**KLE Society’s BCA, RLSI Belagavi**

**Subject: Operating System – Question Bank**

**BCA II Semester (NEP)**

**UNIT 1: INTRODUCTION**

1.Define long term and short term scheduler?

**Ans)** • Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU

• Long-term scheduler (or job scheduler) – selects which processes should be brought into the ready queue

2. Define operating system?

**Ans)** Operating System can be defined as a system software which provides an interface between an user, application and computer hardware.

3.What is meant by a process and program.

**Ans)** A process can be thought of as a program in execution. A process is an active entity. Program is an passive entity.

1. Describe the operating system functions?

**Ans) Functions of Operating System**

* File Management. ...
* Device management. ...
* Process management. ...
* Memory management. ...
* Job Accounting.

1. List out any four information management system calls?

* **Ans)** File Management. ...
* Device Management. ...
* Information Maintenance. ...
* Communication.

1. List out any four process control system calls?

* **Ans)** File Management. ...
* Device Management. ...
* Information Maintenance. ...
* Communication.

7. Define thread.

**Ans)** A thread is a basic unit of CPU utilization it comprises a thread ID, a program counter, register set, stack.

8. List the different types operating system.

**Ans)** 1.Batch Operating System

2.Time-Sharing Operating Systems

3. Distributed Operating System

4.Network Operating System

5. Real-Time Operating System

9. What are the various objectives and functions of Operating systems?

* Security. ...
* Control Over System Performance. ...
* Job Accounting. ...
* Error Detecting Aids. ...
* Coordination Between Other Software and Users. ...
* Memory Management

10 .Define system programs. List the categories of system programs.

**Ans)** A system program is nothing but a special utility program that creates a user friendly environment where the user can perform his desired work efficiently.

Categories of system programs:

➢ File Management

➢ Status information

➢ File Modification

➢ Programming language support

➢ Program loading and execution

➢ Communications

11. What are the models of inter-process communication?

**Ans)** Two models of inter-process communication are:

➢ Shared memory

➢ Message passing

12. Explain the basic instruction cycle with appropriate diagram?

**Ans)** The instruction cycle consists of four phases: **fetching an instruction from memory, decoding the fetched instruction, reading the address from memory, and finally, instruction execution**

13. Draw the Process Control Block diagram

**Ans)**



14. List the different thread models.

**Ans)** 1. MANY TO ONE

2. ONE TO ONE

3. MANY TOO MANY

15. What are the different views of operating system?

**Ans)** There are 2 views for operating system:-

1. User View

2. System View

16. Define Turnaround time, waiting time.

**Ans)** Turnaround time- Turnaround time is the amount of time to execute a particular process (completion time – arrival time).

Waiting time- Waiting time is the amount of time a process has been waiting in the ready queue. (Turnaround time- Bust time)

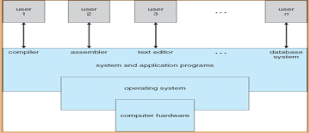
17. Define virtual machine. Give examples.

**Ans)** Virtual machine abstracts the hardware of our personal computer such as CPU, Disk Drive, memory, ect

Examples:Vmware, Java Virtual Machine

18. Explain abstract view of components of computer system.

**Ans)** Operating System can be defined as a system software which provides an interface between a user, application and computer hardware.



Abstract view of components of computer system

A computer system can be divided into four components:-

1. Hardware – The hardware provides the basic computing resources for the system (CPU, memory, I/O devices).

2. Operating system –The operating system controls and coordinates the use of the hardware among the various application programs for the various users.

3. Applications programs –The application programs define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).

4. Users: The users may be people, machines, and other computers.

19. Explain the different views of operating system.

**Ans)** There are 2 views for operating system:-

User View:-

• The user’s view of the computer varies according to the interface being used.

• Users want convenience, ease of use and good performance, they do not care about resource utilization.

• In case of shared computer such as mainframe or minicomputer the operating system is designed to maximize resource utilization to assure that all the available resources are used efficiently and no individual users takes more than her fair share.

• Users of dedicated systems such as workstations have dedicated resources but frequently use shared resources from servers.Therefore, their operating system is designed to compromise between individual usability and resource utilization.

• Some computers have little or no user interface, such as embedded computers in devices and automobiles. Their operating system is designed primarily to run without user intervention.

• System View:-

• From the computer’s point of view, operating system is viewed as a resource allocator.

• A computer system has many resources that may be required to solve problem. The operating system acts as the manager of these resources and resolves conflicting requests for efficient and fair resource use.

• Operating System is a control program, it controls execution of programs to prevent errors and improper use of the computer.

20. List and Explain tasks of operating system.

**ANS)** 1. Booting a Computer: Booting a computer is to load an operating system into the computer's main memory or random access memory (RAM).

2. User-interface: User interfaces. A user interface (UI) refers to the part of an operating system, program, or device that allows a user to enter and receive information.

3. Running Programs: To keep the processor busy at all times, the execution of such a program is halted and the operating system switches the processor to run another program.

4. Managing Files: The system that an operating system or program uses to organize and keep track of files.

5. Memory Management: Memory management is the process of controlling and coordinating computer memory, assigning portions called blocks to various running programs to optimize overall system performance.

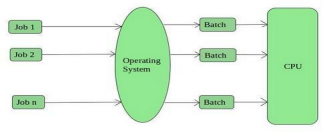
6. Scheduling Jobs: The system handles prioritized job queues that are awaiting CPU time and it should determine which job to be taken from which queue and the amount of time to be allocated for the job.

21. Explain types of operating system with diagram.

**Ans)** The different types of operating systems are:-

1. Batch Operating System

• This type of operating system do not interact with the computer directly. There is an operator which takes similar jobs having same requirement and group them into batches. It is the responsibility of operator to sort the jobs with similar needs.



Advantages

• Multiple users can share the batch systems.

• It is easy to manage large work repeatedly in batch systems.

Disadvantages

• Batch systems are hard to debug.

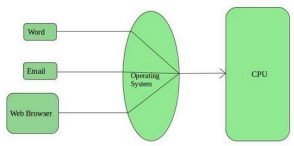
• The computer operators should be well known with batch systems.

• The other jobs will have to wait for an unknown time if any job fails. Examples: IBM’s OS360

2. Time-Sharing Operating Systems

➢ Each task is given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems.

➢ the task can be from single user or from different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.



Advantages

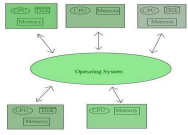
• Each task gets an equal opportunity

• CPU idle time can be reduced

Examples: UNIX, Multics

3. Distributed Operating System

• These are referred as loosely coupled systems or distributed systems. These systems processors differ in sizes and functions. The major benefit of working with these types of operating system is that it is always possible that one user can access the files or software which are not actually present on his system but on some other system connected within this network i.e., remote access is enabled within the devices connected in that network.



Advantages

• Failure of one will not affect the other network communication, as all systems are independent from each other.

• Since resources are being shared, computation is highly fast and durable. Disadvantages

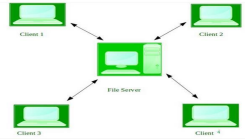
• Failure of the main network will stop the entire communication

Examples: LOCUS

4. Network Operating System

➢ These systems runs on a server and provides the capability to manage data, users, groups, security, applications, and other networking functions. These type of operating systems allows shared access of files, printers, security, applications, and other networking functions over a small private network.

➢ One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections etc. and that’s why these computers are popularly known as tightly coupled systems.



Advantages

• Highly stable centralized servers.

• Security concerns are handled through servers.

Disadvantages

• Servers are costly.

• User has to depend on central location for most operations.

Examples: Microsoft Windows Server 2003, Mac OS X

5. Real-Time Operating System

• These types of operating systems serves the real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called response time.

➢ Hard Real-Time Systems:

These operating systems are meant for the applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or air bags which are required to be readily available in case of any accident. Virtual memory is almost never found in these systems.

➢ Soft Real-Time Systems:

These operating systems are for applications where for time-constraint is less strict. Advantages

• Real time operating system in embedded system: Since size of programs are small, RTOS (real time operating system) can also be used in embedded systems like in transport and others.

Disadvantages

• Complex Algorithms: The algorithms are very complex and difficult for the designer to write.

Examples: robots, air traffic control systems.

1. Compare Time-Sharing Operating Systems and Distributed Operating System

**Ans)** A time-shared OS enables numerous users to share computers simultaneously. On the other hand, a distributed operating system is a concept in which distributed applications run on several systems that are linked together via communications. Multiple users share computer resources in Time-sharing OS.

1. Differentiate between Network Operating System and Distributed Operating System

**Ans)**

|  |  |  |
| --- | --- | --- |
| .NO | Network Operating System | Distributed Operating System |
| 1. | Network Operating System’s main objective is to provide the local services to remote client. | Distributed Operating System’s main objective is to manage the hardware resources. |
| 2. | In Network Operating System, Communication takes place on the basis of files. | In Distributed Operating System, Communication takes place on the basis of messages and shared memory. |
| 3. | Network Operating System is more scalable than Distributed Operating System. | Distributed Operating System is less scalable than Network Operating System. |
| 4. | In Network Operating System, fault tolerance is less. | While in Distributed Operating System, fault tolerance is high. |
| 5. | Rate of autonomy in Network Operating System is high. | While The rate of autonomy |

1. Explain in brief the concept of multiprogramming and time sharing. • Multiprogramming

**Ans)**

• It is needed for efficiency

– Single user cannot keep CPU and I/O devices busy at all times.

– Multiprogramming organizes jobs (code and data) so CPU always has one to execute.

A subset of total jobs in system is kept in memory. One job selected and run via job scheduling. When it has to wait (for I/O for example), operating system switches to another job.



Memory layout for a multiprogramming system

• Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing

– Response time should be than a second

– Each user has at least one program executing in memory is called process.

25. Explain the structure of operating system. **Ans)**• General-purpose Operating System is a very large program.

• Various ways to structure Operating System are:-

1. Simple Structure

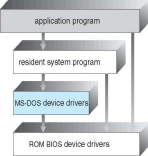
• These operating systems are started as small, simple, limited systems and then grow beyond their original scope...

• Example:MS-DOS

• It was written to provide the most functionality in the least space.

• In MS-DOS, the interfaces and levels of functionality are not well separated. • For instance, application programs are able to access the basic I/O routines to write directly to display and disk drives.

• Such freedom leaves MS-DOS vulnerable to errant programs, causing entire system crashes when user program fails.



MS-DOS layer structure

2. Layered Approach

• The operating system is divided into a number of layers (levels), each built on top of lower layers.

• The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface. • With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.

• A layer is an implementation of an abstract object.

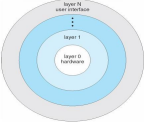
• The object is made up of data and operations that can manipulate the data. • The layer consists of a set of routines that can be invoked by higher layers. • Advantage:

1. Simplicity of construction and debugging.

Disadvantages:

1. Less efficient than other types.

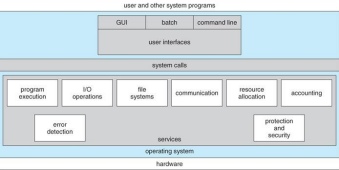
2. Appropriately defining the various layers.



Layered Operating system

26. Explain the operating system services.

**Ans)**



A view of operating system services

• Operating systems provide an environment for execution of programs and services to programs and users.

• One set of operating-system services provides functions that are helpful to the user: • User interface -

➢ Almost all operating systems have a user interface (UI).This interface can take several forms. One is a Trace command-line interface (CLI), which uses text commands and a method for entering them.

➢ Another is a batch interface, in which commands and directives to control those commands are entered into files, and those files are executed.

➢ In Graphical User Interface(GUI), the interface is a window system with a pointing device t direct I/O,choose from menus.

• Program execution - The system must be able to load a program into memory and to run that program, end execution, either normally or

abnormally (indicating error)

• I/O operations - A running program may require I/O, which may involve a file or an I/O device.The operating system must provide a means to do Input/Output(I/O).

