* Operating System :-

Ans-11 An operating System is a software that manages hardware components, software resources and provides basic services to application programs.

⇒ An operating System acts as an interface between the computer hardware and the user.

⇒ Types of Operating System :-

(1) Batch OS
(2) Time Sharing OS
(3) Distributed OS
(4) Network OS
(5) Real Time OS
(6) Mobile OS
(7) Multiprocessing OS

* Batch OS :-

⇒ Batch operating system does not directly interact with the user.

⇒ This OS processes the routine jobs without any interactive user present.

⇒ Jobs with similar needs were batched together and were run through the computer as a group.

=> Thus, the programmer would leave their program with operator, who sorts the program in batches.

=> After each job finished, OS automatically read the next job.

=> Magnetic tapes were primary in use.

* Advantages:-

=> The idle time is very less

=> multiple user can share the same system

=> Increased Performance

* Disadvantages:-

=> Large Turnaround Time

=> More difficult to debug

=> Costly System

* Time sharing OS:-

=> Also known as multitasking OS

=> multiple jobs are executed simultaneously by switching CPU back and forth.

=> Each task is given some time to execute so that all the tasks work conviently.

=> The time that each task gets to execute is called Quantum.

=> After the ~~inver~~ interval is over, OS switches to next task.

* Advantages:-

=> Idle time is reduced.

=> Each tasks get equal chances to execute

=> Easy to use

* Disadvantages:-

=> Reliability issue.

* Real Time OS :-

=> This OS is used when there is strict time requirements on the operation of processor or flow of data

=> Two types:-> (1) Hard Real System
(2) Soft Real System

* Networking OS:-

=> This OS runs on server and provide the capability to manage users, groups, data, other functions.

=> This OS allows shared access of files, printers, etc.

=> Here, all the users are well aware of underlying configurations and other users also.

=> They are also called Hightly coupled Systems.

* Advantages:-

=> Server access possible remotely

=> Easily Scalable

=> Centralized server control

* Disadvantages:-

=> Costly Servers.

=> Maintanance & updates are required timely.

=> User depends on central Server

* Distributed OS:-

=> It is the most recent advancement in OS.

=> Here, various autonomous interconnected computers communicate with each other using a shared communication link / network.

=> Individual System have their own storage & CPU.

=> Also called | loosely coupled System. |

* Advantages:-

=> Reliable data sharing
=> Failure of one computer will not affect the others.
=> Easily scalable

* Disadvantages:-

=> Expensive to set up

=> highly complex

=> Failure of main network will stop the system.

Ans-

10

Contiguous Allocation	Linked list Allocation
=> Here, each file occupies contiguous set of blocks on the disk.	Each file is a linked list of disk blocks which are not contiguous.
=> Allocation is contiguous	Allocation need not be contiguous.
=> Directory entry is responsible for maintenance	Directory entry contains pointer to starting and ending block.
=> It is easy to implement	Some what complex to implement
=> Faster memory access.	Slower than the contiguous allocation
=> Wastage of memory spaces is there	=> Better utilization of memory is achieved
=> Suffers from internal & external fragmentation	=> Does not suffer from external fragmentation

Page:
Date:

ANS- * Definition of the following :-

(2)

=> Inode => It is a special disk block which is created with the creation of the file system.

=> In Unix based OS, each file is indexed by inode.

=> Buffer :=> Buffer is an area in the main memory used to store the data or to hold the data temporarily.

=> It contains data that is stored for short amount of time

=> Buffer cache :- Buffer cache is main memory used as a cache to reduce the number of physical read/writes from mass storage devices.

=> ialloc => It is similar to balloc, where it loops over the Inode structures on the disk, one block at a time looking for one that is marked free.

=> Namei => It follows a path name until a terminal point is found

=> ifree => It is used for creating and deleting files

Ans-(5)

* I/O Devices *

=> I/O Devices are basically the pieces of hardware used by humans to communicate with the computers.

=> They are required to take an application I/O Request and send it to physical device and vice versa.

=> There can be 2 categories of I/O Devices:-
(1) Character Devices
(2) Block Devices.

=> Character Device is one with which the driver communicates by sending and receiving single characters.

=> e.g. serial ports.

=> Block Devices are the one with which the driver communicates by sending entire blocks of data.

=> e.g. HDDs, USB, etc.

* DMA *

=> DMA stands for Direct Memory Access.

=> It means that CPU grants I/O module authority to read or write to memory without involvement.

=> The DMA module itself controls exchange of data between main memory and the I/O Device

=> It needs special hardware called DMA controller.

=> DMAC manages the data transfer and arbritates access to the system bus.

=> In DMA, CPU is only involved in the beginning and the end of transfer and interrupted only after certain / entire block is transferred.

* Device Controller *

=> The I/O unit consist of mechanical and electronic component.

=> The electronic component of I/O device is called Device Controller.

=> It is a card that usually has a connector on it.

