OPERATING SYSTEM & SYSTEM PROGRAMMING

4TH SEMESTER, COMPUTER SCIENCE ENGINEERING/ INFORMATION TECHNOLOGY

CONTENTS

CHAPTER - 1

OPERATING SYSTEM

System is nothing but environment. Environment consists of components.

Operating system:-

⇒ The basic objective of the operating system is to operate several components associated with computer system.

⇒ System is an environment consists of several components.

⇒ System having two components (basically)

i) Hardware components

ii) Software components.

i) Hardware components:-

⇒ There are physical components.

⇒ The components which are visible, touchable part of the computer system that can perform basic function is known as hardware.
Eg. Input devices, O/P devices, m/m, Processor.

ii) Software components:-

⇒ These are the untouchable components having logical existence.

⇒ These are the set of programs that controls the operation of the physical components.

⇒ Programs are the set of instructions.

$$\begin{tikzcd}
 & \text{Software} \arrow[dl, no head] \arrow[dr, no head] & \\
\text{System S/W} & & \text{Application S/W}
\end{tikzcd}$$

System software:-

⇒ It is meant for running of a system.

⇒ It is a set of programs designed to make function able of different components.

Application software:-

⇒ These are the S/W related to user's requirement.

⇒ Thus application software can't be executed alone. It always take the help of system software to be executed.

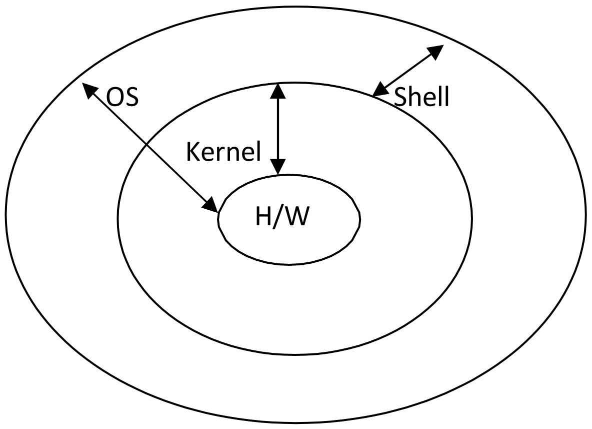
⇒ These are the set of programs that are designed to execute diff. app. Program. e.g. word processor, spread sheet, railway/ air reservation.

OS:- it is a system software. It is acting as an interface between user and hardware.

Components of OS:- Basically OS divides into two components, i.e.

i) Kernel

ii) Shell



i) <u>Kernel:</u>

- ⇒ It is the core part of the OS.
- ⇒ This part of OS deals with h/w (hardware instructions).
- ⇒ It is that part of OS who is always in a running mode.

ii) <u>Shell:-</u>

- ⇒ It is that part of OS who is directly related to the user.
- ⇒ Basically it deals with a high level language or commands or instruction.
- ⇒ It is also acting as a command interpreter.

<u>Relationship between shell and kernel:-</u>

- ⇒ A shell takes the instruction from the user through a high level language.
- ⇒ Then it converts the high level language to a machine level language (0,1 form) through an interpreter.
- ⇒ After getting instruction from the shell, kernel instructs the appropriate hardware to be executed.

<u>Views of an OS:-</u>

There are two views related to an OS, i.e.

i) User view

ii) System view.

i) <u>User view:-</u> In user view the operating system is designed for only fulfilling the requirement of the user.

ii) <u>System view:</u>- In this view OS is designed to be a resource allocator or resource manager.

Q. OS acts as an resource manager. Justify your answer?

A.

- ⇒ A computer system has many resources which are required to complete a task.
- ⇒ OS acts as a resource manager to these resources, such as I/O devices, storage device, memory, CPU time and so on…..
- ⇒ OS allocates these resources to specific processes according to their requirement.
- ⇒ OS uses two methods for sharing or managing the resource.
  - i) Time sharing
  - ii) Space sharing

i) <u>Time sharing:-</u>

- ⇒ When a resource is a time sharing resource, first one of the task gets the resource for some period of time, then the another resource and so on…..
- ⇒ In timesharing method time period is being divided into no. of slices termed as time slice or time quantum.

ii) <u>Space sharing:-</u>

- ⇒ In space sharing method user shares the space of the resource.
- ⇒ Memory is an example of space sharing method.
- ⇒ Memory will be divided into no. of segments. Several processes can be loaded into the memory at a time, so that CPU performance will be more.

According to the functionality of the processor, system will be categorized into two types.

i) Single processor system
ii) Multi processor system

i) Single processor system:- The system which having one processor is known as single processor system. In a single processor system the only CPU is capable of executing general purpose instructions as will as the user process instructions.

ii) Multi processor system:- The system having more than one processor is known as multiprocessor system. In this system multiple no. of processor are connected with each other, sharing the memory peripheral devices, clock etc.

Advantages of multi processor system:-

Following are the advantages:-

Increased throughput:-

- ⇒ Throughput will measure the performance of CPU.
- ⇒ By increasing the processor more work can be done in less time.

Economy of scale:-

- ⇒ Multiprocessor system is less expensive than multiple no. of single processor because memory decrease area.

3. increase the reliability:-

- ⇒ If functions are distributed among different processor, then failure of one processor will not hault the total system, it will just reduce the performance level.
- ⇒ Examples of multiprocessor system:- super computer, main frame computers.
  According to the user accessibility OS are of 2 categories:-
  i) Single user operating system
  ii) Multiuser operating system

i) Single user operating system :- the OS which allows only one user to work at a time is known as single user OS.
Ex. MS DOS.

ii) Multiuser operating system:- the OS which allows multiple no. of users to work at a time is known as multiuser OS.
Ex.UNIX, LINUX, etc.

Functions of an OS:-

Functions of an OS has been divided into different modules.
i.e. 1. Process management, 2. Memory management.

1. Process management:-

- ⇒ This module of an OS takes care of creation and deletion of different processes.
- ⇒ It will allocate processes to different resources by using various scheduling algorithm.
- ⇒ It also make a co-ordination and communication within different processes by performing the synchronization.

Memory management:-

⇒ This module of an OS takes care the allocation and deallocation of memory to various processes according to their requirement.

File management:-

⇒ A system consists of huge number of files.

⇒ A file is a container to store data and information for the future use.

⇒ File management module of an OS involves keeping track of all the different files.

⇒ It will maintain the integrity to the data stored in the file and directories.

⇒ All the request from the user process related to a file and directory is handled by this module of an OS.

Security:-

⇒ In this module of an OS protect the resources and information of an computer from the destruction and unauthorized access.

Input output device management:-

⇒ Co-ordination and control of input output device is an important and primary function of an OS.

⇒ This module involves receiving the request for an input output interrupt and communicating back to the request processes.

Command interpretation:-

⇒ This module of an OS takes care of interpreting the user commands and directing the system resource to bundle the hardware.

Types of OS:-

⇒ Single user

⇒ Multi user

⇒ Single processor

⇒ Multi processor

⇒ Batch OS (serial Processing)

Batch OS:-

⇒ The initial processing system is used by the user is known as serial processing.

⇒ In serial processing the programs are executed.

On after another:-

⇒ The problems arises in serial processing is overcome through another OS is known as batch OS.

Batch:-

⇒ A batch is formed by grouping the jobs with similar.

Need:-

⇒ In this type of processing same type of job batches together and processor executes each batch in sequential manner.

⇒ Processing all jobs of a batch as a single process is known as batch processing system.

⇒ This is the oldest operating system used by the use.

Disadvantages:-

⇒ The waiting time of OS is larger for collecting the similar types of jobs.

⇒ Average waiting time increases with increase of no. of jobs.

