocked until space is available in the queue.

(c) **Unbounded Capacity :** The queue has potentially infinite length. Any number of messages can wait in it. The sender never blocks.

❑❑

1. *Explain convoy effect with example.* **(2023-24)**

The convoy effect, also known as the convoy phenomenon, refers to a situation where multiple tasks or processes are delayed due to the execution of a long-running task or process, even if they are unrelated to each other. This delay occurs because the long-running task occupies shared resources such as the CPU, memory, or I/O channels for an extended period, preventing other tasks from accessing these resources in a timely manner.

Example : A classic example to illustrate the convoy effect is in a computer operating system where multiple processes are waiting to access a shared resource, such as a printer. Consider the following scenario:

Let's say there are three processes: P1, P2, and P3, all trying to print their respective documents. Each process sends a request to the operating system to use the printer.

Process P1 sends its request to print its document.

The operating system grants access to the printer to P1, and P1 starts printing.

Meanwhile, P2 and P3 are waiting in a queue to use the printer.

However, P1's document is very large, so it takes a significant amount of time to print.

While P1 is still printing, P2 and P3 are blocked and cannot proceed until P1 finishes printing.

P2 and P3 experience delays, even though they are unrelated to P1's printing task.

In this scenario, the long-running task (P1's printing) causes a convoy effect, where other processes (P2 and P3) are delayed due to resource contention. Even if P2 and P3's printing tasks are small and could have been completed quickly, they are forced to wait for the completion of P1's task because they all share the same limited resource (the printer).

The convoy effect can lead to inefficiencies in system performance and can be particularly problematic in real-time systems where timely execution of tasks is critical. To

mitigate the convoy effect, techniques such as resource scheduling algorithms and resource allocation strategies are employed to ensure fair and efficient resource utilization among competing tasks.

2. *What are the advantages (any three) and disadvantages (any three) of Contiguous Allocation?* (2023-24)

Advantages of Contiguous Memory Allocation

(1) **Simplicity :** Contiguous memory allocation is relatively simple to implement compared to other memory allocation techniques such as paging or segmentation. It requires minimal overhead for memory management.

(2) **Efficient Memory Access :** Since memory is allocated contiguously, accessing memory locations within a process is straightforward and efficient. This can lead to faster program execution times.

(3) **Less Fragmentation :** Contiguous allocation tends to produce less fragmentation compared to other allocation methods such as dynamic partitioning. External fragmentation, which occurs when memory is divided into small non-contiguous blocks, is minimized.

(4) **Direct Addressing :** Contiguous allocation allows for direct addressing of memory locations. This simplifies memory access, as the physical address of a memory location can be calculated directly using a base address and an offset.

Disadvantages of Contiguous Memory Allocation :

(1) **Limited Flexibility :** Contiguous memory allocation imposes restrictions on the size of processes that can be accommodated in memory. If the available contiguous block of memory is smaller than the size of the process, the process cannot be loaded.

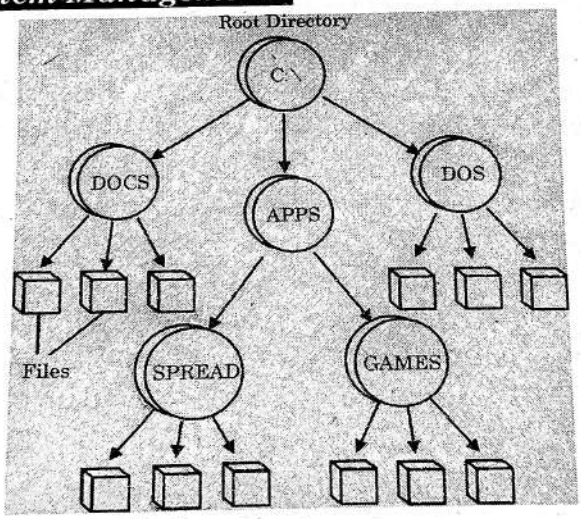
(2) **Fragmentation :** While contiguous allocation reduces external fragmentation, it may still suffer from internal fragmentation. This occurs when the allocated memory block is larger than necessary, resulting in wasted memory space.

(3) **Difficulty in Memory Management :** Managing memory in a contiguous allocation system can be challenging, especially when dealing with variable-sized processes. It may require complex algorithms for memory allocation and deallocation to minimize fragmentation and efficiently utilize available memory.

(4) **Low Memory Utilization :** Contiguous allocation may lead to inefficient memory utilization, especially if the memory is fragmented or if there are many small processes. This can result in a significant amount of wasted memory.

