B.TECH.

ECS-501

FIFTH SEMESTER EXAMINATION 2010-11

OPERATING SYSTEM

Time: 3 Hours *Total Marks: 100*

Note: *Attempt all the questions.*

1. Attempt any four parts of the following: **(5 × 4 = 20)**

(a) What is an Operating System? Discuss the main Services of the operating system.

Ans. An operating system is software consisting of programs and data that runs on computers and manages computer hardware and resources. It provides common services for efficient execution of various application software. It acts as intermediary between computer user and computer hardware. Operating system is the one program running at all times on the computer usually called kernel. Operating system can be classified as multiuser, multiprocessing, multitasting, multithreading and real time system.

Main services provided by an operating system are:—

1. **Program execution:** The operating system loads the contents of a file into memory and begins its execution.
2. **Input/Output Operations:** A running program may require Input/Output, which may involve a file or Input/Output device. The user needs to specify the device and the operation to perform on it, while the system converts that request into device or controller specific commands.
3. **File system manipulation:** Programs need to perform many operation on files and directories. These may be read, write, delete, search for given file and list file information. Operating system provide file systems. Some programs include perinissions management to allow or deny access to files or directories based on file ownership.
4. **Communication:** One process may require exchanging information with other process that may be on same computer or on different computer system or network. Communication may be implemented via shared memory or through message parsing, in which packets of information are moved between processes by operating system.
5. **Error detection:** The operating system needs to be constantly aware of possible errors. Errors may occur in CPU and memory hardware in Input/Output devices and in user programms. For each type of error, operating system should take appropriate action to ensure correct and consistent computing.

(b) Discuss the differences between a time sharing system and real time system.

Ans. Real time operating system is an operating system intended to serve real time application requests. The key feature is the level of its consistency, concerning amount of time it takes to accept and complete an application task. The design goal of such system is not high throughput but rather guarantee of a soft or hard performance category. A Real time operating system is called soft real time operating system if it can meet a deadline. Even if certain delay happens, then such delays are tolerable. A hard Real time operating system is a Real time operating system that need to meet a deadline deterministically. In a real time Operating system, all presses are assigned a process priority. If a process is ready to run and it has a higher priority than other processes, the process is given control of CPU until

the process gets to a point where process is required to wait for Input/Output or completion of some event. If multiple processes with equal priority are ready, CPU is shared between them by using timesharing.

In time sharing system, each process is given a time slice and process can run on CPU for only that time slice. It allow many users to share computer simultaneously. As system switches rapidly from one user to the next, each user is given the impression that the entire computer system is dedicated to his/her use. These system required several jobs to be kept simultaneously in memory. So job scheduling is required. Memory management, CPU scheduling has to be done.

(c) Discuss the objectives of the Multiprocessor system.

Ans. In multiprocessor system, there are more than one processor that works simultaneously. It may have one master processor and other processors may act as slave. If one processor fails, master can assign the task to other slave processor. But if master processor fail, entire system will fail. All these processors may share hard disk, memory and other devices. There are tightly coupled system. They may share common data structure and common system clock. UNIX is most widely used multiprocessing system. Its objectives are:

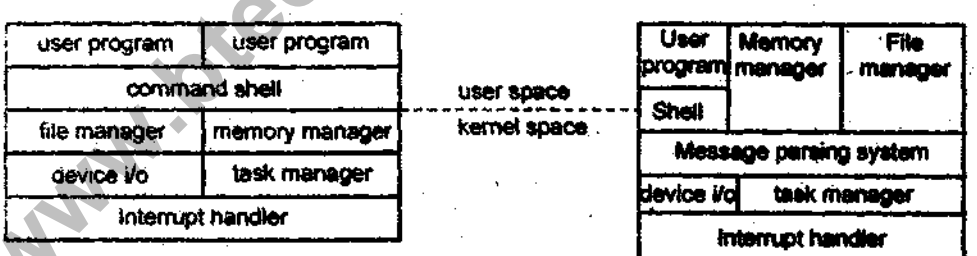
(1) **Reduced Cost:** Multiple processors share same resources. Thus it reduce cost.

(2) **Increased reliability:** If a processor gets failed, then master processor redistribute the workload of failed processor among other processor or to a processor which has less workload. So the failure of one processor does not affect other processors though it will slow down the machine. This is called graceful degradation.

(3) **Increased throughput:** An increase in number of processors complete the work in less time. Doubling the number of processors does not have the time to complete a job. It is due to overhead in communication between processors and contention for shared resources.

(d) What do you mean by Kernel? Explain monolithic kernel and microkernel.

Ans. Kernel is center or core of a computer operating system that provides basic services for all other parts of operating system. Kernel provide many services related to I/O such as scheduling, buffering caching, spooling, device reservation, error handling etc. These are of two types-monolithic and microkernal.



Monolithic Kernel **Microkernel**

In monolithic, all the components of operating system exist in a common address space and everything executes in Kernel mode. Monolithic kernel behaves like a big program. UNIX, LINUX are monolithic kernel. In microkernel, only a small portion of operating system functions in Kernel mode. It include tasks scheduling, interrupt handling. Low level device drivers and a message parsing subsystem to provide inter process communication. Rest of operating system components like memory manager and file system manager run as separate tasks in user space and communicate with each other by using Kernel's message parsing facilities. Ex. Mach, MINIX

Monolithic	Microkernal
1. **Maintainability:** Complicated interaction	Marginally better
2. **Stability:** Kernel error leads to crash	Isolation of errors slower as messaging overhead are there
3. **Performance faster:**	Slower as messaging overhead are there.
4. **Security:** Everything in Kernel mode. So secure	Many functions in uses mode so lesser secure.

(e) Describe the steps involved in Booting.

Ans. Firstly the computer is switched on. Computer then loads data from ROM and checks whether all the major components like processor, hard disk etc. are functioning properly. Computer then loads BIOS (basic input/output system) from ROM. Computer loads operating system from secondary storage (hard desk) into RAM. Operating system has been loaded, so all the functionalism are carried by operating system. The steps are:

1. The boot process starts by executing code in first sector of disk MBR.
2. The MBR looks over the partition table to find ACTIVE PARTITION.
3. Control is passed to that partititon's boot record (PBR) to continue booting.
4. The PBR locates the system specific boot file (such as wing's 10-sys or WINXP's ntoskrnl)
5. Then these boot files continue the process of loading and initializing the rest of the Operating System.

(f) Explain the following:

(i) Multitasking (ii) Multithreading.