Multiprogramming / Time sharing/ Multitasking system:-

⇒ Logical extension of multiprogramming is called time sharing.

⇒ The OS which executes multiple no. of jobs at a time is known as multiprogramming system.

⇒ In this OS the user can perform more than one job at a time.

⇒ Multitasking / times sharing Os is a logical extension of multiprogramming system.

⇒ In timesharing system multiple jobs can execute by sharing the CPU time among different process time slice or time quantum.

Advantages:-

⇒ Though multiple no. of jobs can be executed at a time, it will increase the performance of the CPU as well as fulfilling the request of user.

Disadvantages:-

⇒ When huge no. of processes are requesting for execution the processor is busy in switching from one process to another process by which the efficiency or performance level of the processor is generally decreases.

Real time system:-

⇒ It is associated with deadlines. A system is said to be real time system if every task is executed must satisfying the deadlines. (A restriction related to a time period is called deadline).

⇒ It works towards providing immediate procession and also responding to user commands in very short time.

Eg. RTOs (maruti, OS9, Harmony etc)

⇒ Two types of real time system

i) Hard real time system

ii) Soft real time system

Q: Difference between hard and soft real time system?

Hard real time system	Soft real time system
⇒ In hard real time system deadline must be satisfied otherwise the system is failure.	⇒ In soft real time system some deadlines are relax able (deadlines are not mandatory).
⇒ This type of OS is diff to design.	⇒ This OS is simple and easy to design.
⇒ Eg. Satellite launching missiles testing, research in chemical industries etc.	⇒ Eg. Automobile full ingestion. Washing m/c, microwave etc.
⇒ These are larger in application.	⇒ Small scale application
⇒ Application of hard real time system is restricted (not frequently used).	⇒ Application of soft real time system is widely used.
⇒ It is larger in size .	⇒ Smaller in size.

1. Time sharing system is logical extension of multiprogramming system Justify.
2. Differentiate between hard RT task and Soft RT task with example.
3. Why the multiprogramming approach sometimes reduce/decreases the CPU performance.

Distributed OS (DOS):-

⇒ This OS is known as loosely coupled system.

⇒ A DOS hides the existence of multiple computer 4m the user.

- ⇒ Here multiple computers are used to process the data.
- ⇒ Multiple computers are communicating with each other through various communication lines such as high speed buses.
- ⇒ DOS provides a single system image to its use.

Advantages:-

i) Resource Sharing:- If no. of sites are connected with each other through high speed communication line it is possible to share the resources from one site to another.

ii) Computation speedup:- Though a big task can be distributed into no small tasks so the computation speed is gradually increase to complete the task.

iii) Reliability:- If the resources of the system is failure at site the user can use the resources from another site.

Parallel System:-

- ⇒ Using parallel processing a system can perform several computation job simultaneously in less time.
- ⇒ In this system the task of an OS to support processing by making the division of process it can run on different processor.
- ⇒ It is otherwise known as tightly coupled system.

Network OS:- (NTOS)

- ⇒ This system which includes s/w to communicate with each other via a n/w.
- ⇒ This Os which can be used for networking purpose.
- ⇒ It is the OS which is meant for data transmission through a network.
- ⇒ h/w OS provides support for multiuser operations as well as administrative security and it network management function.

  Eg. Novel s/w, MSLAN Manager.

Services of an OS:-

i) User interface:-

- ⇒ Interface is a medium through which the user can communicate (interact) with kernel (OS).
- ⇒ There are 2 approaches for interaction.
  - a) CLI – Command line interface.
  - b) GUI – Graphical user interface.

a) CLI:-

- ⇒ It is otherwise known as command interpreter which allows the user to directly enter the commands that are to be performed by the OS.
- ⇒ The main function of command interpreter is to get and execute the user specified commands.

  Eg.MS-DOS

b) GUI:-

- ⇒ This approach allow to interact with the OS through a windows system known as GUI.
- ⇒ MAC OS is developed as 1st OS.

ii) <u>Program execution:-</u>

- ⇒ A program is executed to fulfill the use requirement.
- ⇒ Every program before execution should be loaded in the memory.
- ⇒ A program should end its execution in a new and abnormal way.
- ⇒ After the completion of the execution, a put will terminate.

iii) <u>I/O operations:-</u>

- ⇒ Various I/O devices are required for I/O operations to execute a program.
- ⇒ These resources or devices should be provided by the OS.

iv) <u>File management / file system manipulation:-</u>

⇒ in file system, following activities are performed.

i) Creation of a file
ii) A program needs to read/write into file or directory.
iii) Deletion of file or directory by their
iv) Lists the file information
v) Searching of a file
vi) Giving permission to access a file.

All the above file manipulation activities are performed by the instruction of OS.

v) <u>Communication:-</u>

⇒ Communication between the processes occurs in 2 ways

i) Shared memory
ii) Message passing

⇒ Communication occurs between the processes either in the same computer or in different computer.

vi) <u>Error detection:-</u>

⇒ Following errors can occur to the system

i) Power failure
ii) Availability of memory
iii) Unavailability of memory
iv) Connection failure in network
v) System failure
vi) Arithmetic overflow
vii) Attempt to access illegal memory location
viii) Stack overflow

All the above errors has been detected by the OS and OS gives instruction to handle those error.

vii) <u>Resource allocation:-</u>

- ⇒ In multiprogramming system resources must be allocated to each process.
- ⇒ All the resource of computer system are managed by the OS.

viii) <u>Security:-</u>

Security involves providing authentication to outsider by giving an user name.

ix) <u>Protection:-</u>

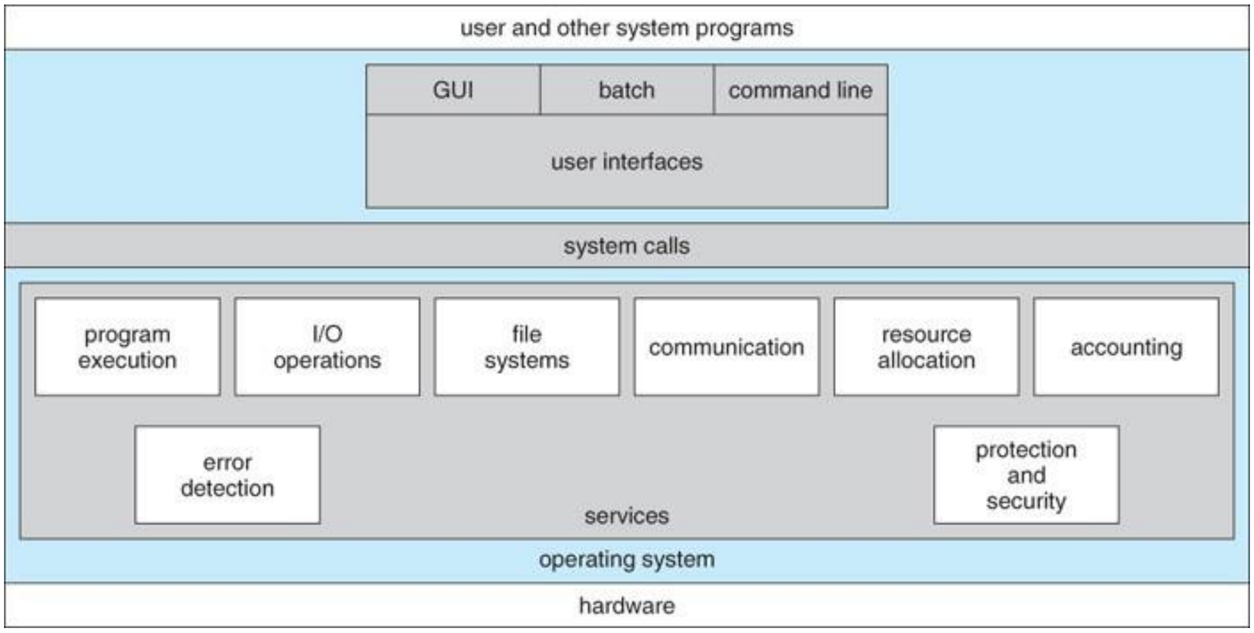
Protection involves ensuring that all the access to the system resources is controlled.

x) <u>Accounting:-</u>

⇒ It specifies to accumulate usage statistics.