3. ◆ ***Define and explain the concept of the file system.*** **(2023-24)**
   ◆ ***Write a short note on 'File Management'. Write down the goals of file system.***

File System Management



(Figure)

Also referred to simply as a file system. The system that an operating system or program uses to organize and keep track of files. For example, a hierarchical file system is one that uses directories to organize files into a tree structure. Although the operating system provides its own file management system. These systems interact smoothly with the operating system but provide more features, such as improved backup procedures and stricter file protection.

Types of File Systems

File system types can be classified into disk file systems, network file systems and special purpose file systems.

Disk File Systems : A disk file system is a file system designed for the storage of files on a data storage device, most commonly a disk drive, which might be directly or indirectly connected to the computer.

Flash File Systems : A flash file system is a file system designed for storing files on flash memory devices

Database File Systems : A new concept for file management is the concept of a database-based file system. Instead of, or in addition to, hierarchical structured management, files are identified by their characteristics, like type of file, topic, author, or similar metadata.

Special Purpose File Systems : A special purpose file system is basically any file system that is not a disk file system or network file system. This includes systems where the files are arranged dynamically by software, intended for such purposes as communication between computer processes or temporary file space. Special purpose file systems are most commonly used by file-centric operating systems such as UNIX. Examples include the procfs (/proc) file system used by some UNIX variants, which grants access to information about processes and other operating system features. Deep space science exploration craft, like Voyager I & II used digital tape-based special file systems. Most modern space exploration craft like Cassini-Huygens used real-time operating system file systems or RTOS influenced file systems. The Mars Rovers are one such example of an RTOS file system, important in this case because they are implemented in flash memory.

4. *Explain the concept of file system protection and security.* **(2023-24)**

File protection in an operating system refers to the various mechanisms and techniques used to secure files from unauthorized access, alteration, or deletion. It involves controlling access to files, ensuring their security and confidentiality, and preventing data breaches and other security incidents.

Operating systems provide several file protection features, including file permissions, encryption, access control lists, auditing, and physical file security. These measures allow administrators to manage access to files, determine who can access them, what actions can be performed on them, and how they are stored and backed up. By implementing file protection measures, organizations can safeguard their files, maintain data confidentiality, and minimize the risk of data breaches and other security incidents.

Type of File Protection : File protection is an essential component of modern operating systems, ensuring that files are secured from unauthorized access, alteration, or deletion. In this context, there are several types of file protection mechanisms used in operating systems to provide robust data security.

(1) **File Permissions :** File permissions are a basic form of file protection that controls access to files by setting permissions for users and groups. File permissions allow the system administrator to assign specific access rights to users and groups, which can include read, write, and execute privileges. These access rights can be assigned at the file or directory level, allowing users and groups to access specific files or directories as needed. File permissions can be modified by the system administrator at any time to adjust access privileges, which helps to prevent unauthorized access.

(2) **Encryption :** Encryption is the process of converting plain text into ciphertext to protect files from unauthorized access. Encrypted files can only be accessed by authorized users who have the correct encryption key to decrypt them. Encryption is widely used to secure sensitive data such as financial information, personal data, and other confidential information. In an operating system, encryption can be applied to individual files or entire directories, providing an extra layer of protection against unauthorized access.

(3) **Access Control Lists (ACLs) :** Access control lists (ACLs) are lists of permissions attached to files and directories that define which users or groups have access to them and what actions they can perform on

them. ACLs can be more granular than file permissions, allowing the system administrator to specify exactly which users or groups can access specific files or directories. ACLs can also be used to grant or deny specific permissions, such as read, write, or execute privileges, to individual users or groups.

(4) **Auditing and Logging :** Auditing and logging are mechanisms used to track and monitor file access, changes, and deletions. It involves creating a record of all file access and changes, including who accessed the file, what actions were performed, and when they were performed. Auditing and logging can help to detect and prevent unauthorized access and can also provide an audit trail for compliance purposes.

(5) **Physical File Security :** Physical file security involves protecting files from physical damage or theft. It includes measures such as file storage and access control, backup and recovery, and physical security best practices. Physical file security is essential for ensuring the integrity and availability of critical data, as well as compliance with regulatory requirements.

5. *What are the various file extensions? Explain.* **(2023-24)**

In an operating system, file extensions are typically used to denote the type or format of a file. They are usually appended to the end of a filename and separated by a dot (.), indicating the file's content or the program that can open or interpret it. Here are some common file extensions found in operating systems along with explanations of their purposes :

(1) **.txt (Text File) :** Text files contain plain text without any special formatting or styling. They are commonly used for storing textual information such as notes, code, configuration files, and documentation.

(2) **.doc/.docx (Microsoft Word Document) :** These file extensions are associated with documents created using Microsoft Word. They can contain formatted text, images, tables, and other elements. .docx is the newer XML-based format introduced with Microsoft Word 2007.