• File-system manipulation - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

• Communications –

➢ Processes may exchange information, on the same computer or

between computers over a network.

➢ Communications may be implemented via shared memory or through message passing ,in which packets of information are moved between processes by the operating system.

• Error detection –

➢ Operating system needs to be constantly aware of possible errors. ➢ Errors may occur in the CPU and memory hardware, in I/O devices, in user program.

➢ For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.

➢ Debugging facilities can greatly enhance the user’s and programmer’s abilities to efficiently use the system.

• Another set of operating-system services provides functions for ensuring the efficient operation of the system itself are:-

• Resource allocation –

• When multiple users or multiple jobs running concurrently, resources must be allocated to each of them.

• Many types of resources - CPU cycles, main memory, file storage, I/O devices.

• Some may have special allocation code,whereas others may have much more general request and release code.

• Accounting - To keep track of which users use how much and what kinds of computer resources.

• Protection and security - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other.

• Protection involves ensuring that all access to system resources is controlled.

• Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts.

27. Explain system calls with diagram.

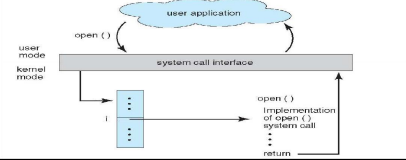
**Ans)**System calls provide the interface between a process and the operating system. An Example of how system calls are used.

➢ Consider a simple program to copy data from one file to another file . ➢ The first input of the program will be the name of both files.

➢ These names can be specified depending on the design of operation system. ➢ Once the file names are obtained, the program opens an input file and create an output file.

➢ Each of these operations requires system calls and may encounter possible error conditions.

The handling of user application invoking the open() system call



➢ The API(Application Programming Interface) specifies set of functions that are available to an application programmer including the parameters that are passed to each function and the return values the programmer can expect.

➢ Users write functions for API’s which calls the system call interface which in turn invokes the system call.

28.Explain the different categories of system calls.

**Ans)**The different categories of system calls are:

➢ Process control

– end, abort

– load, execute

– create process, terminate process

– get process attributes, set process attributes

– wait for time

– wait event, signal event

– allocate and free memory

➢ File management

– create file, delete file

– open, close file

– read, write, reposition

– get and set file attributes

➢ Device management

– request device, release device

– read, write, reposition

– get device attributes, set device attributes

– logically attach or detach devices

➢ Information maintenance These are calls that exist for transferring information between the user program and the operating system.

– get time or date, set time or date

– Number of current users

– Amount of free memory or disk space

➢ Communications :Deals with inter process communication

– create, delete communication connection

– send, receive messages

29. Define system programs? And explain categories of system programs

**Ans)** A system program is nothing but a special utility program that creates a user friendly environment where the user can perform his desired work efficiently. Examples: operating system, interpreter, compiler, editors are all system programs. They provide built in environment to the user, hence the user does not need to write any code for such functions.

Categories of system programs:

➢ File Management: These programs create, delete, copy, rename, print and generally manipulate files and directories.

➢ Status information: Some programs simply ask the system for the date, time, and amount of available memory or disk space, number of users.

➢ File Modification: several text editors may be available to create and modify the content of files stored on disk or other storage devices.

➢ Programming language support: compilers, assemblers debuggers and interpreters for common programming languages(such as C,C++,Java) are often provided to the user with the operating system.

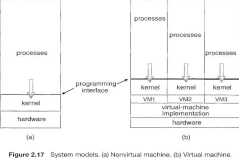
➢ Program loading and execution: once a program is assembled or compiled, it must be loaded into the memory to be executed.The system may provide loaders,linkge editors.

➢ Communications:These programs provide the mechanism for creating virtual connections among processors.They allow users to send messages to one another’s screens,to send e-mail,to login remotely.

30.Explain the concept of virtual machines with examples.

**Ans)**• In computing, a virtual machine (VM) is an emulation of a computer system. • The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory ,disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.

• In figure 2.17(b),each guest process is provided with a (virtual)copy of the underlying computer.



Benefits:

• The host system is protected from the virtual machines, just as the virtual machines are protected from each other.

• A virtual-machine system is a perfect vehicle for operating-systems research and development

Examples:

Vmware

Vmware Workstation runs as an application on a host operating system such as Windows or Linux and allows this host system to concurrently run several different guest operating systems as independent virtual machines.

Java Virtual Machine

The Java Virtual Machine (JVM) is the runtime engine of the Java Platform, which allows any program written in Java or other language compiled into Java bytecode to run on any computer that has a native JVM.

31.What is a process? Draw and explain process state diagram.

**Ans)**A process can be thought of as a program in execution. A process is an active entity.



When a process is created and terminated, from beginning to end during its life cycle a process can remain in many states.

• Start or New: First state is Start or new state. A process is said to be in new state when it is started or created.

• Ready:It is the state next to new state.The process is said to be in ready state when it is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. A process may come into ready state in three situation:-

❑ First is just after the Start state,

❑ Second situation may be while running and getting interrupted by the scheduler to assign CPU to some other process in this situation it comes to ready state so that CPU can execute the new process.

❑ Third situation from wait state when I/O is completed or wait for an even is completed.

• Running: Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions then process is said to be in running state.

• Waiting: When a process is in running state then process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.

• Terminated or Exit: Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.

32. Explain Process Control Block.

**Ans)**



• As the operating system supports multi-programming, it needs to keep track of all the processes. For this task, the process control block (PCB)- also called a Task Control Block is used to track the process’s execution status.

• It contains many pieces of information associated with a specific process including these:

➢ Process state – It stores the respective state of the

process(new,ready,running,waiting,terminated).

➢ Process number – Every process is assigned with a unique id known as process ID or PID which stores the process identifier.

➢ Program counter – It stores the counter which contains the address of the next instruction that is to be executed for the process.

➢ Register – These are the CPU registers which includes: accumulator, base, registers and general purpose registers.

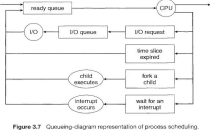
➢ Memory limits – This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.

➢ List of open files – This information includes the list of files opened for a process.

➢ Miscellaneous accounting and status data – This field includes information about the amount of CPU used, time constraints, jobs or process number, etc.

33.Explain process scheduling with diagram.

**Ans)**• The act of determining which process is in the ready state, and should be moved to the running state is known as Process Scheduling.



• All processes, upon entering into the system, are stored in the Job Queue. • Processes in the Ready state are placed in the Ready Queue.

• Processes waiting for a device to become available are placed in Device Queues. There are unique device queues available for each I/O device.

• A new process is initially put in the Ready queue. It waits in the ready queue until it is selected for execution(or dispatched). Once the process is assigned to the CPU and is executing, one of the following several events can occur:

➢ The process could issue an I/O request, and then be placed in the I/O queue. ➢ The process could create a new subprocess and wait for its termination.

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

• In the first two cases, the process eventually switches from the waiting state to the ready state, and is then put back in the ready queue. A process continues this cycle until it terminates.

34. Define scheduler and differentiate between short term and long term schedulers.

**Ans)** Schedulers are special system software which handle 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.

• Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU

– Sometimes the only scheduler in a system

– Short-term scheduler is invoked frequently (milliseconds) ⇒ (must be fast)

• Long-term scheduler (or job scheduler) – selects which processes should be brought into the ready queue

– Long-term scheduler is invoked infrequently (seconds, minutes) ⇒ (may be slow)

– The long-term scheduler controls the degree of multiprogramming • Processes can be described as either:

– I/O-bound process – spends more time doing I/O than computations, many short CPU bursts

– CPU-bound process – spends more time doing computations; few very long CPU bursts

• Long-term scheduler strives for good *process mix*

35. Define thread.

**Ans)** a thread is a basic unit of CPU utilization it comprises a thread Ida program counter, register set, stack.

36. What is the difference between single and multi-threaded processes explain with diagram.

**Ans)**



Single and Multithreaded Processes

Single Thread

O Has single thread of control

O It allows the process to perform only 1 task at a time.

Multi thread

O Has many threads

O Simultaneous execution of different task

37. Explain thread models with diagram.

**Ans)**

1. MANY TO ONE:

➢ many user-level threads mapped to single kernel thread

➢ Used on systems that do not support kernel threads

➢ Examples:

o Solaris Green Threads

o GNU Portable Threads



2.ONE TO ONE

➢ Each user-level thread maps to kernel thread.

➢ Examples

o Windows 95/98/NT/2000

o Linux



3.MANY TO MANY





➢ The many to many model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models.

➢ Users can create any number of the threads.

➢ Blocking the kernel system calls does not block the entire process.

➢ Processes can be split across multiple processors.

38.Explain inter-process communication.

**Ans)** • Cooperating processes need inter process communication (IPC)

• Two models of inter-process communication are:

➢ Shared memory

➢ Message passing



➢ Shared memory:

• An area of memory shared among the processes that wish to communicate.

• The communication is under the control of the user processes not the operating system.

• Major issue is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory.

➢ Message Passing

• Mechanism for processes to communicate and to synchronize their actions.

• Message system – processes communicate with each other without resorting to shared variables.

• Inter process communication facility provides two operations:

➢ send(*message*)

➢ receive(*message*)

• The *message* size is either fixed or variable.

39.Explain scheduling criteria.

**Ans)** • Many criteria have been suggested for comparing CPU scheduling algorithm. The criteria include the following

1. CPU utilization- keep the CPU as busy as possible.

2. Throughput- Number of processes that complete their execution per time unit.

3. Turnaround time- amount of time to execute a particular process (completion time – arrival time).

4. Waiting time- amount of time a process has been waiting in the ready queue.

5. Response time-amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment).

40. What is starvation? Explain with example

**Ans)**

41. Explain the different types of scheduling algorithms.

**Ans)** 1. First-Come, First-Served (FCFS) Scheduling

• First job that requests the CPU is assigned the CPU.

•It is non preemptive i.e a process continues till the burst cycle.

• Advantages

– Simple

– Fair (as long as no process hogs the CPU, every process will eventually run) •Disadvantages

– Waiting time depends on arrival order

– Short processes are stuck waiting for long processes to complete.

2. Shortest Job First (SJF) Scheduling

No preemption

Associate with each process the length of its next CPU burst

Use these lengths to schedule the process with the shortest time

Schedule process with the shortest burst time – FCFS if same

Advantages – Minimizes average wait time and average response time

Disadvantages – Not practical : difficult to predict burst time

Learning to predict future – May starve long jobs

3. Priority Scheduling

In priority scheduling, a priority number (integer) is associated with each

process.

The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)

Problem ≡ Starvation i.e low priority processes may never execute.

Solution ≡ Aging i.e as time progresses, increase the priority of the process.

4. Round Robin (RR) Scheduling

• Each process gets a small unit of CPU time (time quantum *q*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

• Timer interrupts every quantum to schedule next process

• Advantages

– Fair (Each process gets a fair chance to run on the CPU)

– Low average wait time, when burst times vary

– Faster response time

• Disadvantages

– Increased context switching

– High average wait time, when burst times have equal lengths

**UNIT-2 PROCESS COORDINATION (SYNCHRONIZATION AND DEADLOCKS)**

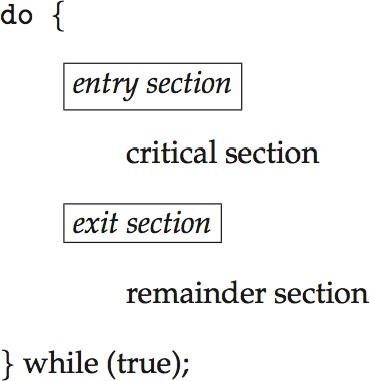
1. **What is semaphore?**

A semaphore is a synchronization tool that does not require busy waiting .It is an integer variable that,apart from initialization,is accessed only through two standard atomic operations:wait() and signal()

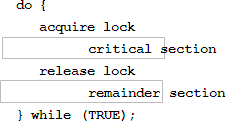
#### List three requirements that a solution to critical section problem must satisfy.