⇒ OS keeps track all the information regarding the amount and types of resources used by diff. processes.

<u>Structure of Operating System:</u>



- User Interfaces - Means by which users can issue commands to the system. Depending on the system these may be a command-line interface (e.g. sh, csh, ksh, tcsh, etc.), a GUI interface (e.g. Windows, X-Windows, KDE, Gnome, etc.), or a batch command systems. The latter are generally older systems using punch cards of job-control language, JCL, but may still be used today for specialty systems designed for a single purpose.
- Program Execution - The OS must be able to load a program into RAM, run the program, and terminate the program, either normally or abnormally.
- I/O Operations - The OS is responsible for transferring data to and from I/O devices, including keyboards, terminals, printers, and storage devices.
- File-System Manipulation - In addition to raw data storage, the OS is also responsible for maintaining directory and subdirectory structures, mapping file names to specific blocks of data storage, and providing tools for navigating and utilizing the file system.
- Communications - Inter-process communications, IPC, either between processes running on the same processor, or between processes running on separate processors or separate machines. May be implemented as either shared memory or message passing, (or some systems may offer both.)
- Error Detection - Both hardware and software errors must be detected and handled appropriately, with a minimum of harmful repercussions. Some systems may include complex error avoidance or recovery systems, including backups, RAID drives, and other redundant systems. Debugging and diagnostic tools aid users and administrators in tracing down the cause of problems.

Other systems aid in the efficient operation of the OS itself:

- Resource Allocation - E.g. CPU cycles, main memory, storage space, and peripheral devices. Some resources are managed with generic systems and others with very carefully designed and specially tuned systems, customized for a particular resource and operating environment.
- Accounting - Keeping track of system activity and resource usage, either for billing purposes or for statistical record keeping that can be used to optimize future performance.
- Protection and Security - Preventing harm to the system and to resources, either through wayward internal processes or malicious outsiders. Authentication, ownership, and restricted access are obvious parts of this system. Highly secure systems may log all process activity down to excruciating detail, and security regulation dictate the storage of those records on permanent non-erasable medium for extended times in secure (off-site) facilities.

<u>System call:-</u>

⇒ It provides an interface to the services provided by an OS.

⇒ System call provides a request by the user to the OS to perform some task.

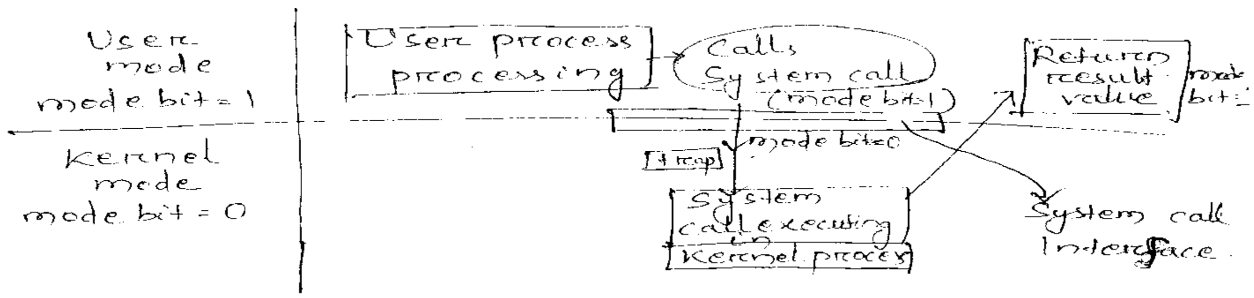
⇒ These calls are generally in the form of languages such as C, C++, assemble etc.

⇒ System call is executed in 2 modes.

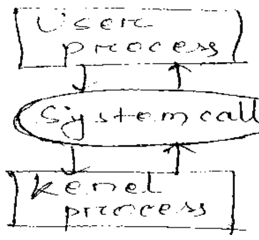
i) Kernel mode

ii) User mode

⇒ The user request is processed by the help of the system call through these modes (Kernel and user) is termed as dual mode operation.



How kernel communicate :-



⇒ A bit is added to the hardware component of the computer to keep track the current mode of execution is known as mode bit.

⇒ The mode bit is used to distinguish between user mode and kernel mode.

⇒ Mode bit is specified as kernel mode = 0, user mode = 1.

⇒ Kernel mode is otherwise known as privilege mode or supervisor mode.

⇒ When the computer system is working in executing the user application, then it is in user mode.

⇒ When a user application request a service from OS, then it must transit from user mode (1) to kernel mode (0) to fulfill the task.

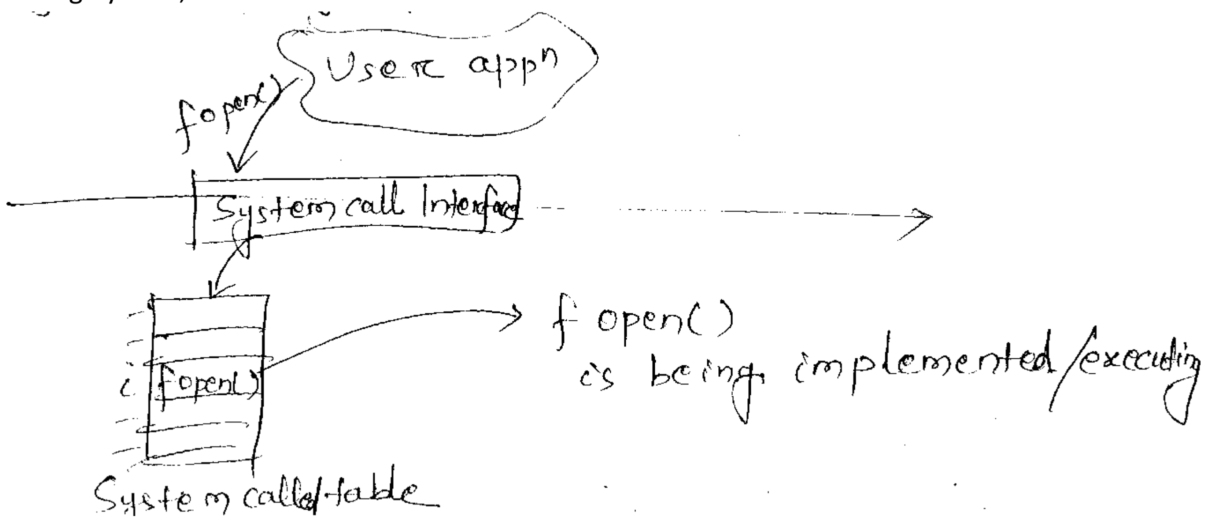
Trap:-

⇒ Trap is a software interrupt.

⇒ It is an abnormal situation or condition detected by the CPU.

⇒ It is also assumes as an indication of error or fault or exception.

Eg: Dividing by Zero, Stack overflow



⇒ A system call interface provides a runtime support for most of the programming language.

⇒ A table is managed by the OS named as system called table where the system calls are maintained through the index value.

⇒ By the help of the system call table appropriate system call is executed and returned the status of the system call or returns any value to user process.