(3) .pdf (P
used fo
that pr
differen
for docu

(4) .xls/.xl
file ex
using
organiz
formula
newer

(5) .ppt/.p
Presen
present
They c
and mu
based f

(6) .jpg/.jp
format
compre
image
photog

(7) .png
popul
lossl
com
logo

(8) .gi
for
cli
ba
a

(9)

(10)

(3) **.pdf (Portable Document Format) :** PDF files are used for creating and sharing documents in a format that preserves the layout, fonts, and graphics across different platforms and devices. PDF is widely used for documents such as manuals, reports, and forms.

(4) **.xls/.xlsx (Microsoft Excel Spreadsheet) :** These file extensions are used for spreadsheets created using Microsoft Excel. They contain tabular data organized into rows and columns, along with formulas, charts, and other features. .xlsx is the newer XML-based format introduced with Excel 2007.

(5) **.ppt/.pptx (Microsoft PowerPoint Presentation) :** These file extensions are used for presentations created using Microsoft PowerPoint. They contain slides with text, images, animations, and multimedia elements. .pptx is the newer XML-based format introduced with PowerPoint 2007.

(6) **.jpg/.jpeg (JPEG Image) :** JPEG is a common file format for storing digital images. It uses lossy compression to reduce file size while preserving image quality. JPEG images are widely used for photographs, web graphics, and other visual content.

(7) **.png (Portable Network Graphics) :** PNG is a popular file format for storing raster graphics with lossless compression. It supports transparency and is commonly used for images with sharp edges, such as logos, icons, and illustrations.

(8) **.gif (Graphics Interchange Format) :** GIF is a file format for storing animated images and short video clips. It supports animation and transparent backgrounds, making it suitable for simple animations, memes, and social media content.

(9) **.mp3 (MPEG Audio Layer III) :** MP3 is a widely-used audio file format for storing compressed audio data. It offers high-quality audio playback with relatively small file sizes, making it popular for music, podcasts, and other audio content.

(10) **.mp4/.avi/.mov (Video File) :** These file extensions are used for storing video files in various formats, such as MP4 (MPEG-4), AVI (Audio Video Interleave), and MOV (QuickTime Movie). They contain compressed video and audio data for playback on multimedia devices and platforms.

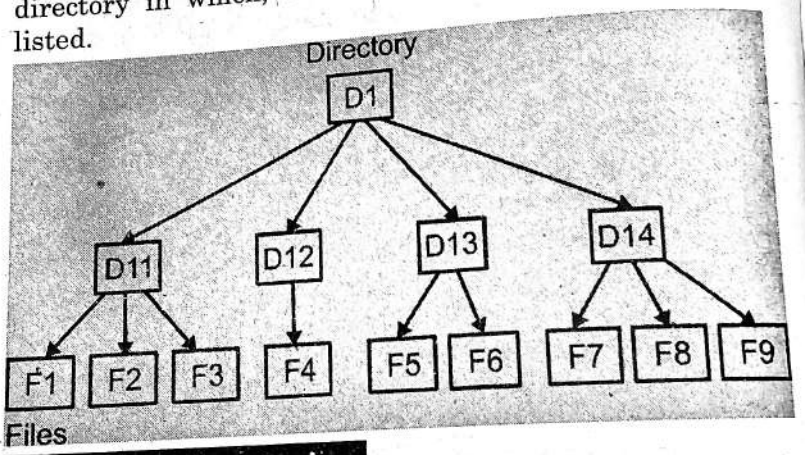
These are just a few examples of common file extensions found in operating systems. There are many more file extensions used for different types of files and purposes, each serving a specific function or representing a particular file format.

6. ◆ ***Define Directory. Explain all types of directories.*** **(2023-24)**

◆ ***Draw and explain various levels of directory structures.*** **(2018)**

◆ ***How can a directory be implemented in File Systems?***

Directory : A directory is a container that is used to contain folders and files. It organizes files and folders in a hierarchical manner. Each partition must have at least one directory in which, all the files of the partition can be listed.



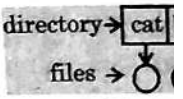
Types of Directories

The file systems can be extensive to manage all these data we need to organize them in partitions of disk that provide several separate areas within one disk treated as a separate storage device. Each partition contains information about files within it. This information is kept in entries in a device directory that records information such as name, location, size and type for all files on that partition.