Mutual Exclusion Progress Bounded Waiting

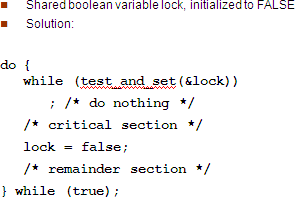
#### Write general structure of critical section problem



1. **Write solution to critical-section problem using Locks**



#### Write code for mutual-exclusion implementation with TestAndSet( )



1. **Explain two primitive semaphore operations.** Two standard atomic semaphore operations are a)**Wait( )**

The wait( ) operation was originally termed p “ to test” b)**Signal( )**

The signal( ) operation was originally called V “to increment”

1. **What are the requirements that a solution to the critical section problem must satisfy?**

## 1. Mutual Exclusion.

## 2. Progress.

## 3. Bounded Waiting

#### Explain two operations used in semaphore implementation with no Busy waiting

* 1. **block** – place the process invoking the operation on the appropriate waiting queue
  2. **wakeup** – remove one of processes in the waiting queue and place it in the ready queue

#### List 3 classical problems of synchronization

* Bounded-Buffer Problem
* Readers and Writers Problem
* Dining-Philosophers Problem

#### Write the structure of producer process

do {

...

/\* produce an item in next\_produced \*/

...

wait(empty); wait(mutex);

...

/\* add next produced to the buffer \*/

...

signal(mutex); signal(full);

} while (true);

#### Write the structure of Philosopher i of Dinning-Philosophers problem

do {

wait (chopstick[i] );

wait (chopStick[ (i + 1) % 5] );

// eat

signal (chopstick[i] );

signal (chopstick[ (i + 1) % 5] );

// think

} while (TRUE);

#### List four necessary conditions for a deadlock situation to arise.

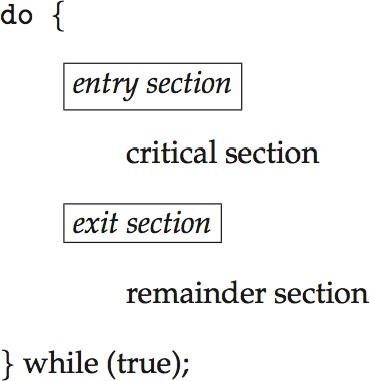
* Mutual Exclusion
* Hold and wait
* No pre-emption
* Circular wait

1. **Explain critical section problem.**

Consider a system consisting of n processes (P0, P1, ………Pn -1) each process has a segment of code which is known as critical section in which the process may be changing common variable, updating a table, writing a file and so on. The important feature of the system is that when the process is executing in its critical section no other process is to be allowed to execute in its critical section. The execution of critical sections by the

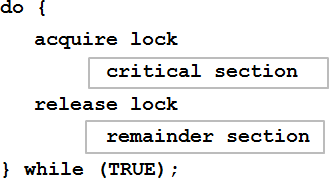
processes is a mutually exclusive. The critical section problem is to design a protocol that the process can use to cooperate each process must request permission to enter its critical section. The section of code implementing this request is the entry section. The critical section is followed on exit section. The remaining code is the remainder section.

#### Example:



1. **Explain solution to critical section problem using locks**

In general, we can state that any solution to the critical-section problem requires a simple tool-a lock. Race conditions are prevented by requiring that critical regions be protected by locks.That is,a process must acquire a lock before entering a critical section;it releases the lock when it exits the critical section.This is illustrated below



#### Explain three requirements that a solution to critical section problem must satisfy.

**A solution to the critical section problem must satisfy the following three conditions.**

1. **Mutual Exclusion :** If process Pi is executing in its critical section then no any other process can be executing in their critical section.
2. **Progress:** If no process is executing in its critical section and some process wish to enter

The

ir critical sections then only those process that are not executing in their remainder section can enter its critical section next.

1. **Bounded waiting:** There exists a bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request.

#### What is semaphore? Explain two primitive semaphore operations.

For the solution to the critical section problem one synchronization tool is used which is known as

semaphores. A semaphore ‘S’ is an integer variable which is accessed through two standard operations such as wait and signal. These operations were originally termed ‘P’ (for wait means to test) and ‘V’ (for single means to increment).

#### The classical definition of wait is

Wait (S)

{

While (S <= 0)

{

Test;

}

S--;

}

#### The classical definition of the signal is

Signal (S)

{ S++;

}

#### Explain producer consumer problem using semaphore Bounded Buffer Problem:

* 1. This problem was commonly used to illustrate the power of synchronization primitives.
  2. In this scheme we assumed that the pool consists of ‘N’ buffer and each capable of holding one item.
  3. The ‘mutex’ semaphore provides mutual exclusion for access to the buffer pool and is initialized to the value one.
  4. The empty and full semaphores count the number of empty and full buffer respectively.
  5. The semaphore empty is initialized to ‘N’ and the semaphore full is initialized to zero. This problem is known as **procedure and consumer problem.**
  6. The code of the producer is producing full buffer and the code of consumer is producing empty buffer.

#### The structure of producer process is as follows:

do {

produce an item in nextp

. . . . . . . . . . . .

Wait (empty);

Wait (mutex);

. . . . . . . . . . .

add nextp to buffer

. . . . . . . . . . . .

Signal (mutex);

Signal (full);

} While (true);

#### The structure of consumer process is as follows:

do {

Wait (full); Wait (mutex);

. . . . . . . . . . .

Remove an item from buffer to nextc

. . . . . . . . . . .

Signal (mutex); Signal (empty);

. . . . . . . . . . . .

Consume the item in nextc;

. . . . . . . .. . . .. .

} While (true);

#### Explain readers-writers problem of synchronization

* A data set is shared among a number of concurrent processes
  + Readers – only read the data set; they do *not* perform any updates
  + Writers – can both read and write

Problem is to allow multiple readers to read at the same time. Only one writer can access the shared data at the same time.

* Shared Data
  + Data set
  + Semaphore **rw\_mutex** initialized to 1
  + Semaphore **mutex** initialized to 1
  + Integer **read\_count** initialized to 0

#### The structure of a writer process

do{

wait(rw\_mutex);

...

/\* writing is performed \*/

...

signal(rw\_mutex);

} while (true);

#### The structure of a reader process do{

wait(mutex); read count++;

if (read\_count == 1) wait(rw\_mutex); signal(mutex);

...

/\* reading is performed \*/

...

wait(mutex); read\_count--;

if (read\_count == 0) signal(rw\_mutex); signal(mutex);

} while (true);

1. **Explain the structure of a reader writer process**

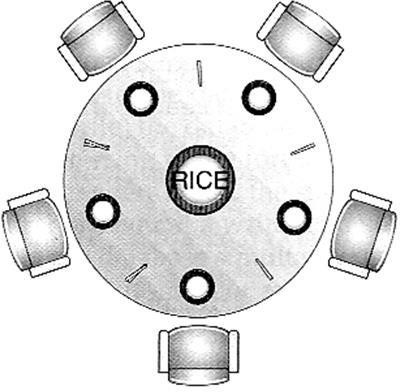
The readers-writers problem relates to an object such as a file that is shared between multiple processes. Some of these processes are readers i.e. they only want to read the data from the object and some of the processes are writers i.e. they want to write into the object.

The readers-writers problem is used to manage synchronization so that there are no problems with the object data. For example - If two readers access the object at the same time there is no problem. However if two writers or a reader and writer access the object at the same time, there may be problems.

To solve this situation, a writer should get exclusive access to an object i.e. when a writer is accessing the object, no reader or writer may access it. However, multiple readers can access the object at the same time.

#### Explain dinning philosopher problem with diagram.

Consider 5 philosophers to spend their lives in thinking &eating. A philosopher shares common circular table surrounded by 5 chairs each occupies by one philosopher. In the center of the table there is a bowl of rice and the table is laid with 6 chopsticks as shown in below figure.



When a philosopher thinks she does not interact with her colleagues. From time to time a Philosopher gets hungry and tries to pickup two chopsticks that are closest to her. A Philosopher may pickup one chopstick or two chopsticks at a time but she cannot pickup

A Chopstick that is already in hand of the neighbor. When a hungry philosopher has both her

Chopsticks at the same time, she eats without releasing her chopsticks. When she finished Eating, she puts down both of her chopsticks and starts thinking again. This problem is Considered as classic synchronization problem. According to this problem each chopstick is represented by a semaphore. A philosopher grabs the chopsticks by executing the wait Operation on that semaphore. She releases the chopsticks by executing the signal operation on the appropriate semaphore. The structure of dining philosopher is as follows:

#### The structure of Philosopher *i* :

do {

wait(chopstick[i]); wait(chopstick[(i+1)%5]);

// eat signal(chopstick[i]);

signal(chopstick[(i+1)%5]);

// think

} while (TRUE);

#### What is monitor? Explain with syntax and diagram.

Monitor is a high-level abstraction that provides a convenient and effective mechanism for process synchronization. It is a *Abstract data type*, in which internal variables are only accessible by code within the procedure.Only ***one*** process may be active within the monitor at a time.It not powerful enough to model some synchronization schemes

Syntax:

#### monitor monitor-name

**{**

#### // shared variable declarations

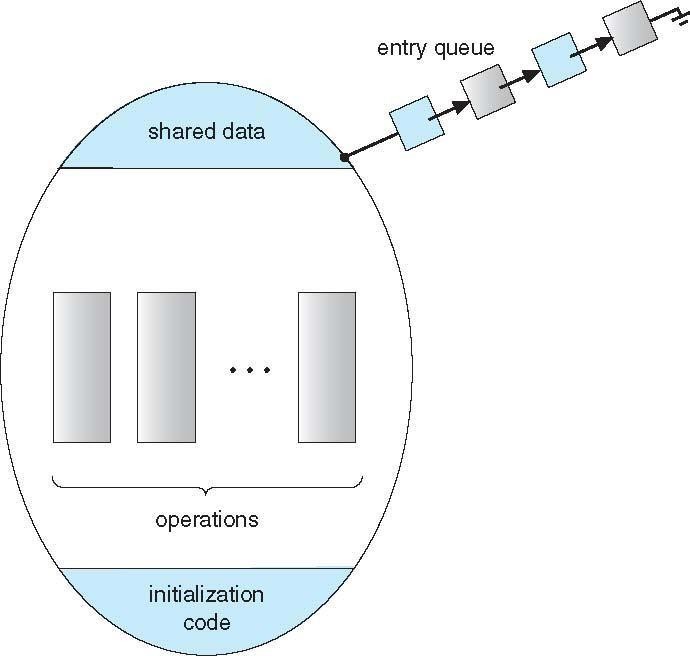
**...**

#### function P1(...) { ... }

**function Pn(...) { ... }**

#### initialization\_code (...) { ... }

**}**



#### Write solution to dining philosopher problem using monitor.

monitor DiningPhilosophers

{

enum {THINKING, HUNGRY, EATING} state[5];

condition self [5]; void pickup (int i) {

state[i] = HUNGRY; test(i);

if (state[i] != EATING) self[i].wait;

}

void putdown (int i) {

state[i] = THINKING;

// test left and right neighbors test((i + 4) % 5);

test((i + 1) % 5);

}

void test (int i) {

if((state[(i + 4) % 5] != EATING) && (state[i] == HUNGRY) && (state[(i + 1) % 5] != EATING) ) {

state[i] = EATING ; self[i].signal () ;

}

}

initialization\_code() {

for (int i = 0; i < 5; i++) state[i] = THINKING;

}

}

#### Describe necessary conditions for a deadlock situation to arise.

* **Mutual exclusion:** only one process at a time can use a resource
* **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes
* **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task
* **Circular wait:** there exists a set {*P*0, *P*1, …, *P*n} of waiting processes 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, …, *Pn*–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.