Question:

1. What is Operating System?
2. What is interrupt?
3. Draw the structure of O.S.

CHAPTER - 2

PROCESS MANAGEMENT

Process:-

⇒ A program in execution.

⇒ Process is a currently executable task.

⇒ Process execution must progress in a sequential manner.

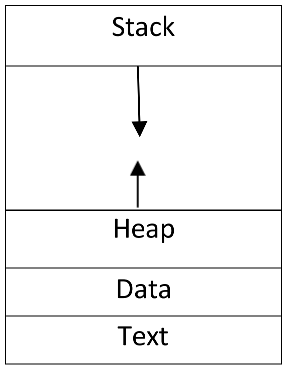
Q: What is the difference between process and program.

Process	Program
i) A process is the set of executable instruction, those are the machine code.	i) It is a set of instruction written in programming language.
ii) Process is dynamic in nature.	ii) Program is static in nature.
iii) Process is an active entity.	iii) Program is a passive entity.
iv) A process resides in main memory.	iv) A program resides in secondary storage.
v) A process is expressed in assemble language or machine level language.	v) A program is expressed through a programmable language.
vi) The time period limited.	vi) Span time period is unlimited.

Process in Memory:-

⇒ A process resides in memory through following section i.e.

1) Stack
2) Heap
3) Data
4) Text



⇒ Stack section contains local variable

⇒ Data section contains global variable

⇒ Text section contains code or instruction.

⇒ Heap section contains memory which will be dynamically allocated during runtime.

PCB/TCB (Program/stack control block):-

⇒ It keeps track all the information regarding a process. Following are the information.

Process no.
Process state
Program counter
Registers
Memory limits
List of open files

Process No.:-

⇒ It specifies a specific no. 2 each processes termed as PID (Process Identify)

⇒ No 2 process has the same PID.

Process state:-

⇒ It specifies the state of a process such as new, ready, running, waiting, terminated.

Program counter:-

⇒ It specifies the address of the next instruction to be executed for the process.

Registers:-

⇒ There are different and several no. of registers are present according to the system architecture. The registers such as index, accumulator, base register etc.

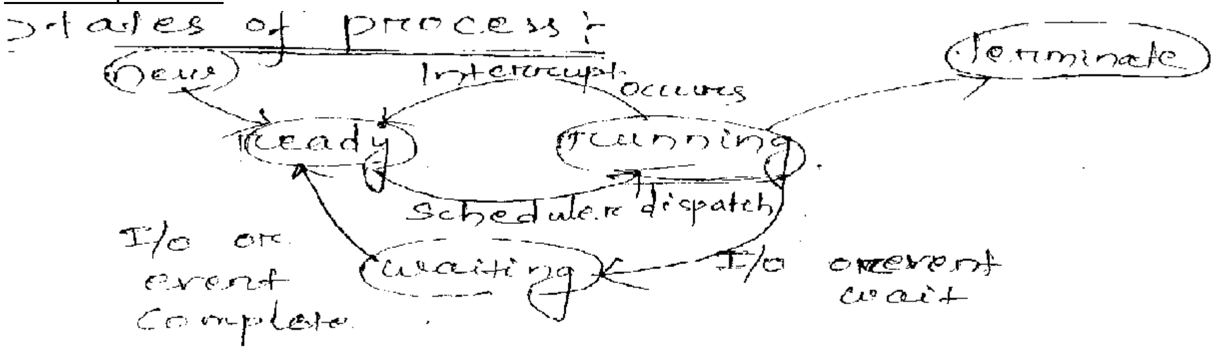
Memory limits:-

⇒ It specifies certain values for allocation of memory by a process.

list of open files:-

⇒ When a process is running along with several associated files are running. These files are named as open files.

States of process:-



- ⇒ A process carries out its execution throughout different activities.
- ⇒ Set of a process is defined in part of the current activity of that process.
- ⇒ The process goes through following 5 stages
  - i) New
  - ii) Ready
  - iii) Running
  - iv) Terminate
  - v) Waiting or blocked

i) <u>New state:-</u>

- ⇒ When the request is made by the user, the process is created.
- ⇒ The newly created process moves into a new statement.
- ⇒ The process resides in secondary memory through a queue named as job queue or job pool.

ii) <u>Ready state:-</u>

- ⇒ A process is said to be ready if it needs the CPU to execute.
- ⇒ Out of total newly created processes, specified processes are selected and copied to temporary memory or main memory.
- ⇒ In main memory they resides in a queue named as ready queue.

iii) <u>Running:-</u>

- ⇒ A process is said to be running if it moves from ready queue and starts execution using CPU.

iv) <u>Waiting state/ blocked state:-</u>

- ⇒ A process may move in to the waiting state due to the following reasons.
  - a) If a process needs an event to occur or an input or output device and the OS does not provide I/O device or event immediately, then the process moved into a waiting state.
  - b) If a higher priority process arrives at the CPU during the execution of an ongoing process, then the processor switches to the new process and current process enter into the waiting state.

v) <u>Terminated state:-</u>

- ⇒ After completion of execution the process moves into the terminated state by exiting the system.
- ⇒ Sometimes OS terminates the process due to the following reasons.
  - a) Exceeding the time limit
  - b) Input/output failure
  - c) Unavailability of memory
  - d) Protection error

<u>Scheduling:-</u>

- ⇒ The order of execution of process by the processor is termed as scheduling.
- ⇒ Following factors are associated with the scheduling.
  i.e.
  i) scheduling queue
  ii) Scheduler
  iii) Context Switch
  iv) Scheduling algorithm

i) <u>Scheduling queue:-</u>

⇒ There are 3 types of scheduling queues.
  a) Job queue
  b) Ready queue
  c) Device queue

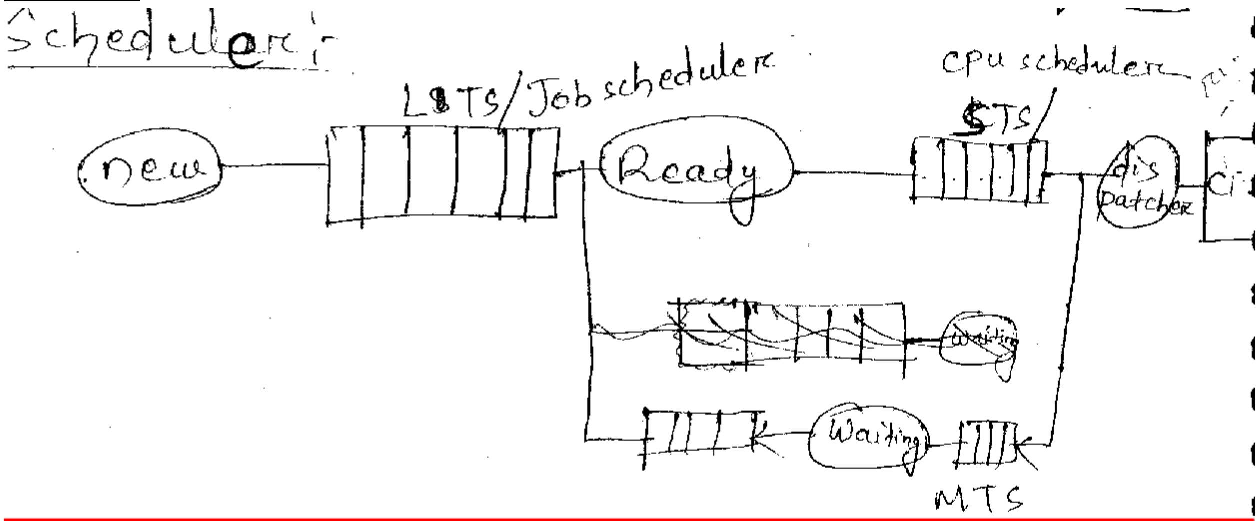
⇒ Scheduling is the basis of multiprogramming system.