Ans. Multitasking: A computer with multitasking capability can execute more than one process simultaneously. Multitasking is a combination of both multi programming and time sharing. A computer has capability to execute number of task and each of the task is given a time slot to execute. The switching of CPU among multiple process happens so fastly that each of the user feels that whole CPU is dedicated to their process only. The operating system is able to keep track of where you are in those tasks and go from one to other without losing information.

Microsoft Windows 2000, Linux are examples of operating system that can do multitasking. Ex: when we open *web browser* and then open *word* at same time, OS is doing multitasking.

Multithreading: It is the ability of operating system to execute different parts of the program called threads simultaneously. The program are designed so that different threads do not interface with each other. Multithreading is used to create scalability. It decreases execution time of the program. Individual programs are all isolated from each other in terms of their memory and data, but individual threads are not as they all share the same memory and data variables.

Q. 2. Attempt any two parts of the following: **(10×2=20)**

(a) Write an algorithm to explain the producer/consumer using semaphores.

Ans. The concurrent process executing in OS can be of two types.

1>Independent 2>Cooperating Process

Cooperating – A Process is cooperating if it can affect or be affected by other process executing in the system and these processes can share the data.

Producer – Consumer Problem:

- A producer process produces information that is consumed by consumer process.
- To allow producer & consumer process to run concurrently, we must have available a buffer of items that can be filled by the producer & emptied by the consumer.
- A Producer & consumer must be synchronized so that consumer does not try to consume an item that has not yet been produced.

The Producer and Consumer process shares the following:

Producer Process

```
# define BUFFER_SIZE 10
typedef struct
{
}item;
item bufer[BUFFER_SIZE];
int in=0;
into out=0;
while(1)
{
/* produce an item in pitem*/
while(((in+1)%BUFFER_SIZE)==out)
/*do nothing*/
buffer[in]=pitem;
in=(in+1)%BUFFER_SIZE;
}
```

CONSUMER PROCESS:

```
while(1)
{
while(in==out)
/*do nothing*/;
citem=buffer[out];
out=(out+1)%BUFFER_SIZE;
/*consumer item in citem*/
}
```

(b) (i) Compare and contrast the use of monitors and semaphores operations.

Ans. A semaphore is a protected variable whose value can be accessed and altered only by the operations *P* and *V* and initialization operation. Semaphore are code which help in synchronization. It allows multiple process to share a common sources uninterrupted. Semaphore may allow any number of processes to read from the resource, but only one process can write to that resource at a time.

P (wait operation) is defined as follows:

P(*S*) If $S > 0$

Then $S = S - 1$

Else (wait on *S*)

V (signal operation) on semaphore *S* is defined as follows:

V(*S*) If (one or more process are waiting on *S*)

Then (Let one the process proceed)

Else $S = S + 1$

Operation *P* and *S* are atomic operation. If several processes attempt *P*(*S*) simultaneously, only one process will be allowed to proceed and other processes will be kept waiting. Implementation of *P* and *V* guarantees that process will not suffer indefinite post ponement.

Monitors

1. A monitor is a collection of procedures, variables and data structures grouped together.
2. Processes can call monitor procedures but cannot access the internal data structures.
3. Only one process at a time may be active in a monitor. A process is active if it is in ready queue or with CPU where program counter is in monitor method.
4. A monitor is a language construct whereas semaphores are usually an OS construct.
5. The compiler usually enforces mutual exclusion in monitor.
6. Condition variables allow for blocking or non blocking.
7. CV Waitly blocks a process. The process is said to be waiting for condition variable CV.

   CV Signal unblocks a process waiting for condition variable. Only one process should be active in monitor. This can be done in several ways.

- On some systems old process (executing the signal) leaves the monitor and the new one enters.
- On some systems, the signal must be last statement executed inside the monitor.
- On some system, old process will block until the monitor is available again.
- On some system, the new process will remain blocked until the monitor is available again.
- If a condition variable is signaled with nobody waiting, the signal is lost.

(ii) What is Critical Section? Discuss.

Ans. Critical section is segment of code in which processes may be changing common variables, updating a table, writing a file, and so on. When a process execute in critical section, no other process is allowed to execute in critical section. A solution to critical section problem must satisfy the following three requirements.

1. **Mutual exclusion:** If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.
2. **Progress:** If no process is executing in its critical section and some processes wish to entry their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.
3. **Bounded waiting:** There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical section after a process has made a request to enter its critical section and before that request is granted.

(c) Discuss one classical problem related to the process synchronisation.

Ans. One of the classical problem related to the process synchronisation is

Dining philosophere problem

Consider five philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounded by five chairs, each belonging to one philosopher. In the center of the table is a bowl of rice, and table is laid with five single chopsticks. When a philosopher thinks, she does not interact with her colleagues. From time to time, a philosopher gets hungry and tries to pick up two chopsticks that are closest to her. A philosopher may pick up only one chospticks at a time. She cannot pick up a chopstick that is already in the hand of a neighbour. When a hungry philosopher has both her chopsticks at the same time, she eats without releasing her chopsticks. When she is finished eating, she puts down both of her chopsticks and starts thinking again.

Dining philosopher's solution

```
# define NS/ * Number of philosophers * /
# define RIGHT (i) (((i) + 1) % N)
# define LEFT (i) (((i) = = N)? 0 : (i) + 1)
typedef enum {THINKING, HUNGRY, EATING}
phil_state;
Phil_state state[N];
semaphore mutex = 1;
semaphore S[N]; /* one per philospher, all 0*/
Void test (int i){
If (State (i) = = HUNGRY &&
State [LEFT (i)]! = EATING & &(
State [RIGHT (i)]! = EATING)
{State [i] = EATING; V(S[L]);}
}
Void get-forks (in i)
{  P(Mutex);
   state[i] = HUNGRY.
   test (i);
   V (mutex);
   P(s[i]);
}
Void put-forks (int i)
{
   P(Mutex);
   State [i] = THINKING;
   test (LEFT(i));
   test (RIGHT(i));
```

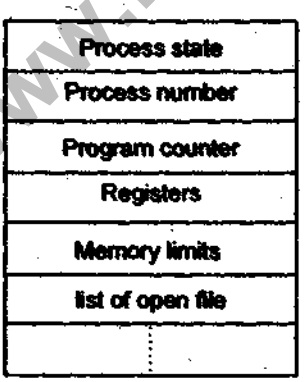
```
V(mutex);
}
Void philosopher(int process)
{ while (i)
{ think ( ))
get_forks (process);
eat ( );
put-forks (process);
}
}
```

Q. 3. Attempt any two parts of the following:

(10 × 2 = 20)

(a) (i) Explain the need for Process Control Block (PCB).