#### Write in detail about Deadlock Avoidance

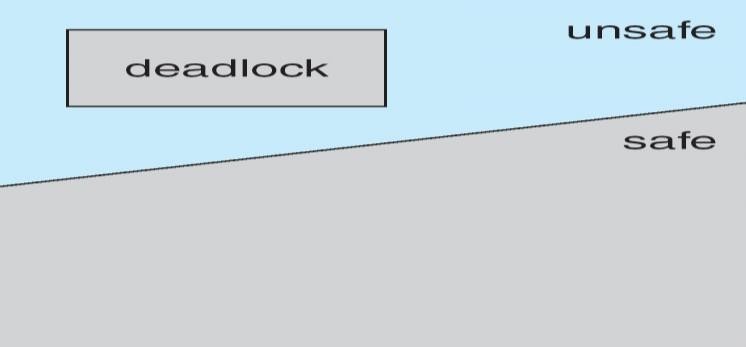
**Requires that the system has some additional *a priori* information available:**

* Simplest and most useful model requires that each process declare the ***maximum number*** of resources of each type that it may need
* The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition
* Resource-allocation *state* is defined by the number of available and allocated resources, and the maximum demands of the processes.

#### o Safe State

* When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state
* System is in **safe state** if there exists a sequence <*P1, P2, …, Pn*> of ALL the processes in the systems such that for each Pi, the resources that Pi can still request can be satisfied by currently available resources + resources held by all the *Pj*, with *j* < *I*
* That is:
  + If Pi resource needs are not immediately available, then *Pi* can wait until all *Pj*

have finished

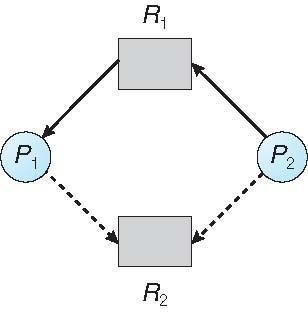
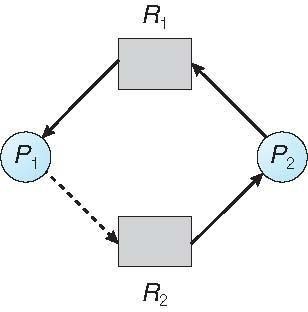
* + When *Pj* is finished, *Pi* can obtain needed resources, execute, return allocated resources, and terminate
  + When *Pi* terminates, *Pi* +1 can obtain its needed resources, and soon

#### BASIC FACTS

* If a system is in safe state  no deadlocks
* If a system is in unsafe state  possibility of deadlock
* Avoidance  ensure that a system will never enter an unsafe state.

#### Explain resource allocation graph algorithm with diagram.

* Single instance of a resource type uses a resource-allocation graph
* **Claim edge** *Pi*  *Rj* indicated that process *Pj* may request resource *Rj*; represented by a dashed line
* Claim edge converts to request edge when a process requests a resource
* Request edge converted to an assignment edge when the resource is allocated to the process
* When a resource is released by a process, assignment edge reconverts to a claim edge
* Resources must be claimed *a priori* in the system

**Fig 1**:Resource allocation graph for deadlock avoidance Fig2: An unsafe state in a resource allocation graph

* Suppose that process *Pi* requests a resource *Rj*
* The request can be granted only if converting the request edge to an assignment edge does not result in the formation of a cycle in the resource allocation graph

#### Explain banker’s algorithm.

* Multiple instances of a resource type use the banker’s algorithm.
* Each process must a priori claim maximum use
* When a process requests a resource it may have to wait
* When a process gets all its resources it must return them in a finite amount of time

#### Data Structures for the Banker’s Algorithm

**Let *n* = number of processes, and *m* = number of resources types**

* **Available***:* Vector of length *m*. If available [*j*] = *k*, there are *k* instances of resource type *Rj* available
* **Max***: n x m* matrix. If *Max* [*i,j*] = *k*, then process *Pi* may request at most *k* instances of resource type *Rj*
* **Allocation***: n* x *m* matrix. If Allocation[*i,j*] = *k* then *Pi* is currently allocated *k*

instances of *Rj*

* **Need***: n* x *m* matrix. If *Need*[*i,j*] = *k*, then *Pi* may need *k* more instances of *Rj* to complete its task

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

#### Explain the methods used to Prevent Deadlocks.

**Methods used to Prevent Deadlocks:**

* **Mutual Exclusion** – not required for sharable resources (e.g., read-only files); must hold for non-sharable resources
* **Hold and Wait** – must guarantee that whenever a process requests a resource, it does not hold any other resources

o Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none allocated to it.

#### No Preemption –

* If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released
* Preempted resources are added to the list of resources for which the process is waiting
* Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting
* **Circular Wait** – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

#### Explain resource-request algorithm

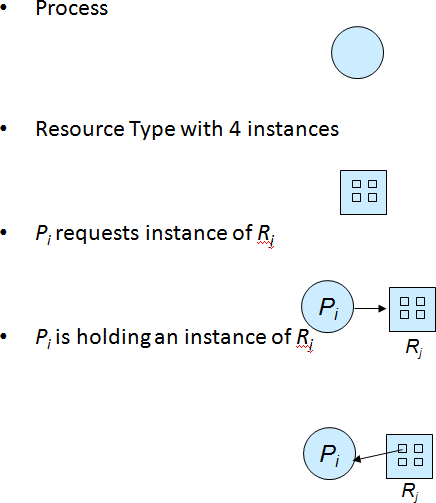
***Requesti* = request vector for process *Pi*. If *Requesti* [*j*] = *k* then process**

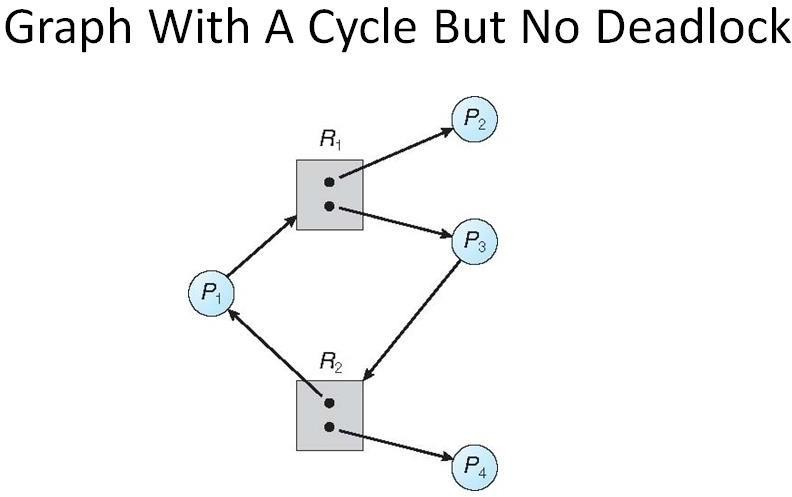
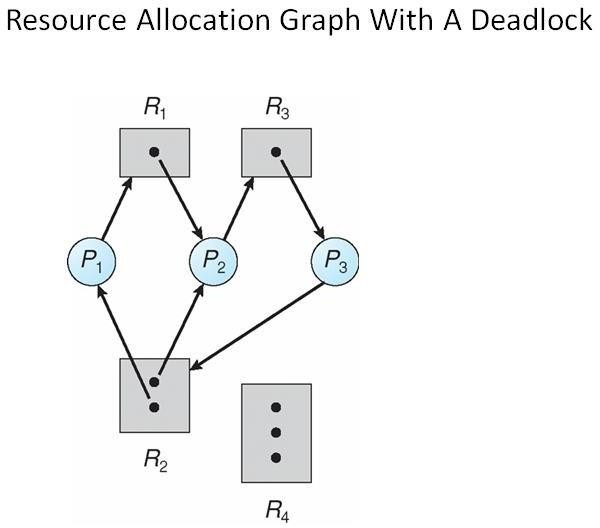
***Pi***

## wants *k* instances of resource type *Rj*

* If *Requesti*  *Needi* go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim
* If *Requesti*  *Available*, go to step 3. Otherwise *Pi* must wait, since resources are not available
* Pretend to allocate requested resources to *Pi* by modifying the state as follows:
* *Available* = *Available* – *Requesti;*
* *Allocationi* = *Allocationi* + *Requesti*;
* *Needi* = *Needi* –*Requesti;*
* If safe  the resources are allocated to *Pi*
* If unsafe  *Pi* must wait, and the old resource-allocation state is restored

1. **Explain symbols of resource allocation graph with Examples**





## UNIT 3 (MEMORY MANAGEMENT AND VIRTUAL MEMORY)

1. **What do you mean by Compaction?**

Compaction is **a technique to collect all the free memory present in form of fragments into one large chunk of free memory**, which can be used to run other processes.

1. **What are Pages and Frames?**

A page, also known as a memory page or virtual page, refers to **a contiguous block of virtual memory with a defined length that is described by one single entry in a page table**.

A frame refers to **a storage frame or central storage frame**. In terms of physical memory, it is a fixed sized block in physical memory space, or a block of central storage. In computer architecture, frames are analogous to logical address space pages

1. **What is the use of Valid-Invalid Bits in Paging?**

With the help of valid-invalid bit, **the system can know, when required, that pages C, D and E are not in the memory**. In short: a 1 in valid-invalid bit signifies that the page is in memory and 0 signifies that the page may be invalid or haven't brought into the memory just yet

1. **What is Virtual Memory?**

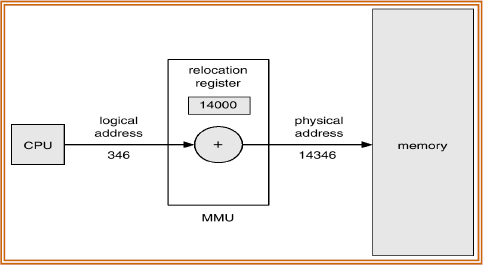
Virtual Memory is **a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory virtual** memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily transferring data from random access memory (RAM) to disk storage.

## What is Demand Paging?

## Demand paging follows that pages should only be brought into memory if the executing process demands them. This is often referred to as lazy evaluation as only those pages demanded by the process are swapped from secondary storage to main memory.

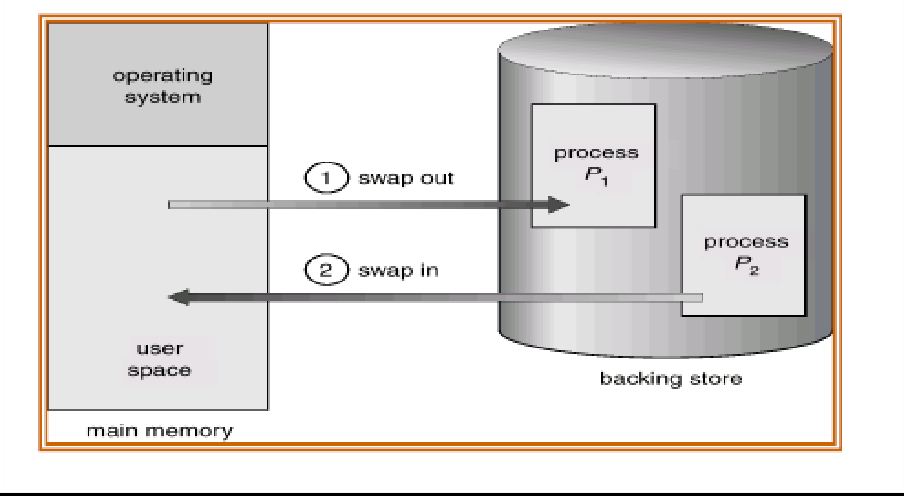
## Explain dynamic relocation using a relocation register.