⇒ In the new state the processes are resides in a queue named as job queue.

⇒ In the ready state processes are resides in a queue named as ready queue.

⇒ During waiting state a list of processes are waiting for a particular device, at that time they reside in a queue named as device queue.

ii) <u>Scheduler:-</u>



⇒ It is a component.

⇒ Selection of a process from one queue to other queue is carried out by a component termed as scheduler.

⇒ OS has 3 types of scheduler.
  i) Long term scheduler/ job scheduler
  ii) Middle term scheduler
  iii) Short term scheduler/ CPU scheduler.

i) <u>Long term scheduler:-</u>

⇒ It is known as job scheduler.

⇒ This scheduler selects the job from the job pool or job queue and loaded the jobs in the main memory.

ii) <u>Short term scheduler:-</u>

⇒ It is otherwise known as CPU scheduler.

⇒ This scheduler selects the job from main memory and gives the control of the processor to that process through the help of a module, known as dispatcher.

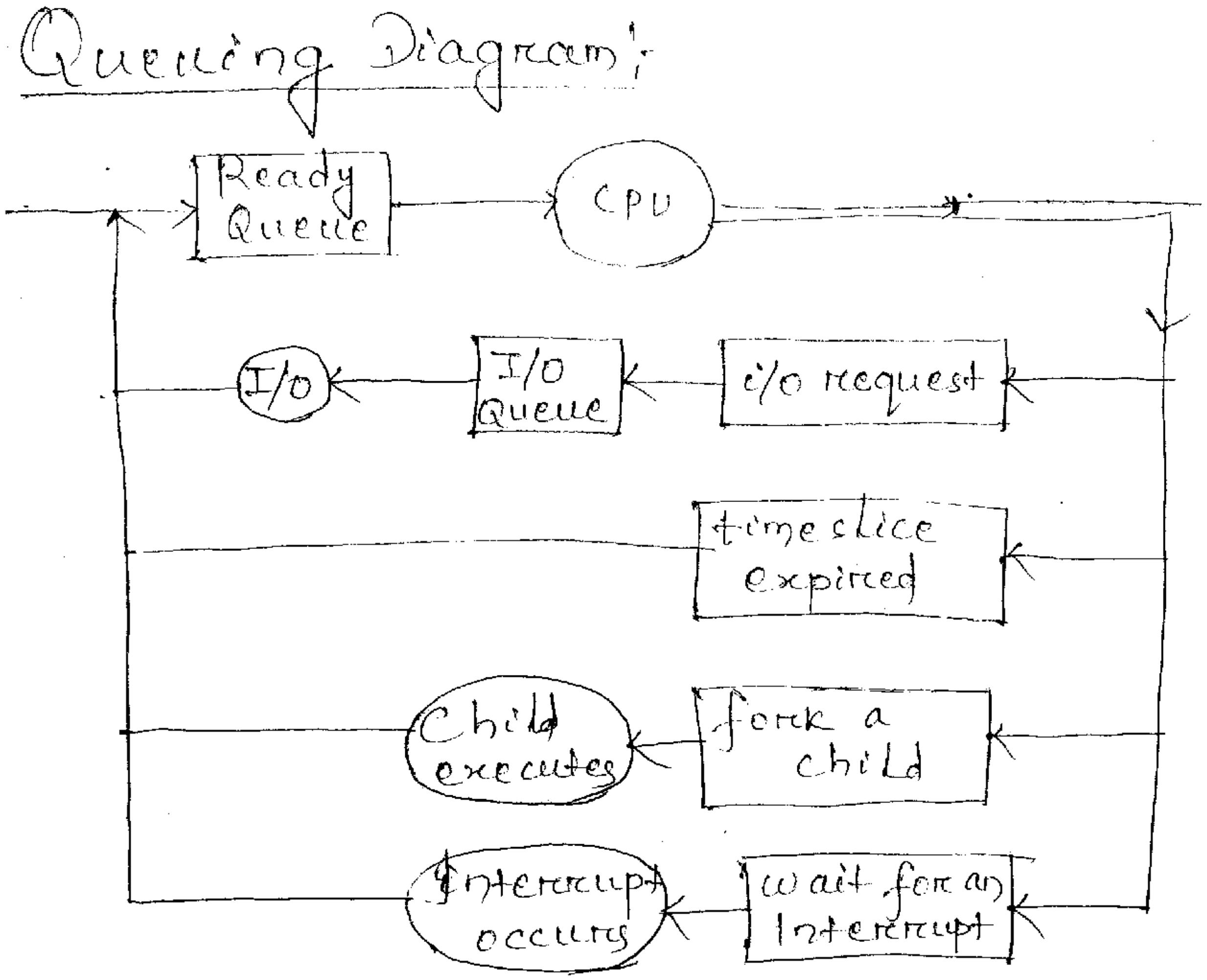
iii) Middle term scheduler:-

⇒ When a process moves from running state to waiting state and from waiting state to ready state, the transition of process occurs through a component named as middle term scheduler.

Q: What is the difference between job scheduler and CPU scheduler?

Job scheduler (long-term scheduler)	CPU Scheduler (Short term scheduler)
⇒ It is used to copy the jobs from job pool to load them into the main memory from execution. ⇒ It is otherwise known as long term scheduler. ⇒ It works in larger interval i.e. executed after a set of jobs are completed.	⇒ It copies the job from main memory to CPU for execution. ⇒ It is otherwise known as short term scheduler. ⇒ It works in smaller intervals. Here the intervals equals to the execution of a single job.

Queuing Diagram:-



- ⇒ Process scheduling is represented by a diagram named as queuing diagram.
- ⇒ Once a process is allocated to the CPU and starts execution following events occurs.
  1) A process could issue an input or output request and then placed in an I/O queue.
  2) The process could create a new sub process and wait for the termination of the sub process/ child process.
  3) A process could remove forceively from the CPU and again put back into the ready queue.
  4) A process could move into the ready queue after the expiring of the time slice/ time quantum or time slots.

- ⇒ State save specifies the current state of CPU whether in user made or kernel mode.
- ⇒ State restore is to resume the operations.
- ⇒ During the switching system does no useful work. So context switch time is overhead.
- ⇒ Because of context switching performance CPU is also reduced.
- ⇒ Context switching time is also dependent the hardware component of the system.

<u>Co-operating process:-</u>

- ⇒ In a multiprogramming system several processes are executing in an OS, are of 2 categories. i.e.
  - i) Dependent process/ cooperating proceed.
  - ii) Independent process.

<u>ii) Independent Process:-</u>

- ⇒ A process is said to be independent, if it is not affected by any other process executing in the system.
- ⇒ Independent processes are not sharing the data and information with other process.

i) <u>Dependent/ Co-operating process:-</u>

- ⇒ A process is said to be dependent, if it is affected by other processes executing in the system.
- ⇒ Co-operating processes can share the data and information with other processes.

<u>Advantages of co-operating process:-</u>

Following are the advantages.

- ⇒ Information sharing
- ⇒ Computation speedup
- ⇒ Modularity
- ⇒ Convenience

<u>Information sharing:-</u>

- ⇒ In multi programming system several users may need the same piece of information.
- ⇒ Co-operating environment allow concurrent access to these type of resources.

<u>Computation speedup:-</u>

- ⇒ To execute a task faster, it must break into subtasks. Each sub tasks executes parallel with each other, such a speed can be achieved only if the computer has multiprocessing elements.