Ans. Each process is represented in operating system by process control block (PCB). The PCB is a store that allows operating system to locate key information about a process. Thus PCB is the data structure that defines a process to the operating system. Since PCB contains critical information for the process, it must be kept in an area of memory protected from normal user access. In some operating system, the PCB is placed at beginning of the Kernel stack of the process since that is a convenient protected location. PCB is called task controlling block. A PCB contain many pieces of information associated with a specific process, including there:



Process control block (PCB)

- **Process state:** The state may be new, ready, running, waiting, halted and so on.
- **Program counter:** The counter indicates the address of the next instruction to be executed for the process.
- **CPU register:** The registers vary in number and type, depending on the computer architecture. They include accumulations, index registers, stack pointers, and general purpose registers etc.
- **CPU scheduling information:** This information includes a process priority, pointers to scheduling queues and any other scheduling parameters.

Memory management information: The information may include such information as the value of base and limit registers, the tables, or the segment tables, depending on memory system used by the operating system.

Accounting Information: This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers and so on.

Input/Output status Information: The information includes the list of I/O devices allocated to the process, a list of open files and so on.

(ii) Discuss the performance criteria for CPU scheduling.

Ans. There are many scheduling algorithms and various criteria to judge their performance. Different algorithms may favour different types of process. Some criteria are:

- **CPU utilization:** CPU must be kept as busy as possible. CPU utilization is more important in real time system and multi-programmed system.
- **Throughput:** The number of processes executed in a specified time period or per unit time is called throguhput. For long processes, their rate may be one process per hour: for short process, it may be ten processes per second.
- **Waiting time:** It is sum of period spent waiting in the ready queue. CPU scheduling algorithm

does not affect the amount of time during which a process execute or does I/O. It affects only the amount of time that a process spends waiting in the ready queue.

- **Turnaround time:** The interval from the time of submission of a process to the time of completion is called turnaround time
- **Response time:** The amount of time between a request is submitted and first response is produced is called response time.

A CPU scheduling algorithm should maximize the following

1) CPU utilization 2) Throughput

A CPU scheduling algorithm should try to minimum the following

1) Turnaround time 2) Waiting time

3) Response time.

(b) (i) Describe the necessary condition for deadlock to occur.

Ans. A deadlock is a situation in which two computer program sharing the same resources are effectively preventing each other from accessing the resource, resulting in both program ceasing to function. In a deadlock, processes never finish executing, and system resources are tied up, preventing other jobs from starting. Deadlock situation can arise if and only 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 that is only one process at a time can use the resources. If another process requests that resource, the requesting process must be delayed until 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 processes exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2, P_{n-1} is waiting for a resource held by P_n and P_n is waiting for a resource held by P_0.

(ii) In the respect of Banker's Algorithm discus whether system is safe or unsafe. If a system is safe, show how it is possible for all users to complete:

	Current loan	Maximum need
user (1)	2	6
user (2)	4	7
user (3)	5	6
user (4)	0	2
Available	1	

Ans. Bankers safety algorithm as follows:

STEP 1: Initialize

Work = Available

for i = 1, 2, ... n

FINISH [i] = false

STEP 2: Find i such that both

a. Finish [i] = false

b. Nect [i] <= Work

If no such i, goto STEP 4

STEP 3:

Work = Work + Allocation [i]

Finish [i] = true

go to step 2

STEP 4:

If finish [i] = true for all i, system is in safe state.

Work = 1, finish [i] = false for i = 1, 2, 3, 4

Need [i] for user 1, 2 is greater than available. Need [3] = available, so user 3 will execute to completion and finish [3] = true work will now be 1 + 5 = 6. Now we have 6 resources. Now either of three remaining user can proceed. Let us consider

	current loan	Need
user 1	2	4
user 2	4	3
user 3	5	1
user 4	0	2

user 4 to execute to completion. So Finish [4] will be set to true Work = 6 + 0 = 6. Now say user 1 execute to completion. So Finish [i] is set to true. Work = 6 + 2 = 8. Now user 2 is executed to completion. Finish [2] = true, work = 8 + 4 = 12. As finish [i] is true for all i = 1, 2, 3, 4. Finally we exist from loop. Safety sequence = {user 3, user 4, user 1, user 2}.

(c) Explain the following scheduling algorithm:

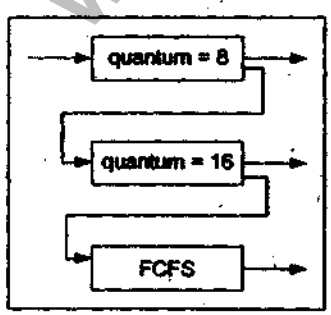
(i) Multilevel feedback queues scheduling

Ans. Multilevel feedback queues scheduling

In Multilevel queue scheduling, processes are permanently assigned to a queue when they enter the system. Such type of scheduling is inflexible. But in multilevel feedback queue scheduling algorithm, processes are allowed to move between queues. The idea is to separate processes according to the characteristics of their CPU burst. If a process uses too much CPU time, it will be moved to a lower priority queue. This scheme leaves Input/Output bound and interactive processes in higher priority queues. If a process waits too long in a lower priority queue, it may be moved to a higher priority queue. This form of aging prevents starvation.

Multilevel feedback queue scheduler is defined by following parameters:

1. Number of queues
2. Scheduling algorithms for each queue
3. Method used to determine when to upgrade a process
4. Method used to determine when to demote a process
5. Method used to determine which queue a process will enter when that process needs service.

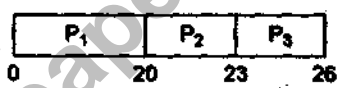


(ii) First in First out (FCFS) Scheduling

It is non preemptive scheduling algorithm. A FIFO queue is a list of available processes awaiting execution by the processor. New processes arrive and are placed at the end of queue. The process at the start of the queue is assigned the processor when it next becomes available, FIFO algorithm assigns priority to processes in the order in which they request the processor. The process that request CPU first is allocated the CPU first.

Let P_1, P_2, P_3 are three processes with CPU burst time 20, 3, 3 respectively.

Gantt chart is



Turnaround time for $P_1 = 20$

Turnaround time for $P_2 = 20 + 3 = 23$

Turnaround time for $P_3 = 20 + 3 + 3 = 26$

$$\text{Average turnaround time} = \frac{20+23+26}{3}$$

$$= \frac{69}{3} = 23$$

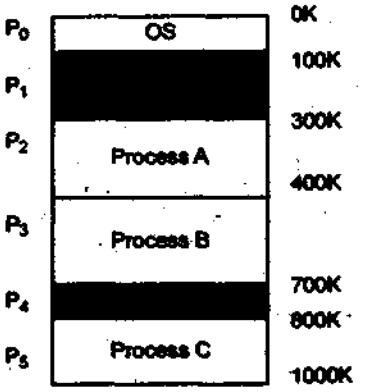
Its disadvantage is if one process acquire CPU 80 if we have first process with long CPU time then rest of the processes have to wait even if they have shorter CPU time. FCFS algorithm is troublesome for time sharing systems, where it is important that each user get a share of CPU at regular intervals.