* The set of all logical addresses generated by a program is a logical address space; the set of all physical addresses corresponding to these logical addresses are a physical address space
* Logical and physical addresses are the same in compile-time and load-time address- bindingschemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.
* The run-time mapping from virtual to physical addresses is done by a hardware device called the memory management unit (MMU)



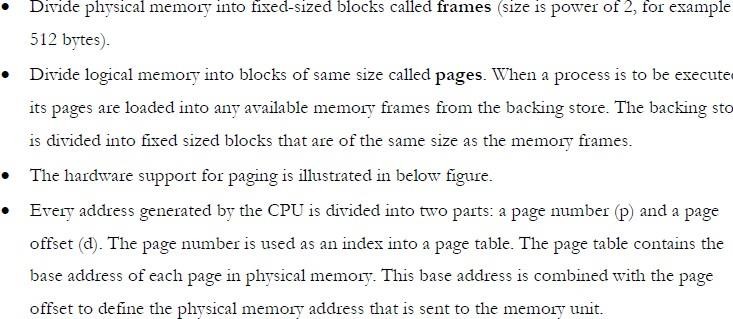
* This method requires hardware support slightly different from the hardware configuration. The base register is now called a relocation register. The value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
* The user program never sees the real physical addresses. The program can create a pointer to location 346, store it in memory, manipulate it and compare it to other addresses. The user program deals with logical addresses. The memory mapping hardware converts logical addresses into physical addresses. The final location of a referenced memory address is not determined until the reference is made.

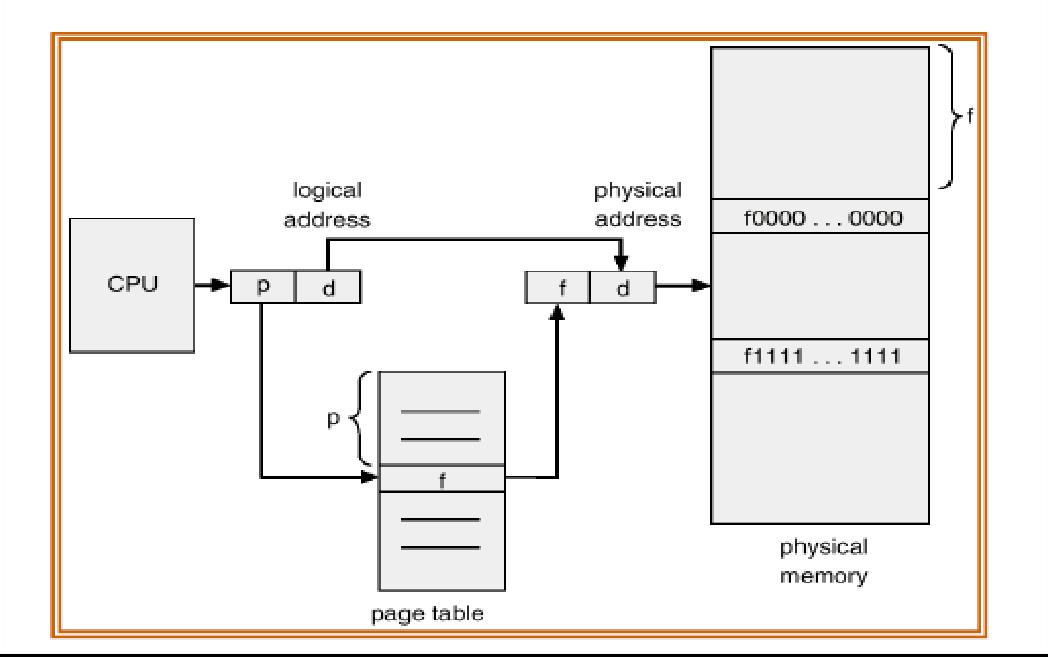
#### Explain swapping with neat diagram



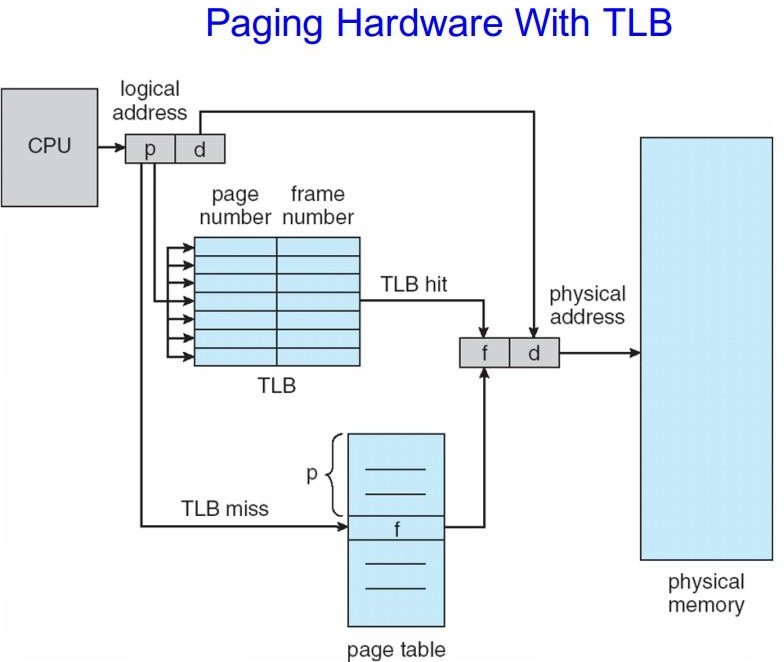
* A process can be swapped temporarily out of memory to a backing store, and then brought backinto memory for continued execution. For example, assume a multiprogramming environmentwith a round robin CPU scheduling algorithm. When a quantum expires, the memory managerwill start to swap out the process that just finished, and to swap in another process to thememory space that has been freed. In the mean time, the CPU scheduler will allocate a time sliceto some other process in memory.
* When each process finished its quantum, it will be swappedwith another process. Ideally, the memory manager can swap processes fast enough that someprocesses will be in memory, ready to execute, when the CPU scheduler wants to reschedule theCPU. The quantum must also be sufficiently large that reasonable amounts of computing aredone between swaps.
* A process can be **swapped** temporarily out of memory to a backing store, and then brought back into memory for continued execution
* Total physical memory space of processes can exceed physical memory
* **Backing store** – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
* **Roll out, roll in** – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
* Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped
* System maintains a **ready queue** of ready-to-run processes which have memory images on disk
* Modified versions of swapping are found on many systems (i.e., UNIX, Linux, and Windows).

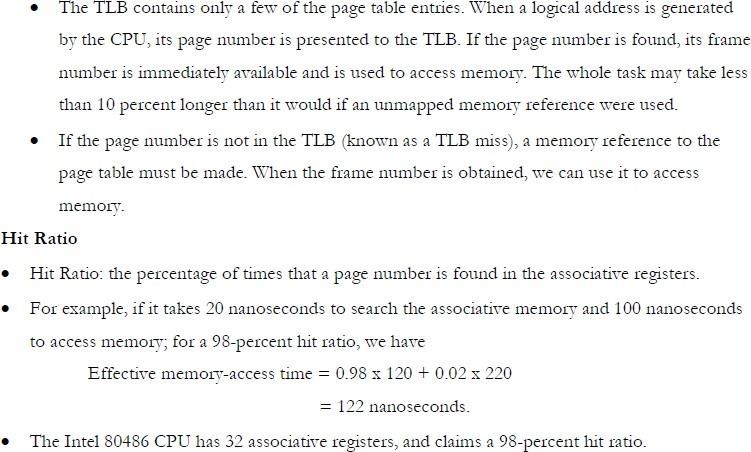
## What is paging? Explain the paging hardware?

Paging is a memory management scheme that permits the physical address space of a process tobe non contiguous.



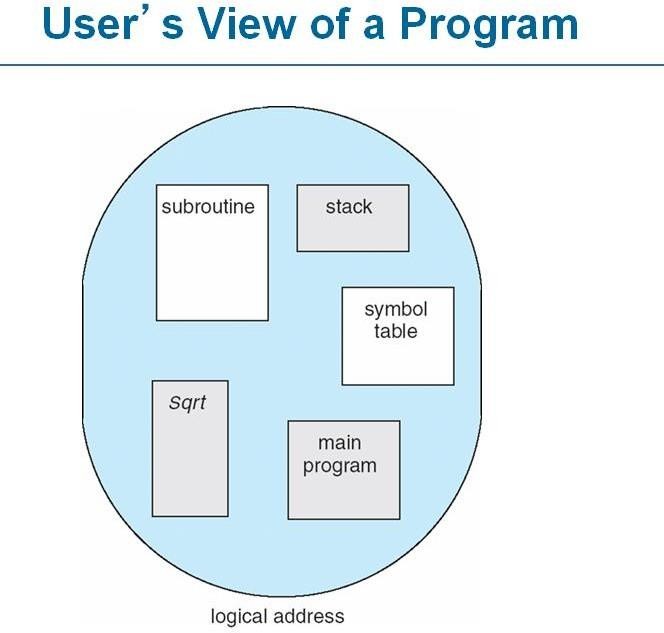
## Explain paging hardware with TLB

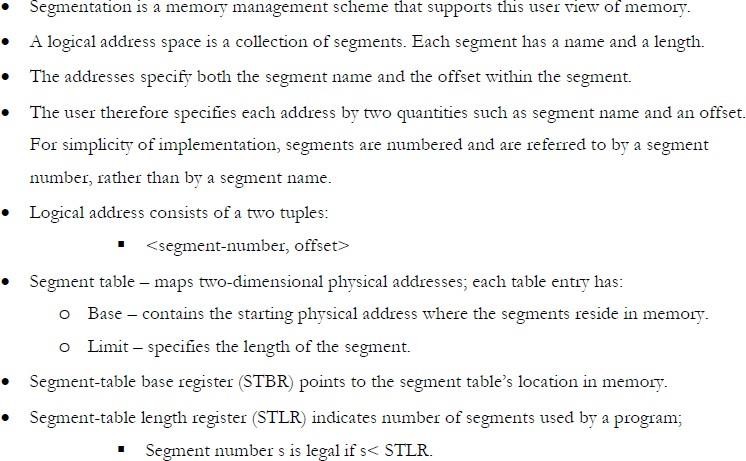


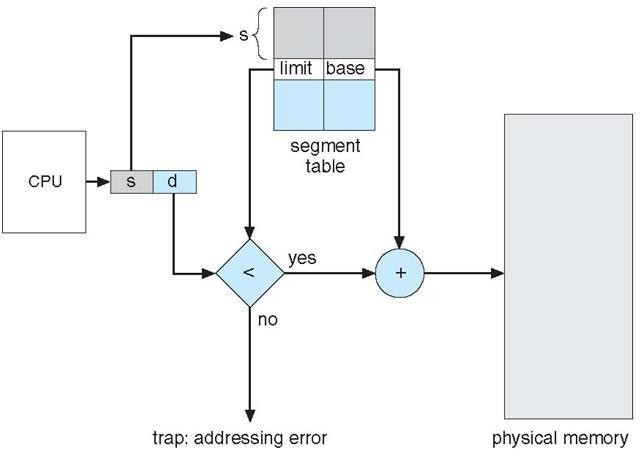
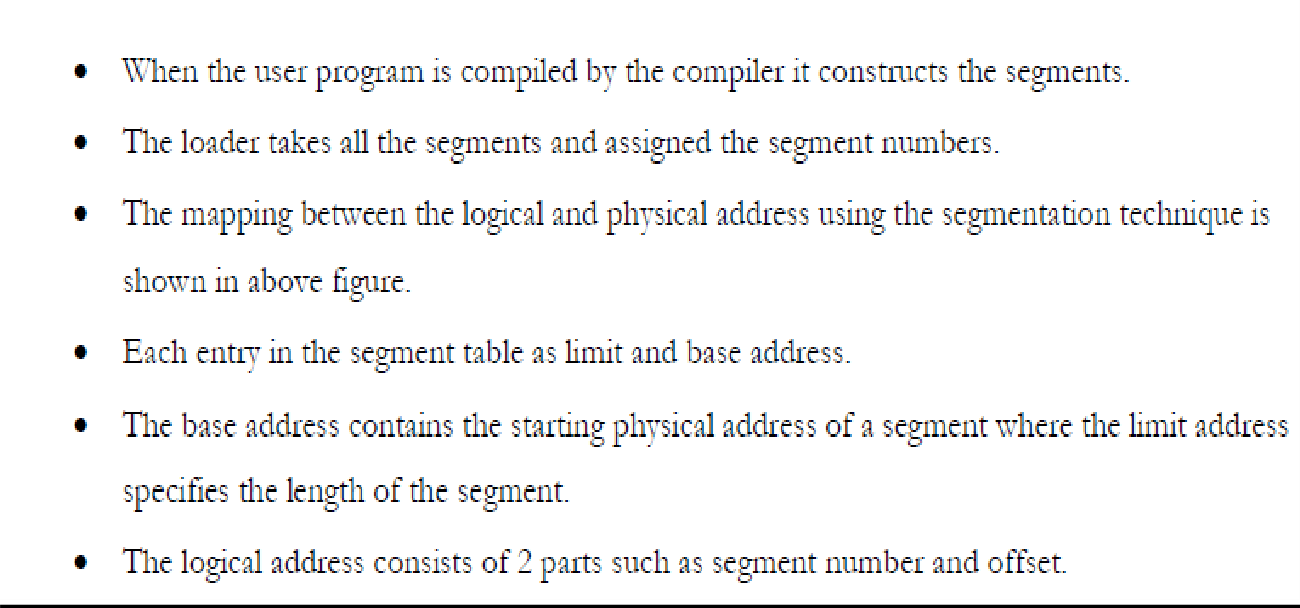