<u>Modularity:-</u>

- ⇒ To construct a system in modular approach divide the system functions into no. of separate processes.

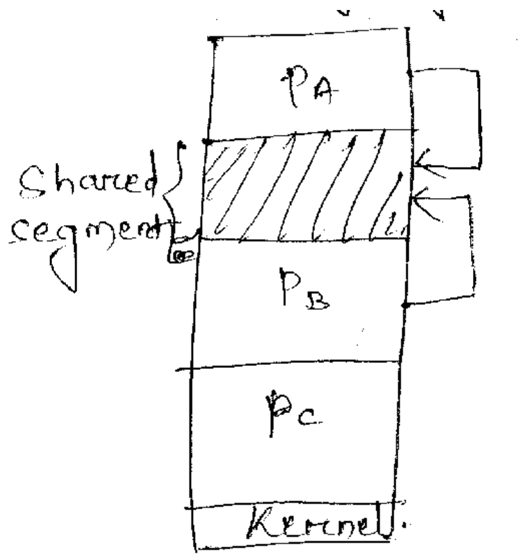
Convenience:-

⇒ A user can perform many task at one time.
  Ex. A user may be editing, printing and compiling in parallel.

IPC (Inter Process Communication:-

⇒ In a multiprocessing system several processes are executing by the system.
⇒ IPC is a mechanism for co-operating processes to allow the processes to exchange or share the data or information.
⇒ There are 2 fundamental model in IPC
  i) Shared memory system
  ii) Message passing system

Shared memory system:-



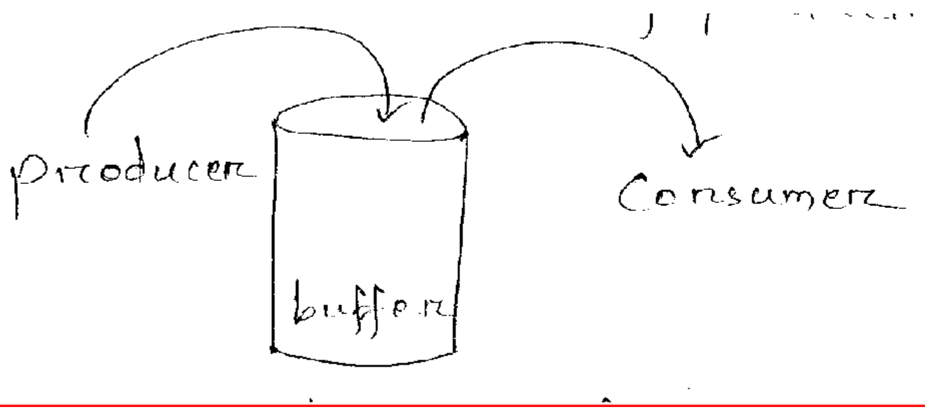
⇒ In shared memory system a region of memory is shared by the co-operating processes.
⇒ Then the processes can share the information by reading and writing data to the shared region.
⇒ Shared m.s. allows maximum speed and convenience of communication.
⇒ In shared memory system the shared region has been created through system calls.
⇒ A shared memory region resides in the address space of the process creating the share region.
⇒ In shared memory region all access are treated as routine memory access.
⇒ Other processes must attached there address space with the shared memory segment for communication.

Q: what is procedure – consumer problem?

A:

⇒ This problem arise in the shared memory system model.

⇒ This problem arises when there will be mismatch occur in rate of producing and consuming.



⇒ This problem are of two categories.

i) Bounded buffer problem

ii) Unbounded buffer problem .

i) Bounded buffer problem:-

⇒ In this problem the buffer size is fixed.

⇒ If the producer is faster than the consumer, then after some time the buffer is full and producer has to wait till the buffer becomes empty.

⇒ When the consuming rate is high than producing, then after some time buffer becomes empty and the consumer has to wait.

ii) Unbounded buffer problem:-

⇒ Here the buffer size can be vary.

⇒ In this case if the producer is faster than consumer, then the buffer is overloaded with data and if the consumer is faster than the producer then the consumer has to wait till the buffer becomes empty.

Message passing system:-

⇒ Message passing system is a mechanism to allow processes to communicate and synchronies their activities without sharing the address space.

⇒ Message passing mechanism/ concept is generally used in distributed environment.

⇒ 2 primitives or operations are there in message passing system.

i) Send () :- To send a message.

ii) Receive () :- To receive message.

⇒ Messages used for communication are of 2 types i.e.

⇒ Fixed sized msg

⇒ Variable sized msg

⇒ In message passing system communication occurs in 3 ways.

i) Direct / indirect comm..
ii) Synchronous/ asynchronous comm..
iii) Explicit/ implicit (automatic buffering)

i) <u>Direct Comm:-</u>

⇒ In direct comm.. the processes that want to communicate with each other must know each others name.

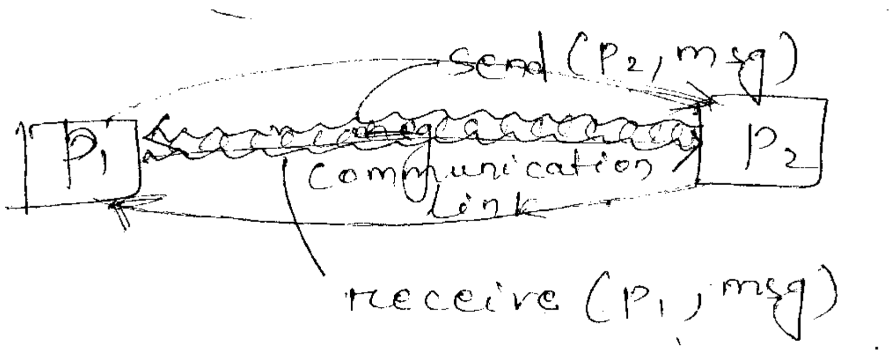
⇒ In direct comm. There are 2 types of addressing or naming .

a) Symmetric addressing
b) Asymmetric

a) <u>Symmetric:-</u>

⇒ In symmetric addressing send () operation will occur as send (receiving/ msg process name):

⇒ The receive operation occur as receive (sender process, msg name);

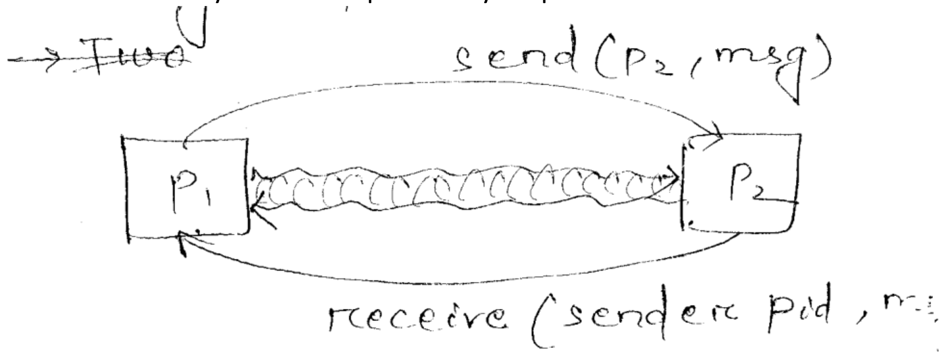


<u>Characteristics of communication link in direct communication:-</u>

⇒ It is established automatically between every pair of process that wants to communicate.
⇒ The link must be established with exactly two processes.
⇒ Between each pair of processes, there should be exactly one link.
⇒ The link may be unidirectional or bidirectional.

b) <u>Asymmetric:-</u>

⇒ In this addressing only sender names the receiver.
⇒ The receiver will identify the sender process by his process id



⇒ Two operation will be occur i.e.
i) Send (receiving process, msg name)
ii) Receive (sender process, msg)