Q. 4. Attempt any two parts of the following: (10×2=20)

(a) What do you understand by fragmentation? What are the different techniques to remove fragmentation in case of multiprogramming with fixed partitions and variable partitions? Discuss.

Ans. In Computer storage, **fragmentation** is a phenomenon in which storage space is used inefficiently, reducing storage capacity and in most cases performance. The term is also used to denote the wasted space itself.

(a) Fixed Partitioned memory Management System: In this scheme the main memory is divided into various sections called "partitions". These partitions could be of different size, but once decided at the time of system generation, they could not be changed.



Partition ID	Partition		
	Starting Address	Size	Status
0	0	100K	ALLC
1	100K	200K	FREE
2	300K	100K	ALLC
3	400K	300K	ALLC
4	700K	100K	FREE
5	800K	200K	ALLC

Fixed Partition and Partititon Description Table.

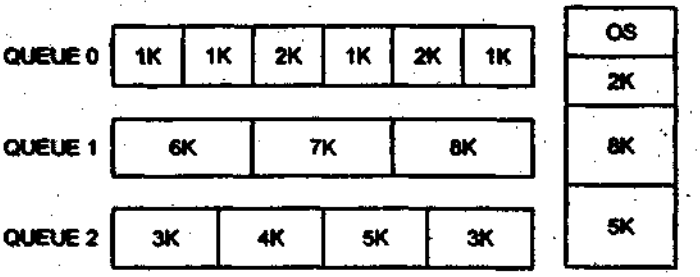
The OS creates a Partition Description Table (PDT). Initially all the entries are marked as 'FREE'. However as and when a process is loaded into one of the partitions, the status for that partition is changed to "ALLOCATED". Consider the above fig. The free partitions are only 1 and 4. Thus if a new process has to be loaded, we have to choose from these two partitions. The strategies of partition allocation are first fit, best fit and worst fit. For instance, if the size of the program to be loaded is 50K, the first fit would give Partition ID = 1 because the first free partition to accommodate this program is Partition ID = 1 and size of the partition with Partitition ID = 1 is 200K which is > 50K. The best fit strategy for the same task would yield Partition ID = 4, because the size of the partition is 100K, which is the smallest partition capable of holding this program. The processes waiting to be loaded in the memory are held in a queue by the OS. There are two methods of maintaining this queue, multiple queue and single queue.

First fit: Allocate the first hole (free partition) that is big enough.

Best fit: Allocate the smallest hole that is best enough. Search must be made in the entire list.

Worst fit: Allocate the largest hole. We must search the entire list unless it is stored by size.

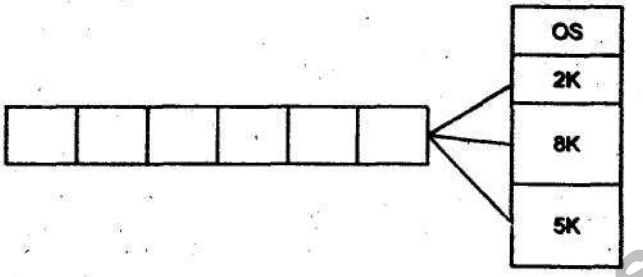
(1) Multiple Queue: There is one separate queue for each partition. Where a process wants to occupy memory, it is added to a proper queue depending upon size of the process. For instance, queue 0 will hold processes with size 0-2K, queue 2 will be processes with size between 3K-5K and queue 1 will take care of processes with size between 2K and 8K etc. as shown in fig.



Multiple queue

Advantage of this scheme is that a very small process is not loaded in a very large partition; thus avoid memory wastage.

(2) Single Queue



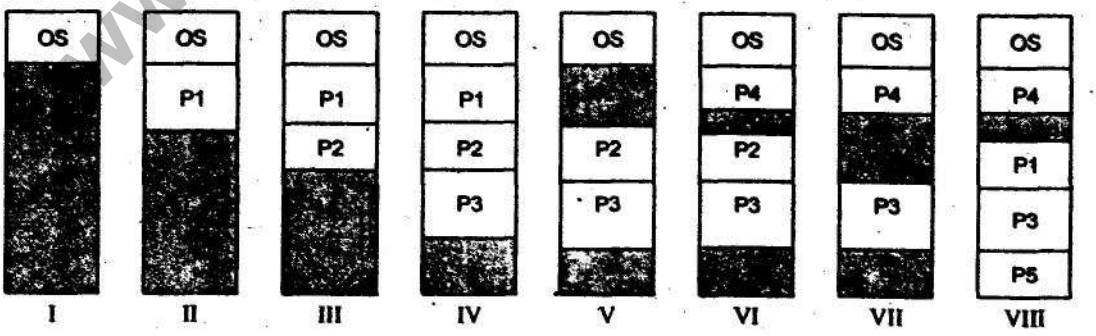
In this scheme, whenever a partition becomes free, the job closest to the front of the queue that fits in it could be loaded into the empty partition and run. Since it is undesirable to waste a large partitioin on a small job, a different strategy is to search the whole input queue whenever a partition becomes free and picks the largest job that fits.

Swapping: With batch system, organizing memory into fixed partition is simple and effective. With time sharing, the situation is different. There are normally more users than there is memory to hold all their processes, so it is necessary to keep excess processes on the disk. To run these processes they must be brought into the main memory. Moving processes from the main memory to disk and back is called swapping.