The directory can be viewed as a symbol table that translates file names into their directory entries. There are some operations can be performed on directory such as : Create a File, Delete a File, List a directory, Rename a file

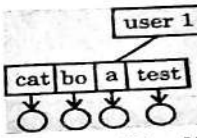
Traverse the File Sys
is many types as :

Single Level Direct
the same directory. I
between different use



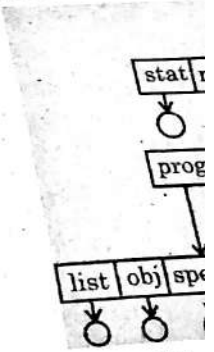
(Figur

Two-Level Directo
user has own User F
of a single user, whe
system's master file
user name and acco
the UFD for that us



(Figure : MFD = M
U

Tree-Structured
directory system i
has a common roo



(Fig

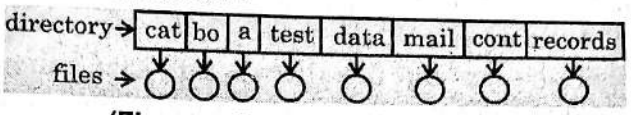
Ever
path nar
subdirec

7. Define

F
inform
to bo

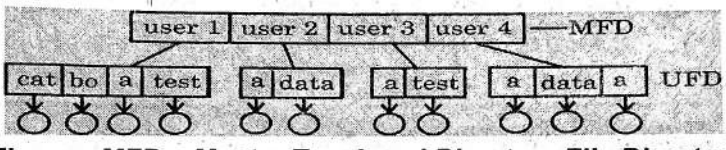
Traverse the File System. The logical structure of directory is many types as :

Single Level Directory : In this, all files are contained in the same directory. It often leads to confusion of file names between different users :



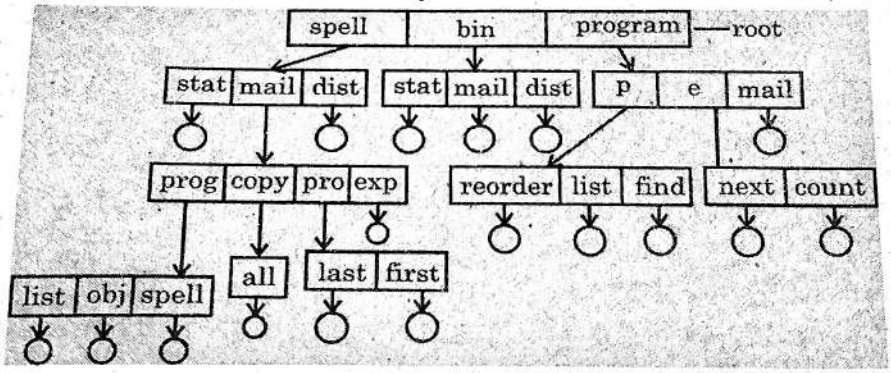
(Figure : Single-Level Directory)

Two-Level Directory : In the two-level directory, each user has own User File Directory (UFD) lists only the files of a single user, when a user job starts or a user logs in, the system's master file system is searched that is indexed by user name and account number, and each entry points to the UFD for that user :



(Figure : MFD = Master Two-Level Directory File Directory UFD = User File Directory)

Tree-Structured Directory : The tree structured directory system is the most common directory structure. It has a common root directory.



(Figure : Tree-Structured Directory System)

Every file in the system has unique path name. A path name is the path from the root, through all the subdirectories, to a specified file as shown by diagram.

7. *Define file and explain file attributes.* (2022-23)

Files are used for all input and output (I/O) of information in the operating system, to standardize access to both software and hardware. Input occurs when the

contents of a file is modified or written to. Output occurs when the contents of one file is read or transferred to another file.

File Attributes : A file attribute is an element that describes a characteristic of a file and provides information the system needs to handle the file. Examples of file attributes are the file title, record size, number of areas, and date of creation. For disk files, permanent file attribute values are stored in the disk file header.

8. ◆ ***What is contiguous memory allocation? Explain the hardware support for memory protection.*** **(2022-23)**
   ◆ ***Discuss the contiguous and linked allocation methods for disk space.*** **(2017)**
   ◆ ***Discuss different file allocation methods.*** **(2015)**

Contiguous Allocation

Requires that each file occupy a set of contiguous blocks on the disk. Disk addresses define a linear ordering on the disk with this ordering, assuming that only one job is accessing the disk, accessing block b+1 after block b normally requires no head movement. When head movement is needed, the head need only move from one track to the next. Thus, the number of disk seeks required for accessing continuously allocated files is minimal, as is seek time when seek is finally needed.

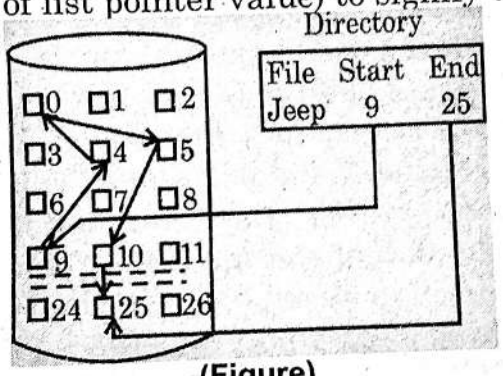
Contiguous allocation of a file is defined by the disk address and length of the first block. It the file is n blocks long and starts at location b, them it occupies block b, b+1, b+2..., (b+n-1). The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file. One advantage of contiguous allocation is that all successive records of a file are normally physically adjacent to each other. This increases the accessing speed of records.