2) <u>Indirect communication:-</u>

⇒ In indirect communication the messages are send and received by mailbox or port.
⇒ Mailbox is an abstract object in which the messages can be placed and removed.
⇒ Two processes can communicate only if they share the same mailbox.
⇒ 2 operation will occur as
Send (mailbox name, msg)
Receive (mailbox name, msg)
⇒ Indirect communication has the following properties
i) A link is established between every pair of process if they share a common mail box.
ii) A link may be unidirectional or bidirectional
iii) A link may be associated basically with two process (more than two process also possible)
iv) <u>Synchronization:-</u>
⇒ Two process can communicate among each other by performing the primitive operations i.e. send () and receive().
⇒ Msg passing may be occur in 4 ways i.e.
⇒ (i), (iii) <i> Blocking send
Asynchronies<ii> Non-blocking send Asynchronous
Synchronous <iii> Blocking receive
(ii), (iv)<iv> Non Blocking receive Synchronous
⇒ In blocking send, the sending process is block until the msg has been received by receiving process. It means sender is waiting.
⇒ In non-blocking send, the sending process sends the msg without waiting for an acknowledgement for on the receiving process.
⇒ Blocking receive, the receiver blocks until a msg is available.
⇒ Non-blocking receive, the receiver receives either valid msgs or null msgs.

<u>Buffering:-</u>

⇒ A link capacity determines a no. of msgs can reside in temporary queue.
⇒ A link can be assume as a queue like structure.
⇒ This link can be implemented in 3 ways.
i> Zero capacity (no buffing)
ii> Bounded capacity
iii> Unbounded capacity.
⇒ In zero capacity the queue maximum length is zero, the link can't hold nay msgs waiting in it.
⇒ In bounded capacity, the queue has finite length 'n'. hence at most 'n' msgs can reside the link.
⇒ In unbounded capacity any no. of msgs can be reside.

CPU Scheduling:-

- ⇒ This algorithm determines the order of allocation of CPU to the processes.
- ⇒ The selection of process is carried out by short term scheduler from the ready queue for the execution.

Classification of CPU scheduling algorithm

- ⇒ There are 2 types of scheduling algo.
  i.e.
  i> Preemptive scheduling algo
  Non preemptive scheduling algo.

ii> Non-Preemptive:-

- ⇒ In this scheduling once the CPU is assigned to a process, the processor doesn't relase until the completion of that process.
- ⇒ The processor will assigned to another job only after the completion of the previous job.

i> Preemptive:-

- ⇒ In preemptive scheduling the CPU can release the processes even in the middle of the executing if request is made by another process.

Measuring factor for CPU scheduling algo:-

<i> Throughput:-the performance of the CPU is measured by this unit.

- ⇒ Throughput means the no. of jobs are completed per unit time.

<ii> Brust time:-the time required by the processor to execute a job is termed as burst time.

- ⇒ It generally expressed in millisecond (MS).

<iii> Waiting Time:-It is the sum of the periods spent by a process in the ready queue.

- ⇒ Waiting time = starting time – Arrival time
- ⇒ Least waiting time shows efficiency of an algorithm.

<iv> Turn around time:-The time interval between summation of the process and time of completion, is known as (TAT).

- ⇒ TAT = Finish time – Arrival time

Types of CPU scheduling algorithm:-

i) FCFS – First come first serve
ii) SJF – Shortest job first
iii) RR – Round Robin
iv) Multilevel Queue scheduling

v) SRTF – shortest remaining time first
vi) Priority scheduling

Steps for solving the algorithm:-

1) Draw the gantt chart
2) Calculate the turn around time for individual process.
3) Calculate the averages turnaround time.
4) Calculate the waiting time for individual process.
5) Calculate the average waiting time.
   Gantt Chart:- The order of execution of processes and their time limit is maintained through a chart named as gantt chart.

1) FCFS:-The process with enters the ready queue first will be solved by the CPU first.
   ⇒ It is a non-preemptive type scheduling algorithm.

Q:-

Process	Burst time
P1	24
P2	3
P3	3

Assume arrival time = 0 ms. Calculate the avg. turn around time and average waiting time.

A:-Step-I:- Draw the gantt Chart

P1	P2	P3

0 24 27 30

Step-2:- TAT for P1 = 24-0=24 ms
P2=27-0=27 ms
P3=30-0=30 ms

Step-3:-Avg. TAT = $\frac{24+27+30}{3}$
= 27 ms

Step-4:- Waiting time for P1=0-0=0 ms
P2=24-0=24
P3=24-0=27

Step-5:-Avg. waiting time = $\frac{0+24+27}{3}$ = 17 ms

A: Calculate the avg. TAT and avg. waiting time for the following.

Process	Burst time
P1	5
P2	24
P3	16
P4	10
P5	3

Advantage of FCFS:-
Time algorithm is easy to implement because once a process enters the ready queue, it can't execute the task continuously.

Disadvantages:-

This algorithm is not applicable for time sharing system.

⇒ In this algorithm the average waiting time is little bit more which decreases the CPU performance.

Priority algo:-

In this type of scheduling a numerical value is assigned to individual process is termed as priority. The priority value is assigned by the OS.

⇒ The CPU will allocate the process which having higher priority value.

⇒ The priority can be mentioned in two ways.
  i) Lower no = higher priority
  ii) Higher no = higher priority

⇒ Priority can be defined by the OS in two
  i) Internal priority
  ii) External priority.

⇒ Internal priority can be defined by considering the time limit, memory requirement no. of open files etc.

⇒ External priority can be defined by considering the resource requirement, burst time etc.

Q: calculate the average TAT and Average waiting time for the following.

Process	Burst time	Priority
P1	10	3
P2	5	2
P3	2	1

A:-Step-I:- Assume higher no = higher priority

P1	P2	P3

0 10 15 7

Step-2:- TAT for P1 = 10-0=10

P2=15-0=15

P3=17-0=17

Avg. TAT = $\frac{10+15+17}{3}$ = 14

Step-4:-W.T. for P1=0-0=0
P2=10-0=10
P3=15-0=15

Step-5:- Waiting time for P1=0-0=0 ms
P2=24-0=24
P3=24-0=27

Step-5:-Avg. waiting time = $\frac{10+15+0}{3}$ = 8.3

Q: Calculate the TAT and Average W.T. for the following.

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

Arrival time = 0
Lower no = higher priority

A:-

```
      | P2 | P5    | P1    | P3     | P4    |
    0        1       6      16       18   19
```

Advantage:-

⇒ This is easy to implement

Disadvantages:-

⇒ A problem will be arise in priority scheduling named as starvation.

⇒ It is otherwise known as indefinite blocking. A priority scheduling can leave some low priority process in the device queue for a longer period of time. Because every time some higher priority process arrives which always be in executing.

Note:-

Aging:-

To overcome the problem of starvation for low priority process a technique comes named as ageing.

⇒ Aging is a technique of gradually increasing the priority of processes those are waiting in the device queue for a longer period of time.

SJF (Shortest Job First)

⇒ In this scheduling algorithm, the P which having the less burst time will be consist as shortest job.

⇒ The shortest job which will be executed fire by the processor having less burst time.

⇒ This algorithm is a non primitive type scheduling (some times it can be also premtive type).

⇒ If 2 process having the same burst time then they will be executed in FCFS manner.