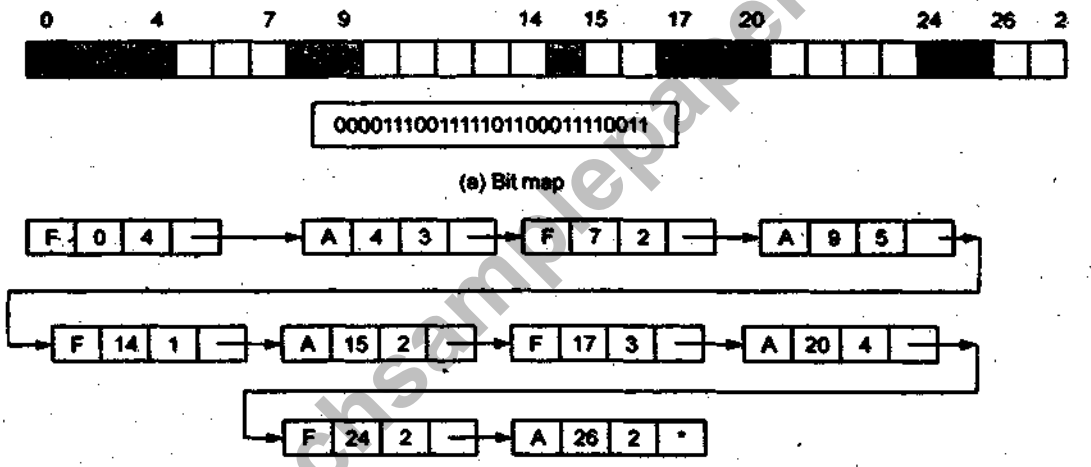
Fragmentation: Internal fragmentation: if a partition of size 100k is allocated to a process of size 60k then the 40k space of that partition is wasted and cannot be allocated to any other process. This is called 'internal fragmentation'. **External fragmentation:** It may happen that two free partitions of size 20k and 40k are available and a process of 50 k has to be accommodated. In this case both the available partitions cannot be allocated to that processes, because it will violate our assumption. For this scheme, viz. that only contiguous memory should be allocated to that process. As a result there is wastage of memory space. This is called 'external fragmentation'.

(b) Variable Partitioned Memory Management System

Ans. Variable partitions come into existence to overcome the problem of fragmentation in fixed partition scheme. When variable partition are used, the number and size of the processes in the memory vary dynamically. The following figure shows how variable partitions work.



- The OS is loaded in the memory. All the rest of the memory is free.
- A program P1 is loaded in the memory and it starts executing (After which it becomes a process).
- A program P2 is loaded in the memory and it starts executing.
- A program P3 is loaded in the memory and it starts executing. V) The process P is blocked. After a while a new program P4 wants to occupy the Memory. Let us assume that P4 is smaller than P1 but bigger than the free area available at the bottom.
- P4 is now loaded in the memory in the same space where P1 was loaded and starts executing.
- P2 terminates only P4 & P3 continues. The free area at the top and the one released by P2 is now combined.
- P1 is now swapped in as the OS has completed its I/O. The free space in the middle is sufficient to hold P1 now. Another process P5 also loaded. At this stage only a little free space left.



(b) Linked list

Compaction: External fragmentation problem in the variable partition can be solved by using the technique called compaction. Two holes at separate locations are not possible to Coalesce. If a process which requires memory more than each hole individually, but less than both hole put together, we can use compaction technique to combine these holes to accommodate this process.

In this method, the operating system has to move P3 (size=400) and P4(size=800). Hence the total movement is 1200.

(b) Define virtual memory concepts and also discuss page replacement algorithms in brief.

Ans. Virtual memory is a technique that allows the execution of processes that may not be completely in memory and process may be larger than main memory. Virtual memory abstract main memory into an extremely larger uniform array of storage. It is separation of user logical memory from physical memory. This separation allows an extremely large virtual memory to be provided for programmer when only a smaller physical memory is available. Virtual memory can be implemented by demand paging.

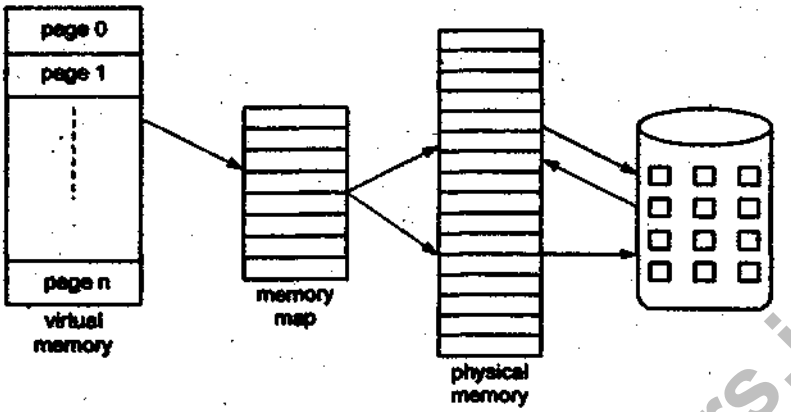
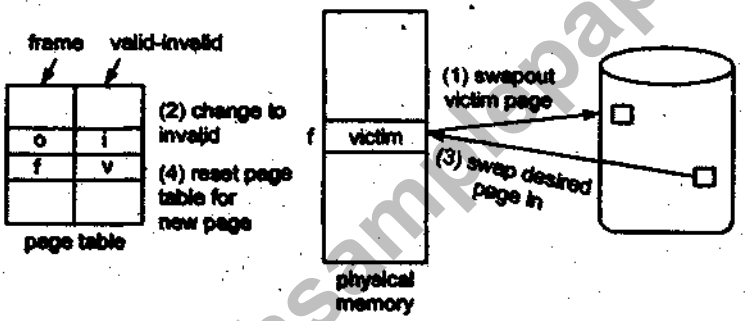


Diagram showing virtual memory that is larger than physical memory

Page replacement algorithm



(1) Find location of desired page on desk

(2) Find a free frame

— If there is a free frame, use it.

— If there is no free frame, use a page replacement algorithm to select a victim frame

(3) Read desired page into newly free frame. Update the page and frame tables.

(4) Restart the process

There are number of page replacement algorithm

1. **FIFO page replacement**

   It associates with each page the time when the page was brought into memory. When the page need to be replaced, the oldest page is chosen.

2. **Optimal page replacement (OPR)**

   Whenever a page need to be replaced, a page is searched which will not be used for longest period of time for replacement.

3. **Least Recently used algorithm (LRU)**

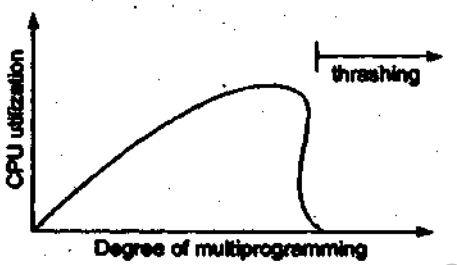
   It associates with each page the time of the page last used. When the page must be replaced, LRU chooses that page which has not been used for longest period of time.

(c) Write short notes on:

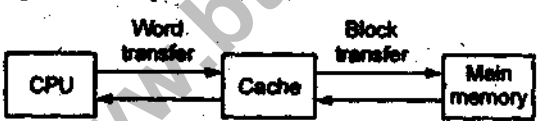
(i) Thrashing

(ii) Cache memory organisation.