Indexed Allocation : Linked allocation solves the external fragmentation and size – declaration problems of contiguous allocation. However in the absence of FAT, linked allocation cannot support efficient direct access, since, the pointers to the blocks are scattered with the

blocks themselves all over the disk and must be retrieved in order. Indexed allocation solves this problem by bringing all the pointers together into one location, the index block.

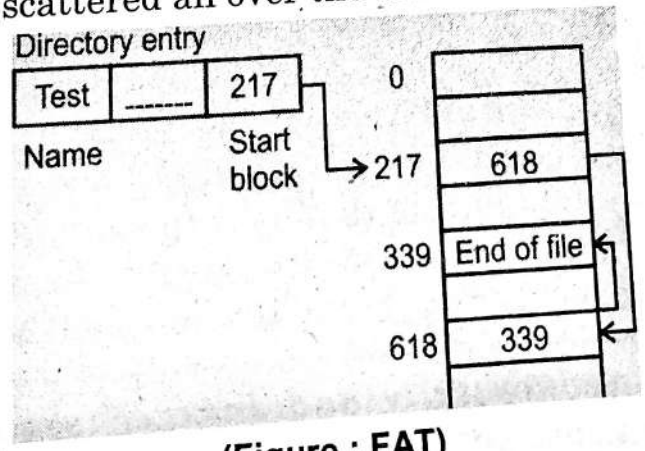
Each file has its own index block, which is an array of disk – block addresses. The i^{th} entry in the index block points to the i^{th} block of the file. The directory contains the address of the index block. To find and read the i^{th} block, we use the pointer in the i^{th} index block entry. One disadvantage with linked allocation method is that it does not support direct accessing since, blocks are scattered all over the disk. This problem is solved by indexed allocation by placing all of the pointers together into an index block.

Linked Allocation : It solves all the problems of contiguous allocation with linked allocation, each file is a linked list of disk blocks. The directory contains a pointer to the first and last blocks of file. This pointer is initialized to null (end of list pointer value) to signify the empty file.



(Figure)

The major problem is that it can be used effectively only for sequential-access files. Another problem is the space required for the pointers. The usual solution to this problem is to collect blocks into multiples, called clusters and to allocate the clusters rather than blocks. Another problem is reliability. Since the files are linked together by pointers scattered all over the disk.



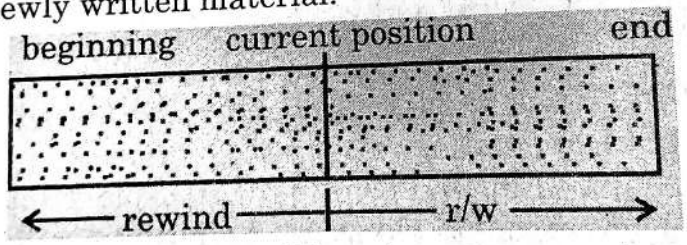
(Figure : FAT)

An important variation on the linked allocation method is the use of File Allocation Table (FAT). The FAT is used much as is a linked list. The table entry indexed by that block number then contains the block number to the next block in the file.

9. ◆ ***Discuss the access methods and file structure.*** **(2022-23)**
◆ ***Name the various file access methods with explanation.*** **(2018)**
◆ ***Name the different file access methods and describe each in brief.*** **(2017)**
◆ ***Discuss sequential access and direct access methods of file management.*** **(2015, 2016)**

Files store information that must be accessed and read into computer memory by several ways such as :

Sequential Access : The most simplest access method. In this, the information of the file is processed in order. As the bulk of the operations on a file is read and writes one by next portion of the file and automatically advances a file pointer, which tracks the input/output location and appends to the end of the file and advances to the end of the newly written material.



(Figure)

Direct Access : In direct access method, there is no order of storage of files. It allows arbitrary blocks to be read or write. This type of access method is very useful for immediate access to large amount of information.

Index Sequential Access Methods : Other access methods can be built on top of a direct access method such as index system having the index (pointers) to the various blocks. To find a record, we first search the index and then by using this index reaches to the desired record.

10. ***Discuss sequentially and Indexed sequential file organization methods.*** **(2015)**

Sequential

A sequ
order they w
The records
blocks with
Records in
sequentially
made short
can be upda
by replacin
records will

If the
sequential
records you
report prin
prefer to do

Indexed-S

improved
structure is
are pairs,
position in
subset of
data file,
records.