1. **Explain segmentation with diagram**
   1. Memory-management scheme that supports user view of memory
   2. A program is a collection of segments
   3. A segment is a logical unit such as:

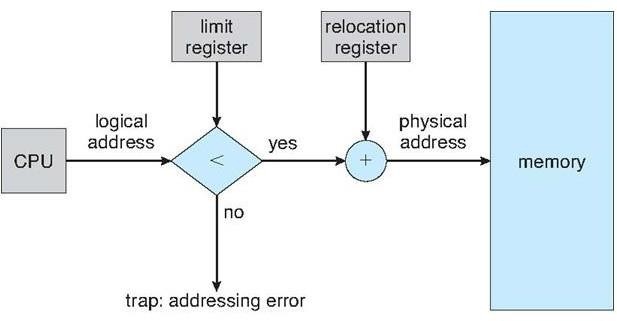
Main program, procedure, function, method, object, local variables, global variables, common block, stack, symbol table, arrays.

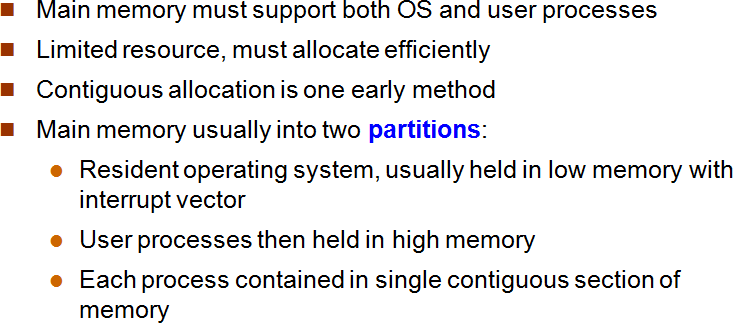


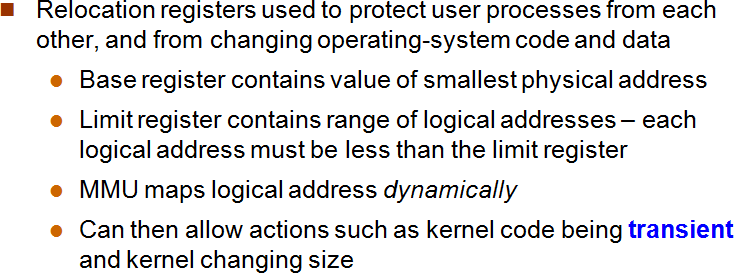




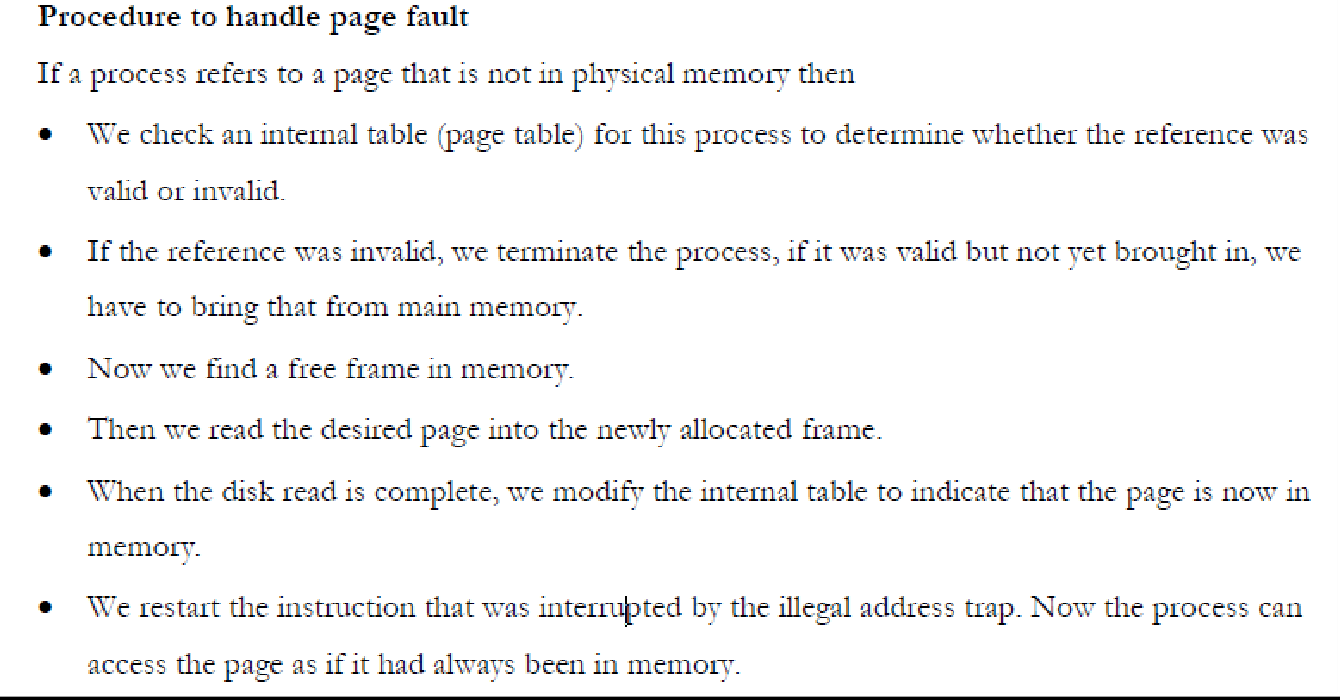
## Explain contiguous memory allocation with diagram.