Ans. Thrashing: It is high level paging activity. A process is thrashing if it is spending more time in paging than executing. Thrashing is caused by under allocation of minimum number of pages required by processs, forcing it to continuously do page fault. The system can detect thrashing by evaluating level of CPU utilisation as compared to level of multiprogramming. It can be eliminated by reducing level of multiprogramming.



Cache memory organization: It is memory that a computer microprocessor can access more quickly than it can access any other memory. As the microprocessor processes data, it looks first in cache memory and if finds the data there, it uses the data from cache. Cache memory is placed between processor and main memory. It is located either on processor chip or on a separate module.



Cache operation

1. Processor request the contents of data from memory.
2. The cache is checked for requested data.
3. If found, the requested word is delivered to processor.
4. If not found, a block of main memory is first read into the cache, then requested word is delivered to the processor. A method or technique is needed to determine which main memory block occupies which position in cache. Three techniques are used: direct, associative and set associative.

Q. 5. Attempt any two parts of the following: **(10×2=20)**

(a) What are the different file organizations? Discuss access mechanism.

Ans. File organization is the methodology which is applied to structure computer files. Files contain computer records which can be documents or information which is stored in a certain way for later retrieval. File organization refers primarily to the logical arrangement of data (which can itself be organized in a system of records with correlation between the fields/ columns) in a file system. It should not be confused with the physical storage of the file in some types of storage media. There are certain basic types of computer file, which can include files stored as blocks of data and streams of data, where the information streams out of the file while it is being read until the end of the file is encountered.

We will look at two components of file organization here:

- The way the internal file structure is arrange and
- The external file as it is presented to the O/S or program that calls it. Here we will also examine the concept of file extensions.

We will examine various ways that files can be stored and organized. Files are presented to the application as a stream of bytes and then an EOF (end of file) condition. A program that uses a file needs to known the structure of the file and needs to interpret its contents.

Types of File Organization: Organizing a file depends on what kind of file it happens to be: a file in the simplest form can be a text file, (in other words a file which is composed of ascii (American Standard Code for Information Interchange text.) Files can also be created as binary or executable types (containing elements other than plain text.) Also, files are keyed with attributes which help determine their use by the host operating system.

Techniques of File Organization

The thee techniques of file organization are:

- Heap (unordered)
- Sorted
- Sequential (SAM)
- Line Sequential (LSAM)
- Indexed Sequential (ISAM)
- Hashed or Direct

In addition to the three techniques, there are four methods of organizing files. They are **sequential, line-sequential, indexed-sequential, inverted list** and **direct or hashed access** organization.

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, continuous 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.

Line-Sequential Organization: Line-sequential files are like sequential files, except that the records can contain only characters as data. Line-sequential files are maintained by the native byte stream files of the operating system. In the COBOL environment, line-sequential files that are created with WRITE statements with the ADVANCING phrase can be directed to a printer as well as to a disk.

Indexed-Sequential Organization: Line-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.

Inverted List: In file organization, this is a file that is indexed on many of the attributes of the data itself. The inverted list method has a single index for each key type. The records are not necessarily stored in a sequence. They are placed in the are data storage area, but indexes are updated for the record keys and location.

Content-based queries in text retrieval systems use **inverted indexes** as their preferred mechanism. Data items in these systems are usually stored compressed which would normally slow the retrieval process, but the compression algorithm will be chosen to support this technique.

Direct or Hashed Access:

With **direct or hashed access** a portion of disk space is reserved and a "hashing" algorithm computes the record address. So there is additional space required for this kind of file in the store. Records are placed randomly through out the file. Records are accessed by addresses that specify their disc-location. Also, this type of file organization requires a disk storage rather than tape. It has an excellent search retrieval performance, but care must be taken to maintain the indexes. If the indexes become corrupt, what is left may as well go to the bit-bucket, so it is as well to have regular backups of this kind of file just as it is for all stored valuable data.

(b) Explain the following

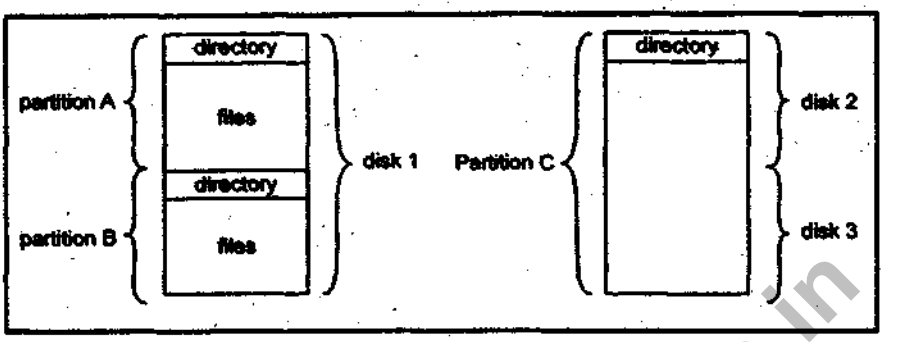
1. Directory System

2. File Protection

Ans. 1. Directory System

Directory is a collection of nodes containing information about all files. Both the directory structure and the files reside on disk. Backups of these two structures are kept on tapes.

A Typical File-system Organization



Information in a Device Directory

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID (who pays)
- Protection information