Thi
is compa
key com
is perfo
until th
to by t
file ac
the a
sequ
sche
itsel
ind
hie
to
th

to
t
I

Sequential Organization

A sequential file contains records organized in the order they were entered. The order of the records is fixed. The records are stored and sorted in physical, contiguous blocks within each block the records are in sequence. Records in these files can only be read or written sequentially. Once stored in the file, the record cannot be made shorter, or longer, or deleted. However, the record can be updated if the length does not change. (This is done by replacing the records by creating a new file.) New records will always appear at the end of the file.

If the order of the records in a file is not important, sequential organization will suffice, no matter how many records you may have. Sequential output is also useful for report printing or sequential reads which some programs prefer to do.

Indexed-Sequential Organization : Key searches are improved by this system too. The single-level indexing structure is the simplest one where a file, whose records are pairs, contains a key pointer. This pointer is the position in the data file of the record with the given key. A subset of the records, which are evenly spaced along the data file, is indexed, in order to mark intervals of data records.

This is how a key search is performed : the search key is compared with the index keys to find the highest index key coming in front of the search key, while a linear search is performed from the record that the index key points to, until the search key is matched or until the record pointed to by the next index entry is reached. Regardless of double file access (index + data) required by this sort of search, the access time reduction is significant compared with sequential file searches. Hierarchical extension of this scheme is possible since an index is a sequential file in itself, capable of indexing in turn by another second-level index, and so forth and so on. And the exploit of the hierarchical decomposition of the searches more and more, to decrease the access time will pay increasing dividends in the reduction of processing time.

There is however a point when this advantage starts to be reduced by the increased cost of storage and this in turn will increase the index access time. Hardware for Index-Sequential Organization is usually disk-based;

rather than tape. Records are physically ordered by primary key. And the index gives the physical location of each record. Records can be accessed sequentially or directly, via the index. The index is stored in a file and read into memory at the point when the file is opened. Also, indexes must be maintained.

11. *Write short note on File allocation methods.* (2014)

File Allocation Methods

(1) Contiguous allocation.
(2) Linked allocation.
(3) Indexed allocation.
(4) Multilevel indexed allocation.
(5) Combined scheme.

12. *What is Frame Allocation and why it is used?*

Allocation of Frames

This is the scheme used to allocate the fixed amount of free memory among the various processes. The simplest case of virtual memory is the single user system with the 128 *KB* memory composed of pages of size 1 *KB*, thus there are 128 frames. The operating system takes 35 *KB*, leaving 93 frames for the user process. Under pure demand paging, all 93 frames would initially be put on the free frame list. When a user process started execution would generate a sequence of page faults. The first 93 page faults would all get free frames from the free-frame list. When the free-frame list was exhausted, a page replacement algorithm would be used to select on the 93 in memory pages replaced with 94^{th} and so on. There are many variations on this as we can require that the operating system allocate its entire buffer and table space form the free-frame list. When this space is not in use by the operating system, it can be used to support user paging. We can try to keep 3 free frames reserved on the free frame list at all time. Thus when a page fault occurs, there is a free frame available to page into, while the page swap is taking place, a replacement can be selected, which is then written to the disks as the user process continues to execute.

❑❑

BCA

(SEM. IV) EXAMINATION, MAY, 2018
BCA – 402 (N) : OPERATING SYSTEM

Time : Three Hours *Maximum Marks : 75*

Note : Attempt questions from *all* Sections as directed.
Inst. : The candidates are required to answer only in serial order. If there are many parts of a question, answer them in continuation.

SECTION – A
(Short Answer Type Questions)

Note : *All* questions are compulsory. Each question carries 3 marks.

1. (A) Define operating system. Describe briefly the kind of services provided by an operating system.
(B) What are the main functions of memory management?
(C) Name the various file access methods with explanation.
(D) What is Spooling? Draw a diagram and explain.
(E) Discuss shared devices and virtual devices for device management.
(F) Write the advantages and disadvantages of demand paging.
(G) Write the comparison between paging and segmentation.
(H) Differentiate the logical and physical address with an example.
(I) What is the role of critical section in process synchronization?

SECTION – B
(Long Answer Type Questions)

Note : Attempt *any two* questions. Each question carries 12 marks.

2. Consider the following set of jobs with their arrival time, execution time (in minutes) :

Job	Arrival Time	Execution Time
A	0	5
B	1	15
C	3	12
D	7	25
E	10	5

Calculate the average waiting time for FCFS, SJF preemptive, SJF non – preemptive.

3. (a) Write the name of disk scheduling algorithms. Write the method and explain the working of any three methods.
(b) Draw and explain various levels of directory structures.

4. (a) What is the fragmentation problem? Describe internal and external fragmentation.
(b) Explain the terms mutual exclusion, hold and wait, preemption and circular wait in deadlocks with examples.