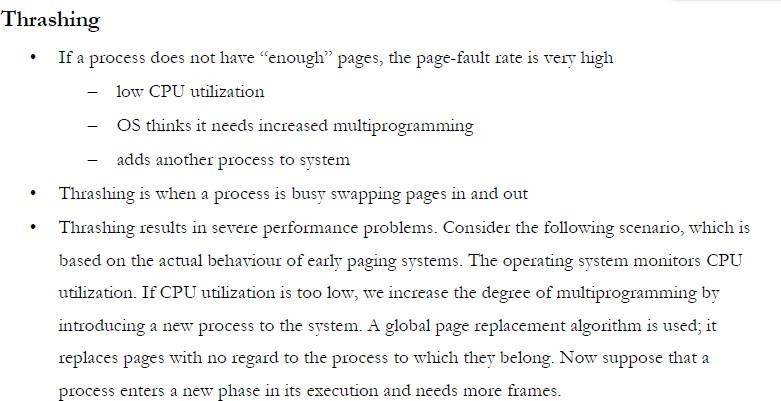


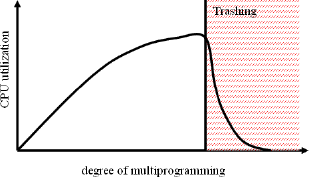


1. **Explain Procedure to handle page fault.**

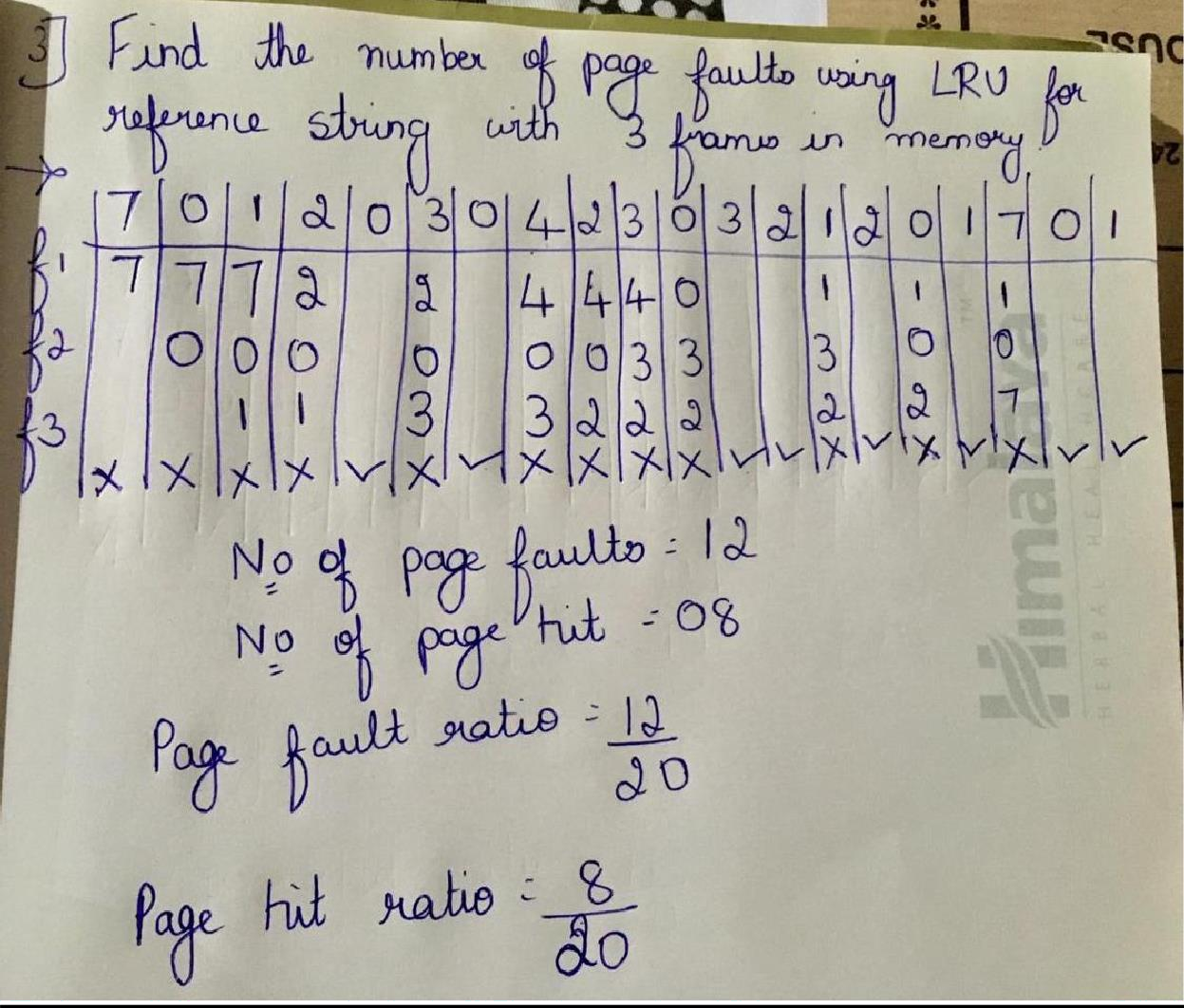


## Explain thrashing.





1. **Problems on LRU page replacement:**



## Explain page replacement with diagram

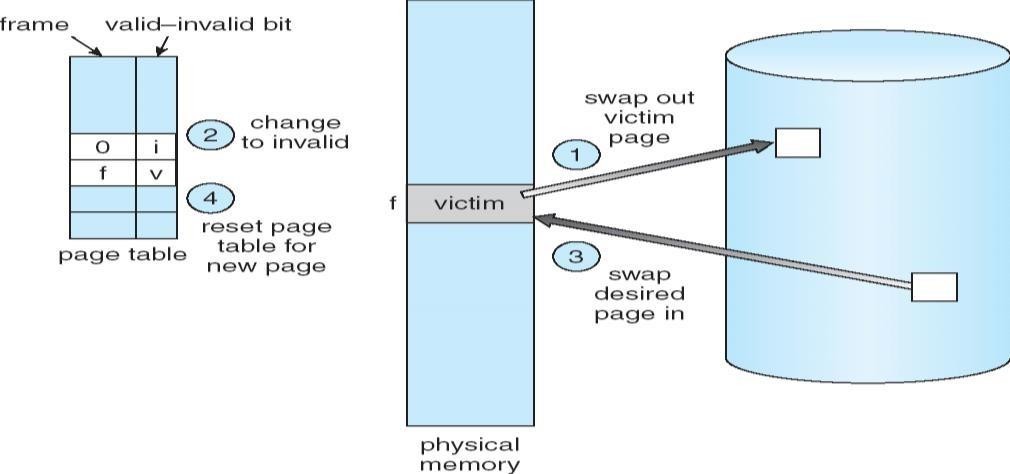
* Prevent over-allocation of memory by modifying page-fault service routine to include page replacement
* Use **modify (dirty) bit** to reduce overhead of page transfers – only modified pages are written to disk
* Page replacement completes separation between logical memory and physical memory – large virtual memory can be provided on a smaller physical memory

## Basic page replacement

* Find the location of the desired page on disk
* 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

* Read the desired page into the (newly) free frame. Update the page and frame tables.
* Restart the process



1. **Define Logical Address and Physical Address.**

**The logical address is rendered by CPU while a program is executed, whereas the physical address directs to a location in the memory unit**.

1. **What is the main function of the Memory-Management unit**

An important function of the Memory Management Unit (MMU) is **to enable the system to run multiple tasks, as independent programs running in their own private virtual memory space**

1. **Define Swapping**

Swapping is **a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes**.

1. **What is External and Internal Fragmentation?**

Internal fragmentation occurs when memory is divided into fixed-sized partitions. External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.

1. **What is Virtual Memory?**

Virtual memory is **a common technique used in a computer's operating system (OS)**. Virtual memory uses both hardware and software to enable a computer to compensate for physical memory shortages, temporarily transferring data from random access memory (RAM) to disk storage.

1. **Explain about advantages and disadvantages of paging? And Explain difference between paging and segmentation?**

## Advantages :

### Allocating memory is easy and cheap

### Any free page is ok, OS can take first one out of list it keeps

### Eliminates external fragmentation

### Data (page frames) can be scattered all over PM

## Disadvantages :

### Longer memory access times (page table lookup)

### Can be improved using TLB

### Guarded page tables

### Inverted page tables

| .NO | Paging | Segmentation |
| --- | --- | --- |
| 1. | In paging, the program is divided into fixed or mounted size pages. | In segmentation, the program is divided into variable size sections. |
| 2. | For the paging operating system is accountable. | For segmentation compiler is accountable. |
| 3. | Page size is determined by hardware. | Here, the section size is given by the user. |
| 4. | It is faster in comparison to segmentation. | Segmentation is slow. |
| 5. | Paging could result in internal fragmentation. | Segmentation could result in external fragmentation. |
| 6. | In paging, the logical address is split into a page number and page offset. | Here, the logical address is split into section number and section offset. |
| 7. | Paging comprises a page table that encloses the base address of every page. | While segmentation also comprises the segment table which encloses segment number and segment offset. |
| 8. | The page table is employed to keep up the page data. | Section Table maintains the section data. |
| 9. | In paging, the operating system must maintain a free frame list. | In segmentation, the operating system maintains a list of holes in the main memory. |
| 10. | Paging is invisible to the user. | Segmentation is visible to the user. |

**UNIT-4: FILE AND DISK MANAGEMENT**

# (SYSTEM, SECONDARY STORAGE STRUCTURE):

1. **What is a File?**

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks

1. **List the various File Attributes.**
   * Name
   * Identifier
   * Type
   * Location
   * Size
   * Protection
   * Time, data and User Identification
2. **What are the various File Operations?**
   * Creating a file
   * Writing a file
   * Reading a file
   * Delete a file
   * Truncating the file
   * Repositioning within a file
3. **Define the term seek time, rotational latency and disk bandwidth.**

Seek Time is defined as **the time required by the read/write head to move from one track to another**

Rotational latency (in milliseconds) **describes the time required to position a specific sector under the read–write head**. Average latency is typically given as the time it takes the drive to perform half a rotation of the platter, and is directly dependent on its RPM rating.

The disk bandwidth is **the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer**.

1. **What are the different Accessing Methods of a File?**
   * Sequential Access
   * Direct Access
2. **Mention different file access types.**

: **Sequential-Access, Direct Access, Index sequential Method**

1. **What is Directory?**

A **directory** is a location for storing files on your computer. **Directories** are found in a hierarchical file system, such as Linux, MS-DOS, and Unix.

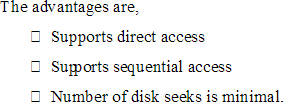
1. **Define Seek Time and Latency Time.**

Seek Time is measured defines the amount of time it takes a hard drive's read/write head to find the physical location of a piece of data on the disk. Latency is the average time for the sector being accessed to rotate into position under a head, after a completed seek.

1. **List disadvantages of using a single directory.**

**Disadvantages**

* We cannot have two files with the same name.
* The directory may be very big therefore searching for a file may take so much time.
* Protection cannot be implemented for multiple users.
* There are no ways to group same kind of files.

1. **What are the advantages of Contiguous Allocation?**
2. **List the advantages and disadvantages of two-level directory?**
3. **What do you mean by sequential access method**
4. **Define direct access method in files.**
5. **List and explain the various File Attributes.**
6. **Explain various file operations**
7. **What are the advantages of Contiguous Allocation?**
8. **List the advantages and disadvantages of two-level directory?**

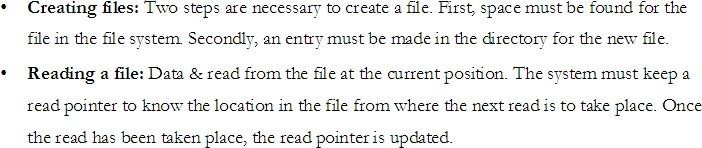
**Advantages**

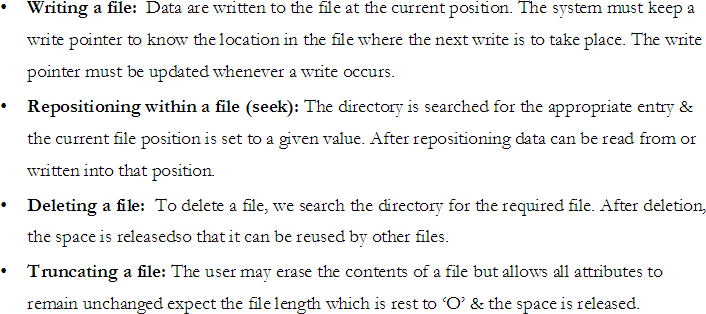
**Different users can have the same file name**. Searching becomes more efficient as only one user's list needs to be traversed. The same kind of files cannot be grouped into a single directory for a particular user.

**Disadvantages**

Although the two-level directory structure solves the name-collision problem, it still has disadvantages. This structure **effectively isolates one user from another**.

1. **List and explain the various File Attributes.**
2. **Name.** It is the only information which is in human-readable form.
3. **Identifier**. The file is identified by a unique tag(number) within file system.
4. **Type**. It is needed for systems that support different types of files.
5. **Location**. Pointer to file location on device.
6. **Size**. The current size of the file.
7. **Protection**. This controls and assigns the power of reading, writing, executing.
8. **Time, date, and user identification**. This is the data for protection, security, and usage monitoring.
9. **Explain various file operations**





1. **Explain various file operations**

**1.Create operation:**

This operation is used to create a file in the file system. It is the most widely used operation performed on the file system. To create a new file of a particular type the associated application program calls the file system. This file system allocates space to the file. As the file system knows the format of directory structure, so entry of this new file is made into the appropriate directory.

**2. Open operation**

This operation is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.

**3. Write operation:**

This operation is used to write the information into a file. A system call write is issued that specifies the name of the file and the length of the data has to be written to the file. Whenever the file length is increased by specified value and the file pointer is repositioned after the last byte written.

**4. Read operation:**

This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.

**5. Re-position or Seek operation:**

The seek system call re-positions the file pointers from the current position to a specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files.

**6. Delete operation:**

Deleting the file will not only delete all the data stored inside the file it is also used so that disk space occupied by it is freed. In order to delete the specified file the directory is searched. When the directory entry is located, all the associated file space and the directory entry is released.

**7. Truncate operation:**

Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.

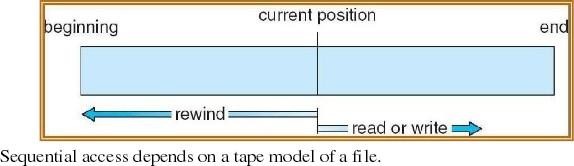
**8. Close operation:**

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

1. **Explain different file accessing method**

# Sequential Access:

1. Sequential access is the simplest access method. Information in the file is processed in order, one record after another.
2. Editors and compilers access the files in this fashion.
3. Normally read and write operations are done on the files.
4. Aread operation reads the next portion of the file and automatically advances afile pointer, which track next i/I track.
5. Write operation appends to the end of the file and such a file can be next to the beginning.



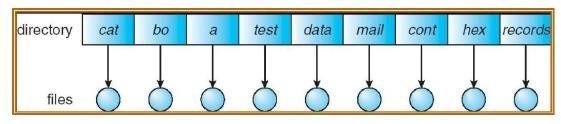
**Direct Access**

1. Direct access allows random access to any file block.
2. This method is based on disk model of afile.A file is made up of fixed length logical records. It allows the program to read and write records rapidly in any order.A direct access file allows arbitrary blocks to be read or written.
3. **Eg**:-User may need block 13, then read block 99 then write block 12. For searching the records in large amount of information with immediate result, the direct access method is suitable. Not all OS support sequential and direct access. Few OS use sequential access and some OS uses direct access.
4. It is easy to simulate sequential access on a direct access but the reverse is extremely inefficient.

## Explain the various Directory structures

1. **Single level directory:** It is the simplest directory structure. All files are present in the same Directory. So it is easy to manage & understand.

**Limitation:** A single level directory is difficult to manage when the no. Of files increases or When there is more than one user. Since all files are in same directory, they must have unique Names. So, there is confusion of file names between different users.