Operations Performed on Directory

Search for a file

Create a file

Delete a file

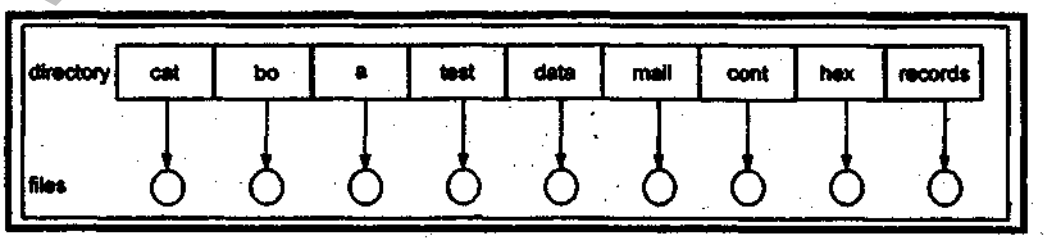
List a directory

Rename a file

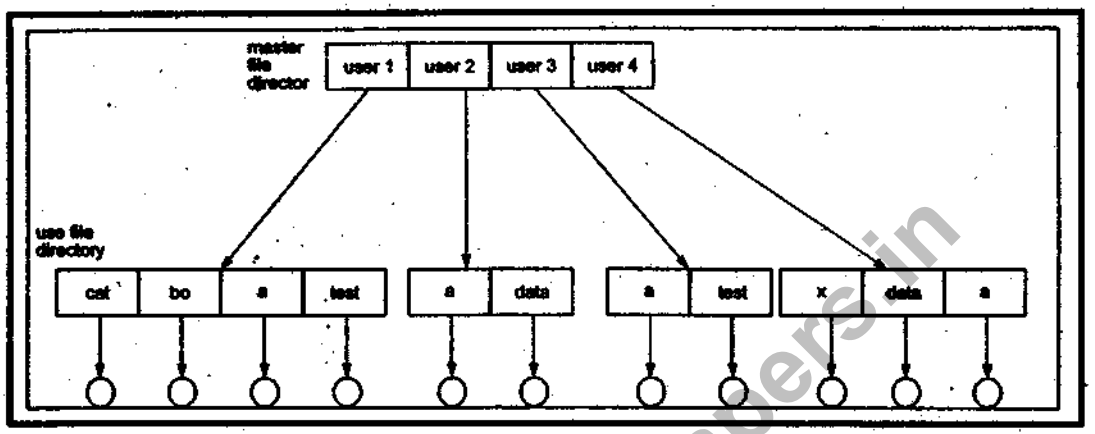
Traverse the file system

Directory Structure

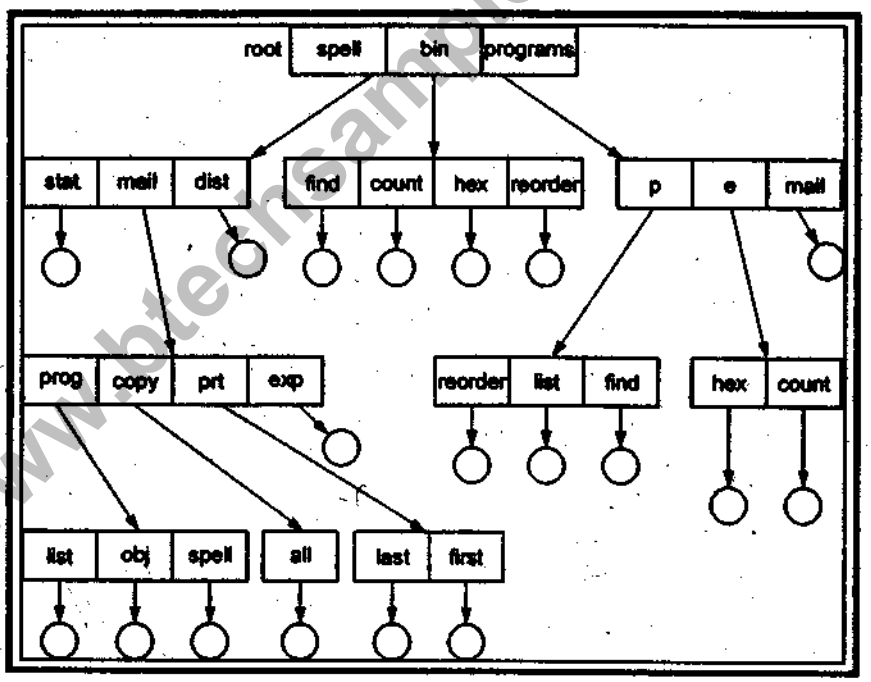
• **Single-Level Directory:** A single directory for all users. There arises Naming problem and Grouping problem



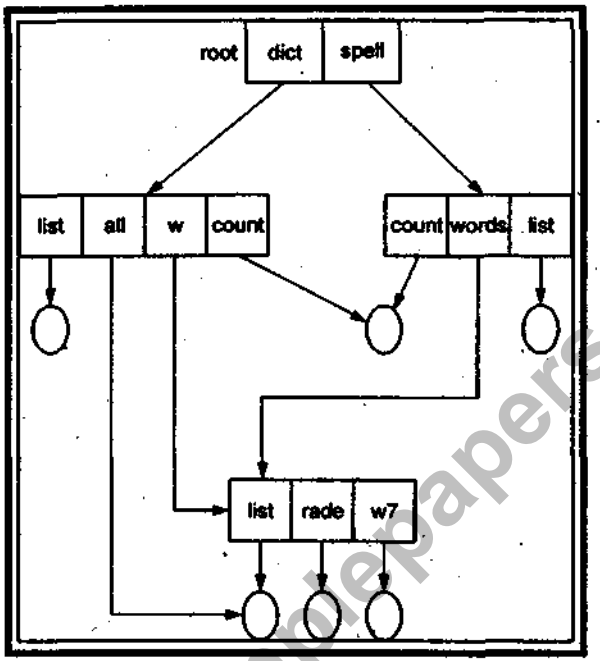
Two-level Directory: Here there is a Separate directory for each user. It can have the same file name for different user. It has efficient searching and no grouping capability.



Tree-structured directories: It can have efficient searching and grouping Capability



Acyclic-Graph Directories :It can have shared subdirectories and files.



2. File Protection: Protection mechanism provide controlled access by limiting the types of file access that can be made. File owner/creator should be able to control what can be done and by whom. Types of access are:

Read: Read from the file

Write: Write or rewrite the file

Execute: Load the file into memory and execute it

Append: Write new information at the end of the file

Delete: Delete the file and free its space for possible reuse

List: List the name and attribute of the file **Access Lists and Groups**

- Mode of access: read, write, execute
- Three classes of users

(a) owner access: The user who created the file is the owner.

(b) group access: A set of users who are sharing the file and need similar access.

(c) public access: All other user in the system constitutes the universe.

- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game)* or subdirectory, define an appropriate access.

(c) Discuss the Disk Scheduling Algorithms.

Ans. The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth. Access time has two major components

Seek time is the time for the disk are to move the heads to the cylinder containing the desired sector.

Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.

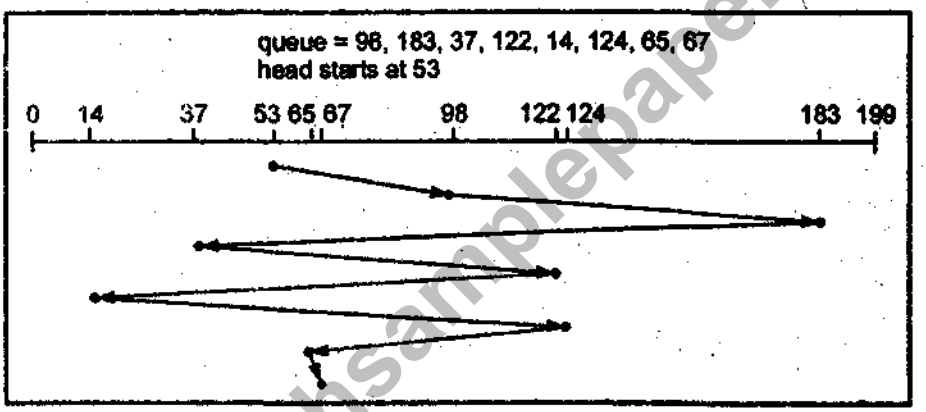
Several algorithms exist to schedule the servicing of disk I/O requests. We illustrate them with a request queue (0-199).