SECTION – C
(Long Answer Type Questions)

Note : Attempt *any two* questions. Each question carries 12 marks.

5. (a) Consider the following current resource allocation state :

Process	Allocation			Maximum			Available		
	A	B	C	A	B	C	A	B	C
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	3			

Answer the following questions using Banker's algorithm :

(i) What is the content of the Need Matrix?

(ii) Is the system in safe state?

6. (a) Draw and explain Process Control Block (PCB).

(b) Write the name of various page replacement algorithms. Explain any two methods.

7. (a) Define and explain the concept of virtual memory. Write the advantages of using virtual memory.

(b) Write a short note on Resource Allocation graph.

❑❑

BCA

(SEM. IV) EXAMINATION, MAY, 2019
BCA – 402 (N) : OPERATING SYSTEM

Time : Three Hours *Maximum Marks : 75*

Note : Attempt questions from ***all*** Sections as directed.
Inst. : The candidates are required to answer only in serial order. If there are many parts of a question, answer them in continuation.

SECTION – A
(Short Answer Type Questions)

Note : *All* questions are compulsory. Each question carries 3 marks.

1. (A) Differentiate between hard real time system and soft real time system.
   (B) Explain Belady's anomaly.
   (C) What do you understand by synchronization Hardware?
   (D) How jobs are scheduled in multiple processor system?
   (E) What are four necessary conditions responsible for deadlocks?
   (F) Discuss when a resource allocation graph contains a cycle but no deadlock.
   (G) How the logical address is converted to physical address? Explain.
   (H) What do you understand by internal fragmentation? How this can be resolved?
   (I) What is a hash table? How is it used?

SECTION – B
(Long Answer Type Questions)

Note : Attempt *any two* questions. Each question carries 12 marks.

2. (a) Discuss Second chance page replacement algorithm. **6**
   (b) For given reference string, calculate the number of page faults using optimal page replacement algorithm. Given page size = 3. **6**
   7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

3. (a) What is thrashing? Explain causes of thrashing in detail. Also suggest its solution. **8**
   (b) Differentiate between segmentation and paging. **4**

4. (a) Differentiate between Asymmetric multiprocessing and Symmetric multiprocessing. **4**
   (b) Consider the following set of processes with the length of the CPU burst given in milliseconds : **8**

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

The processes are assumed to have arrived in the order, P_1, P_2, P_3, P_4, P_5, all at time O.

(i) Draw four Gantt Charts that illustrate the execution of these processes using the following scheduling algorithms : FCFS, SJF, non – pre – emptive priority (taking smaller priority number implies a higher priority) and RR (quantum = 1).

(ii) What is the turnaround time of each process for each of the scheduling algorithms in part (i)?

(iii) What is the waiting time of each process for each of the scheduling algorithm in part (i)?

(iv) Which of the Algorithm in part (i) a results in the minimum average waiting time (over all processes)?

5. (a) What do you understand by Critical Section problem? What requirements must be satisfied by its solution? **6**

(b) What is the Dining – Philosophers problem Suggest its solution? **6**

SECTION – C
(Long Answer Type Questions)

Note : Attempt *any two* questions. Each question carries 12 marks.

6. (a) The long–term scheduler controls the degree of multiprogramming. Justify by giving examples. Also discuss the need of mid- term scheduler. **6**

(b) Discuss the Disk – space management methods. **6**

7. Suppose the head of moving disk is currently serving at request at track 60. If the queue of request is kept in FIFO order, what is the total head movement to satisfy these requests for the following disk scheduling algorithms :

(i) FCFS
(ii) SSTF
(iii) SCAN
(iv) C–SCAN

Disk request come into the disk drives for cylinder 65, 170, 35, 120, 10, 140 in that order. **12**

8. (a) Discuss Banker's Algorithm. **4**

(b) Consider the following snapshot of a system :

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

answer the following questions using the Banker's algorithm :

(i) What is the content of the matrix need? **1**

(ii) It the system in a safe – state? **4**

(iii) If a request from process P1 arrives for (0, 4, 2, 0) can be request granted immediately. **3**

❑❑

BCA

(SEM. III) EXAMINATION, 2022-23

BCA – 3003 : OPERATING SYSTEM

Time : 2 Hours *Maximum Marks : 75*

Note : This paper consists of three Sections A, B and C. Carefully read the instructions of each Section in solving the questions paper. Candidates have to write their answers in the given answer-copy only. No separate answer copy (B-Copy) Will be provided.

SECTION – A
(Short Answer Type Questions)

Note: All questions are compulsory. Answer the following questions as short answer type questions. Each question carries 5 marks.