=) The controllers are used to convert the serial bit stream into a block of bytes and perform any error correction if necessary.

(1) : 19DCS098

Ans - (4)

SSTF => stands for shortest seek Time First.

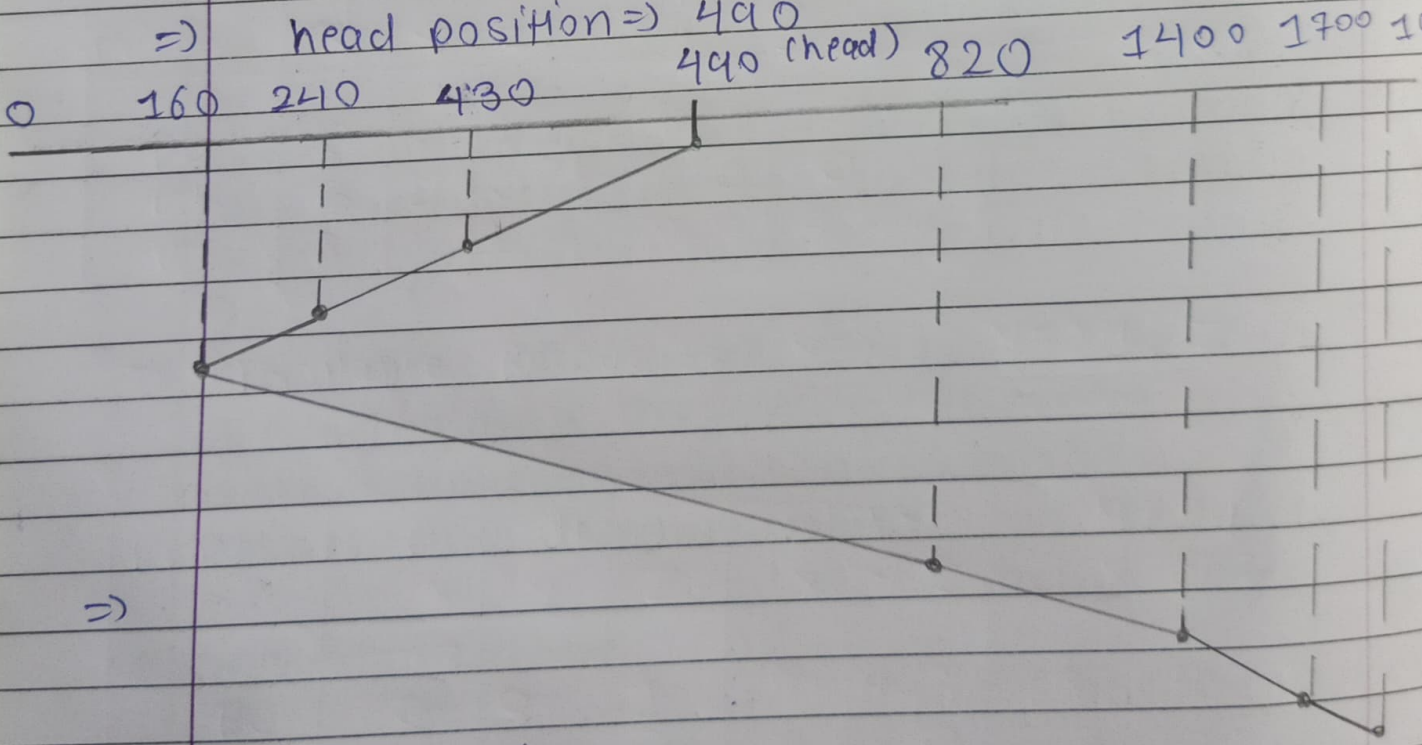
=> The Basic idea is the tracks that are closer to current disk head position should be serviced first.

=> The goal is to minimize Seek Time.

=> order Request :-

820, 1700, 430, 1400, 240, 160, 1900.

=> head position => 490



=>

=> Total Seek Time

= (490 - 430) + (430 - 240) + (240 - 160) + (820 - 160) + (1400 - 820) + (1700 - 1400) + (1900 - 1700)

= 60 + 190 + 80 + 660 + 580 + 300 + 200

= 330 + 1740 = 2070

=) Here, the head is at 440, shortest seek time is for 430; so, arm moves to 430.

=) So, at each position it goes to the position which has shortest seek time from the order Request.

Ans-(8)

=> Banker's Algorithm, is a resource allocation algorithm and also it is a deadlock avoidance algorithm.

=> The algorithm tests for safety by simulating the allocation for predetermined maximum possible amounts of all the resources.

=> Example given :->

Process	Allocation X	Y	Z	max X	Y	Z
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

=> Available Resources => X => 3 ; Y => 1, Z => 2

Process	Allocation X	Y	Z	max X	Y	Z	Need X	Y	Z
P0	0	0	1	8	4	3	8	4	2
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

=>

P0 => Need > available

=> 8 4 2 > 3 1 2

=> cannot execute P0

=> P1 => Need < Available

=> 3 0 0 < 3 1 2

=> P1 is executed

=> New available = (3,1,2) + (3,2,0)

= 6 3 2

=> Safe Sequence: P1 →

=> For P2 => Need < Available

=> 1 2 2 < 6 3 2

=> P2 is executed.