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

Head pointer 53

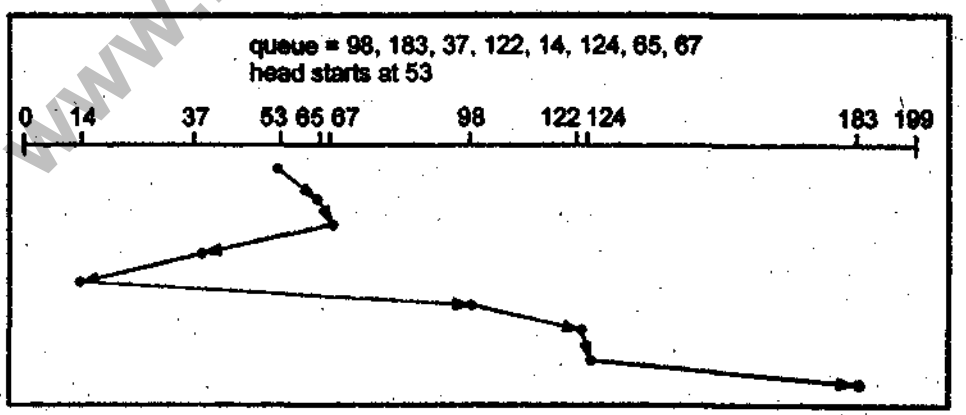
1. FCFS

The simplest form of disk scheduling is first come first serve (FCFS) algorithm. This algorithm is intrinsically fair but it generally does not provide fastest service.



2. SSTF

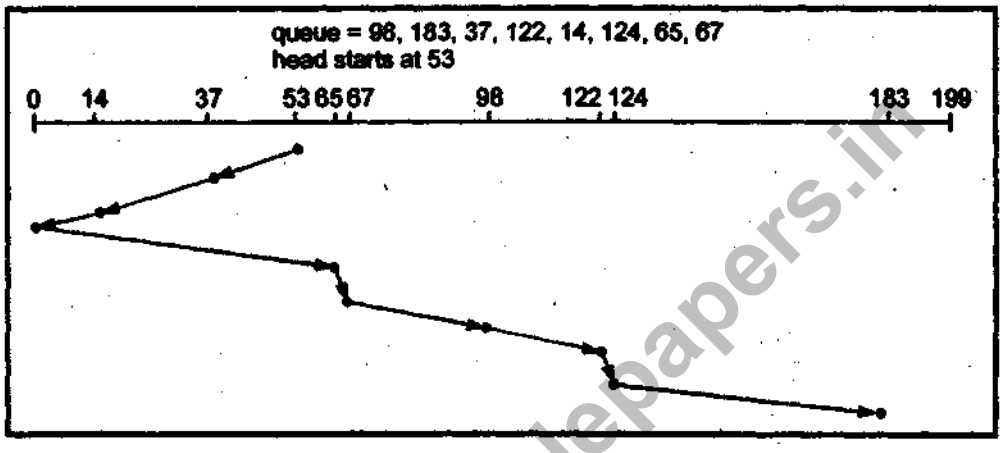
It selects the request with the minimum seek time from the current head position. SSTF scheduling is a form of SJF scheduling and it may cause starvation of some requests.



3. SCAN

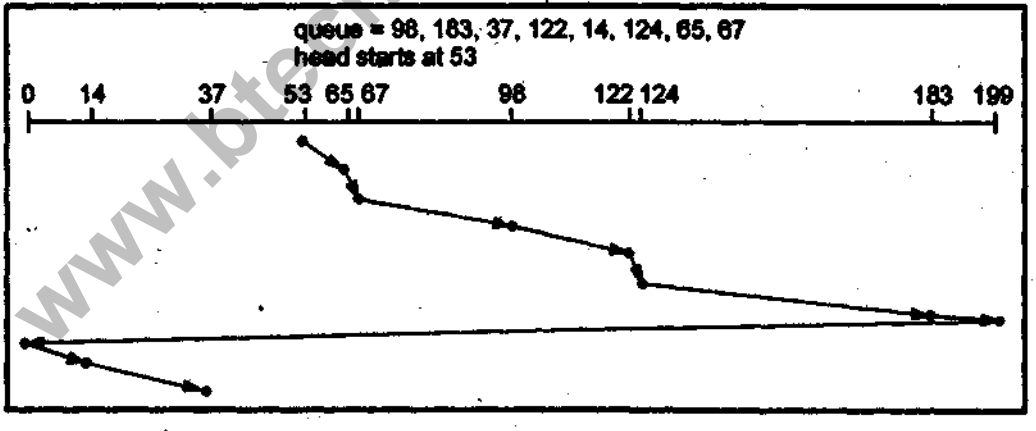
The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues. It is sometimes called the *elevator algorithm.*

Illustration shows total head movement of 208 cylinders.



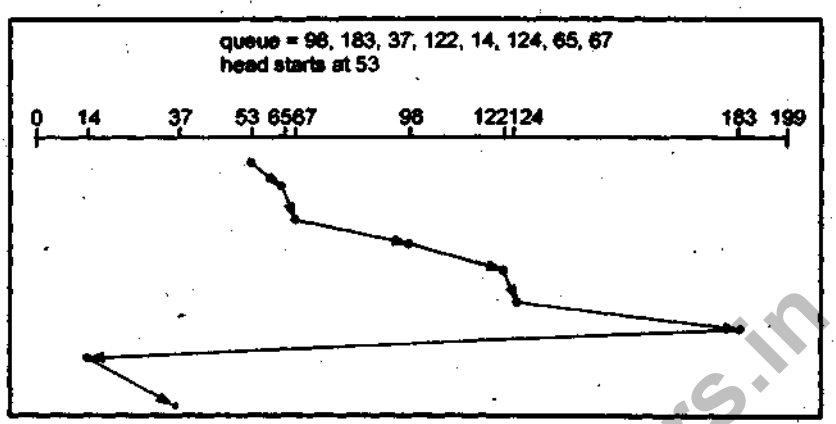
4. C-SCAN

It provides a more uniform wait time than SCAN. The head moves from one end of the disk to the other servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip. It treats the cylinders as a circular list that wraps around from the last cylinder to the first one.



5.C-LOOK

It is a version of C-SCAN. Here arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.



Selecting a Disk-scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different alorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.