1. (A) Define operating systems and discuss its role from different perspectives.
   (B) Write short note on following
   (i) Simple Batch Systems.
   (ii) Parallel Systems.
   (C) Explain the components of operating system.
   (D) Explain three requirements that a solution to critical-section problem must satisfy.
   (E) Explain the resource allocation graph.
   (F) What is demand paging? Explain.
   (G) Define file and explain file attributes.
   (H) Discuss dedicated devices and shared devices.
   (I) Differentiate between pre-emptive and Non-preemptive Scheduling.

SECTION – B
(Long Answer Type Questions)

Note: This section contains four questions from which one question is to be answered as long question. Each question carries 15 marks.

2. What do you mean by PCB? Where is it used? What are its contents? Explain.

Or

3. Explain paging scheme of memory management. What hardware support is needed for its implementation?

Or

4. What is contiguous memory allocation? Explain the hardware support for memory protection.

Or

5. Given memory partitions of 100 KB, 500 KB, 200 KB, 300 KB and 600 KB (in order), how would each of the first-fit, best-fit and worst-fit algorithms place processors of 212 KB, 417 KB, 112 KB and 426 KB (in-order)? Which algorithm makes the most efficient use of memory?

SECTION – C
(Long Answer Type Question)

Note: This section contains four questions from which one question is to be answered as long question. Each question carries 15 marks.

6. What is a process? Draw and explain process state diagram.

Or

7. (a) Explain four necessary conditions for deadlock.
   (b) Consider the following snapshot of the system:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Answer the following questions using the Banker's algorithm :

(i) What is the content of the Matrix Need?
(ii) Is the system in a safe state?

Or

8. (a) Discuss the properties of different CPU scheduling algorithms.
   (b) Consider the following set of processes, with burst time and arrival time.

Process	Burst Time	Arrival Time
P_1	19	0
P_2	14	1
P_3	11	2
P_4	6	3

Calculate the turnaround time and waiting time for both pre-emptive and non-preemptive scheduling using shortest job first algorithm.

Or

9. (a) Discuss the access, method and file structure.
   (b) Discuss the following page replacement algorithm with an example :
   (i) Optimal
   (ii) LRU.

❑❑

BCA

(SEM. III) EXAMINATION, 2023-24
BCA – 3003 : OPERATING SYSTEM

Time : Two Hours *Maximum Marks : 75*

Note : This paper consists of three Section A, B and C. Carefully read the instructions of each Section in solving the question paper. Candidates have to write their answers in the given answer-copy only. No separate answer – copy (B copy) will be provided.

SECTION – A
(Short Answer Type Questions)

Note : *All* questions are compulsory. Answer the following questions as short answer type questions. Each question carries 5 marks.

1. (A) Write the comparison between Hard Real Time and Soft Real Time (any three).
(B) What is the purpose of system call?
(C) Explain convoy effect with example.
(D) Distinguish between batch systems and time sharing systems.
(E) Write the comparison between turnaround time and completion time.
(F) Distinguish between preemptive and non preemptive Scheduling.
(G) Define Starvation in deadlock.
(H) Explain Demand Paging.
(I) List two differences between logical and physical addresses.

SECTION – B
(Long Answer Type Questions)

Note : This section contains four questions from which *one* question is to be answered as long question. Each question carries 15 marks.

2. Write short note on the following :
(a) Thrashing
(b) Deadlock detection
(c) Segmentation

Or

3. (a) Write the difference between long term scheduler and short term scheduler.
(b) Define and explain page replacement algorithms.

OR

4. Consider the following page reference string :
1, 2, 3, 4, 2, 1, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6 Identify the no. of page faults would occur for the following page replacement algorithms, assuming three frames are initially empty.
(a) LRU replacement
(b) Optimal replacement

Or

5. Calculate Average waiting time by using Shortest Remaining Time First (SRTF) algorithm.

Process No.	A.T.	B.T.
P1	0	2
P2	1	5
P3	2	1
P4	3	2
P5	3	3
P6	6	6

SECTION – C
(Long Answer Type Questions)

Note : This section contains four questions from which *one* question is to be answered as long question. Each question carries 15 marks.

6. (a) What are the various disk-scheduling algorithms? Explain.
   (b) What are the advantages (any three) and disadvantages (any three) of Contiguous Allocation?

Or

7. (a) Define and explain the concept of the file system.
   (b) Explain the concept of file system protection and security.

Or

8. (a) What are the various file extensions? Explain.
   (b) Define Directory. Explain all types of directories.

Or

9. Given the following queue – 95, 180, 34, 119, 11, 123, 62, 64 with the Red – write head initially at the track 50 and the tail track being at 199.
   Calculate the Total Head movements by using :
   (a) First Come First Serve (FCFS)
   (b) Shortest Seek Time First (SSTF)

❑❑