=> New available => (6,3,2) + (2,1,1)

= 8 4 3

=> Safe Sequence => P1 → P2 →

Again for P0 => Need < Available

=> P0 is executed

=> Safe Sequence => P1 → P2 → P0

=> < P1, P2, P0 >

Ans-(7) Most Optimal Page Replacement Algo:-

=> This is an algorithm that works when a page needs to be swapped in, the OS swaps out the page whose next use will occur farthest in the future

=> Page Sequence => 2, 3, 4, 1, 1, 2, 4, 3, 1, 1, 5, 6, 7, 6, 4, 3, 2, 1

=> As Number of Page Frame not given; we are considering the size as 3

Page Trace	PF 0	PF 1	PF 2	Hit/Miss
2	2			miss
3	2	3		miss
4	2	3	4	miss
1	2	1	4	miss
1	2	1	4	Hit
2	2	1	4	Hit
4	2	1	4	Hit
3	3	1	4	miss
1	3	1	4	Hit
1	3	1	4	Hit
5	3	5	4	miss
6	3	6	4	miss
7	7	6	4	miss
6	7	6	4	Hit
4	7	6	4	Hit
3	7	6	3	miss
2	7	6	2	miss
1	7	6	1	miss

(2) 19DCS093

* Least Recently Used Algo. :-

=> Here, the page will be replaced which is least recently used.

=> Again, considering Frames = 3

PageTrace	PF0	PF1	PF2	Hit / Miss
2	2			miss
3	2	3		miss
4	2	3	4	miss
1	3	4	1	miss
1	3	4	1	Hit
2	4	1	2	miss
4	1	2	4	Hit
3	2	4	3	miss
1	4	3	1	miss
1	4	3	1	Hit
5	3	1	5	miss
6	1	5	6	miss
7	5	6	7	miss
6	5	7	6	Hit
4	7	6	4	miss
3	6	4	3	miss
2	4	3	2	miss.
1	3	2	1	miss

Total References = 18

Number of Hit = 4

Number of Misses = 14

Hit Ratio = 0.222

Ans-(a) Producer-consumer problem is a classical synchronization problem.

=> A semaphore is an integer variable that can be accessed only through two standard operations: wait() and signal().

=> wait() reduces the value by 1

=> signal() increases the value by 1

=> Here, buffer is of fixed size 8.

=> It contains 4 items.

=> Producer can produce an item and place it in buffer.

=> A consumer can pick item from buffer and can consume them.

=> We need to make sure that when producer ~~consumer~~ is placing an item, then, ~~but~~ consumer should not pick any item.

=> This is solved by using semaphore.

=> We need 2 counting semaphores

=> Full & Empty.

=> Full keeps track of number of items in the buffer

=> Empty keeps track of number of unoccupied slots.

* Initialization of semaphores *

=> mutex = 1
Full = 4 // 4 items contains
Empty = 4 // 8-4 = 4

* For Producer:-

```
do {
    // produce an item

wait (empty);
wait (mutex);

// place in buffer

signal (mutex);
signal (full);

} while (true)
```

=> When producer produces, empty is reduced by 1.

=> Value of mutex is also reduced.

=> Producer placed item; value of full increases by 1.

Now; Full = 5
empty = 3

* Solution for consumer:

```
do {
    wait( full);
    wait ( mutex);

    // remove item from buffer

    signal (mutex);
    signal (empty);

    // consumes item
} while (true)
```

=> As consumer consumes item, value of full reduced by 1

=> Value of mutex also reduced.

=> Producer cannot access the buffer at this moment.

=> Value of empty increased by 1.

=> Value of mutex is also increased as producer can access the buffer now.

* Dining Philosopher Problem *

=> The problem states that K philosophers seated around a circular table with one fork / chopstick between each pair of philosophers.

=> A philosopher can eat if he picks two forks adjacent to him.

=> Semaphores is used to solve the problem

=> Total forks = 3.

=> Structure of fork =>

```
semaphore fork[3];
```

=> The structure of random philosopher:-

```
do {
  wait ( fork [i] );
  wait ( fork [(i+1) %3] );

    // eating

  signal (fork [i]);
  Signal ( fork[(i+1)%3] );

    // Thinking
} while (true);
```

=> Here, first wait operation is ~~for~~ done on fork[i] and fork[(i+1)%03]

=> So, philosopher i has picked forks on his sides.

=> Then eating function is performed

=> After that, signal operation is performed on fork[i] and fork[(i+1)%03]

=> This means, philosopher i has eaten done and put down forks.

=> Again, he goes back to thinking.