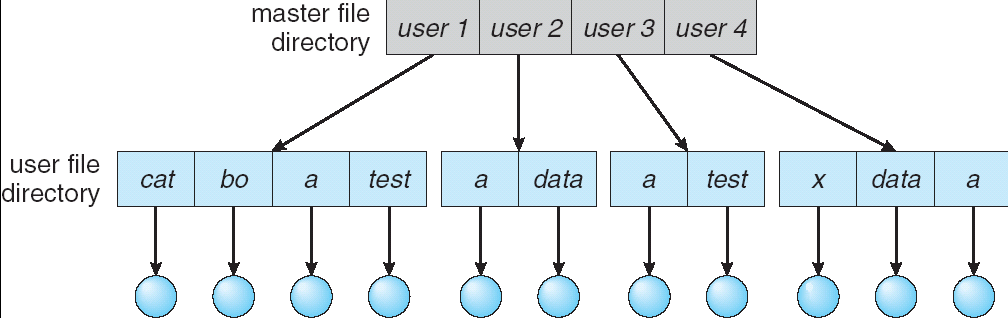


1. **Two Level Directory:**

The solution to the name collision problem in single level directory is to

Create aseparate directory for each user. In atwo leveldirectory structure, each user has its own user file directory. When a user logs in, then master file directory is searched. It is indexed by user name & each entry points to the UFD of that user.

**Limitation:** It solves name collision problem. But it isolates one user from another. It is an advantage when users are completely independent. But it is a disadvantage when the users need



## Tree structured directories:

* + It is the most common directory structure. A two level directory is a two level tree. So, the generalization is to extend the directory structure to a tree of arbitrary height.
  + It allows users tocreate their ownsubdirectories & organize their files. Every file in the system has a unique path name. It is the path from the root through all the sub-directories to a specified file.
  + A directory is simply another file but it is treated in a special way. One bit in each directory entry defines the entry as a file (O) or as sub- directories. Each user has a current directory.
  + It contains most of the files that are of current interest to the user. Path names can be of two types: An absolute path name begins from the root directory & follows the path down to the specified files. A relative path name defines the path from the current directory.

If the current directory is root/spell/mail, then the relative path name is prt/first & the absolute pathname is root/ spell/ mail/ prt/ first. Here users can access the files of other users also by specifying their path name

1. **With a diagram explain file system structure**

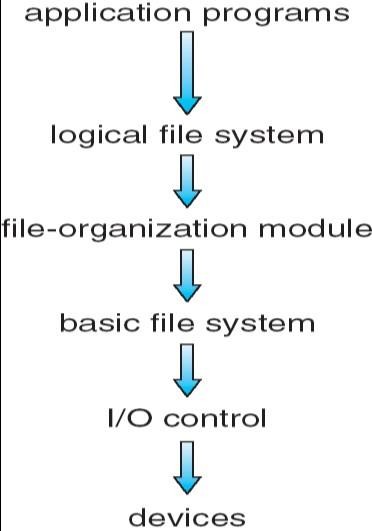
To provide efficient and convenient access tothe disks, theOS provides the filesystem to allow the data to be stored, located and retrieved. A file system has two design problems:

* 1. **How the file system should look to the user.**
  2. **Selecting algorithms and data structures that must be created to map logical file**

system on to the physical secondary storage devices. The file system itself is composed of

different levels. Each level uses the feature of the lower levels to create new features for use by higher levels.

**The following structures shows an example of layered design**



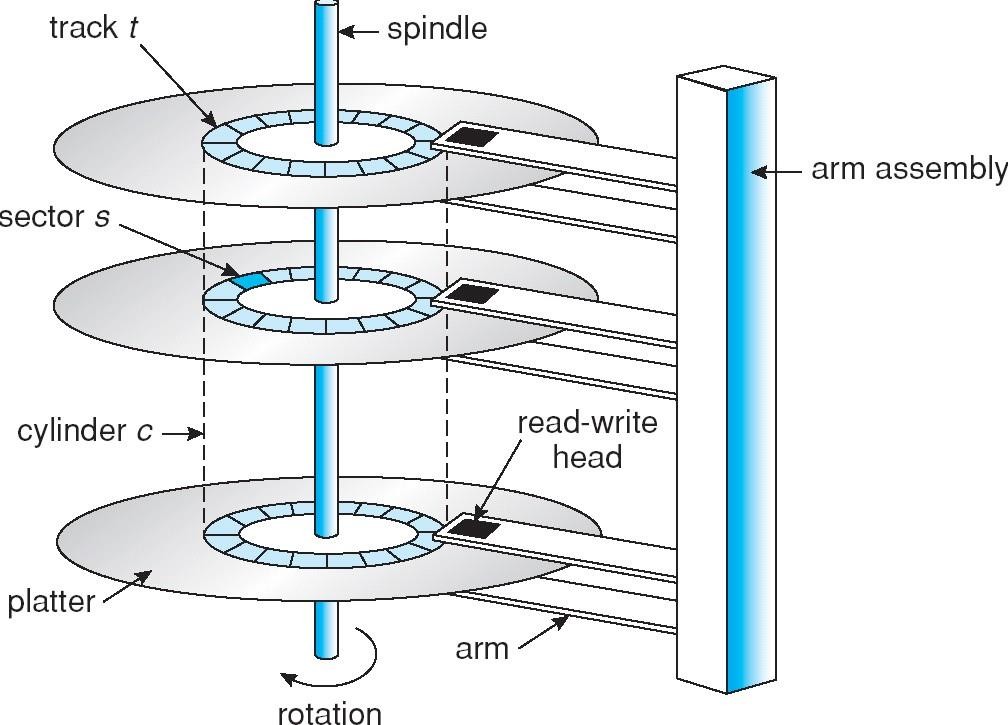
The lowest level is the i/o control consisting of device drivers and interrupt handless to transfer the Information between memory and the disk system.

The device driver is like a translator. Its input is a high level command and the o/p consists of low level hardware specific instructions, which are used by the hardware controllers which interface I/O device to the rest of the system. The basic file system needs only to issue generic commands to the appropriate device drivers to read and write physical blocks on the disk. The file organization module knows about files and their logical blocks as well as physical blocks. By knowing the type of file allocation used and the location of the file, the file organization module can translate logical

1. **Define SCAN and C-SCAN scheduling algorithms.**
   1. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.
   2. **C-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.
2. **Define swap space management?**

Swap space management is another low-level task of the operating system. Virtual memory uses disk space as an extension of main memory. Since disk access is much slower than memory accessf using swap space significantly decreases system

## Explain Magnetic Disk Structure



**One or more platters in the form of disks covered with magnetic media.**

* Each disk platterhas a flat circular shape.
* Each platter has two working **surfaces** that stores data.
* Digital information is stored on magnetic disks in the form of microscopically small, magnetized needles.
* Data is stored on either or both surfaces of discs in concentric rings called tracks.
* Each track is divided into a number of sectors.
* To read information, the arm is positioned over the correct track.
* Data isreadandwrittenbya diskdrivewhichrotates thediscs andpositions the read/write heads over the desired track.

**WRITE MECHANISM**

-> Writes binary data by magnetizing small areas or zones of the disk.

-> Current through the coil produces magnetic field.

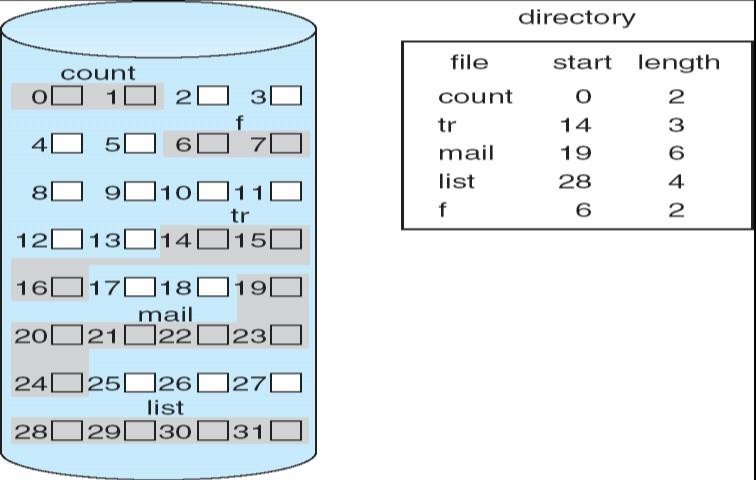
->Corresponding magnetic pattern is recorded on the surface.

**READ MECHANISM**

->It reads data by detecting current pulses induced in a coil.

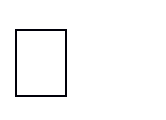
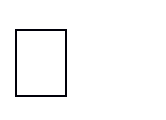
->Bits could be read back as a time sequence of pulse (one) or no pulse (zero).

1. **Explain contiguous allocation method with neat diagram**

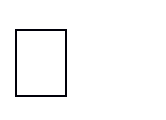


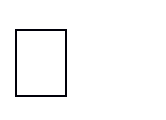
In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: *b, b+1, b+2,……b+n-1.* This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file.

The directory entry for a file with contiguous allocation contains

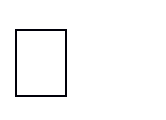
Address of starting block Length of the allocated portion.

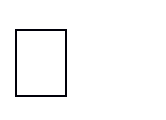
**Advantages:**

Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).

This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

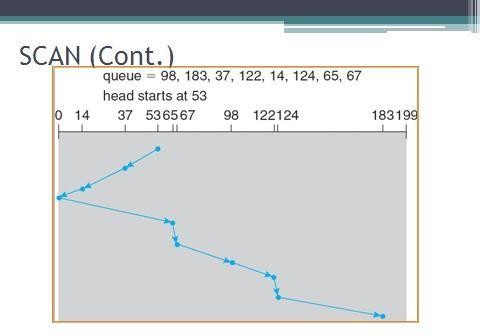
### Disadvantages:

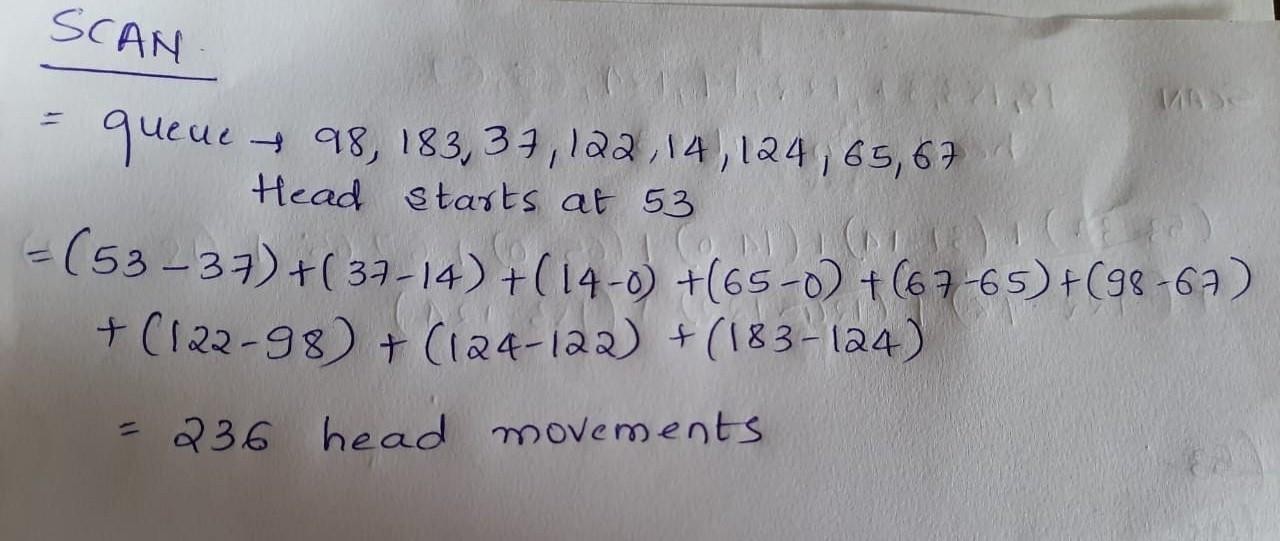
This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.

Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

**24.Explain scan and c-scan scheduling algorithm with example**

**Note: SCAN :** In the Scan Algorithm, the disk starts at one end of the disk and moves toward the 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 movements reversed, and servicing continues. The head continuously scans back and forth across the disk. The SCAN algorithm is sometimes called elevator algorithm ,the since the disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.





**C-SCAN**

Circular Scan scheduling is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip (Figure 12.7). The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.