Q:- Calculate the average TAT and Waiting time by using SJF algorithm from the flowing information

Process	Burst time
P1	6
P2	8
P3	7
P4	3
	24

A:- Gantt Chart"=

P1	P2	P3	P4	X

0 6 14 21 24

TAT for P1 = 6-0=6 ms
P2 = 14-0= 14 ms
P3 = 21-0=21 ms
P4 = 24-0=24 ms
Avg TAT = $\frac{6+14+21+24}{4}$ = 16.25

Waiting time for S.T. - A.T.
P1=0-0=0
P2=6-0=6
P3=14-0=14
P4=21-0=21
Avg. waiting time = $\frac{0+6+14+21}{4}$ = 10.25

P4	P1	P3	P2

0 3 9 16 24

⇒ The SJF algorithm is considered as an optimal algorithm, because this algo is having minimum average waiting.

⇒ This algorithm is also otherwise known as CPU burst algorithm.

<u>RR –Round Robin algorithm:-</u>

⇒ This scheduling algo is basically used in the time sharing system.

⇒ It is a preemptive type scheduling.

⇒ In this scheduling a time (Same time slot) slot is given to individual processes for the execution. This is termed as time quantum or time slice.

⇒ The time quantum is generally ranged from 10-100 ms considering the burst time of the processes.

⇒ In this algorithm the ready queue is a circular queue.

⇒ The CPU scheduler picks the job from the ready queue and according to the time slot it will be executed by the processor and after the finishing of the time slot the processor dispatch the process.

⇒ The performance of the RR depends on the size of the time quantum.

⇒ If the time quantum is large, then the RR scheduling may be similar with FCFS. (RR scheduling is an extension of FCFS algorithm).

Q:- Calculate the Average TAT and Average W.T. by using RR algorithm.

Process	Burst Tiem	arrival time = 0
P1	24	time slice = 3 ms
P2	3	
P3	3	0-3

Gantt chart

P1	P2	P3	P1	P1	P1	P1	P1	P1	P1

0 3 6 9 12 15 18 21 24 27 30

TAT for P1 = 30-0=30
P2=6-0=6
P3=9-0=9
Avg. Tat = $\frac{30+6+9}{3} = \frac{45}{3} = 15$
WT for P1 = (0-0)+(9-3) = 6
P2 = 3-0=3
P3 = 6-0=6
Avg WT = $\frac{6+3+6}{3} = \frac{15}{3} = 5$

Q: process burst time

P1	30 25 20	At = 0
P2	6 x	time quntum = 5 ms
P3	9 3	

Gantt chart

P1	P2	P3	P1	P2	P3	P1	P1	P1	P1

0 5 10 15 20 21 24 29 34 39 44

TAT for P1=(44-0)=44

P2 = 21-0=21

P3 = 24-0=24

Avg. TAT = $\frac{44+21+24}{3} = \frac{89}{3} = 29.66$
WT for P1-(0-0)+(15-5)+(24-20)
=10+4
=14
P2=(5-0)+(20-10)
=15+10
=15
P3=(10-0)+(21-15)
=10+6
=16
Avg. WT = $\frac{14+15+16}{3} = 15$

Q: Process Burst Time

P1	$2^{0}0$ 16 12.8x	
P2	12 8 4 x	A-T=0
P3	18 14 10 6 2	Time quantum – 4 ms

P4	16. 12.8 4
P5	4 x

	P1	P2	P3	P4	P5	P1	P2	P3	P4	P1	P2	P3	P4	P1	P3	P4	P1	P3
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	70

TAT for P1 = (68-0) = 68
P2 = (44-0) = 44
P3 = (70-0)=70
P4 = (64-0)=64
P5=(20-0)=20

Avg. TAT = $\frac{68+44+70+64+20}{5}$

=53.2 ms

WT for P1=(0-0)+(20-4)+(36-24)+(52-40)+(69-56)
=0+16+12+12+8
= 48
P2 = (4-0)+ (24-8)+(40-28)
= 4+16+12
=32
P3 = (8-0)+(28-12)+(44-32)+(44-32)+(52-48)+(68-60)
=8+16+12+18+8
= 52
P4 = (12-0)+(32-16)+(48-36)+(60-52)
= 12+16+12+8
= 48
P5 = 16-0 = 16

Avg. WT = ~~52+32+56+46+16~~
5
= ~~39.2 ms~~

= $\frac{196}{5}$ = 39.2 ms

<u>Multilevel queue scheduling:-</u>

⇒ This scheduling occurs by categorizing the processes.

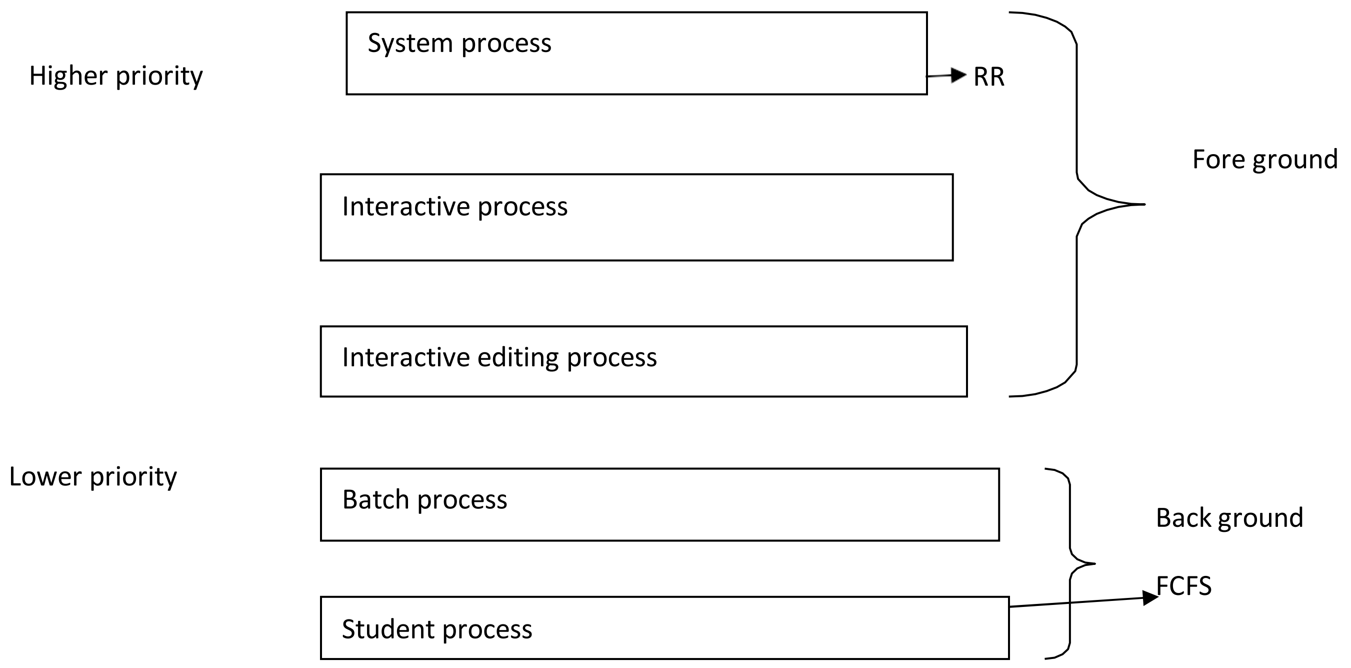
⇒ It classifies the processes into two categories.
i.e.
1) foreground process (Interacting process)
2) Background process (Batch Process)

⇒ This scheduling partition the ready queue into several separate queue.

⇒ The process are assigned to queue according to their category.

⇒ Each queue has its own scheduling algo.

⇒ Generally foreground process – RR Scheduling background process – FCFS scheduling
⇒ Scheduling among the queues done through priority scheduling.
⇒ The foreground process having the high priority and background process having the low priority.



Q: - Calculate TAT and Average WT by using priority scheduling.

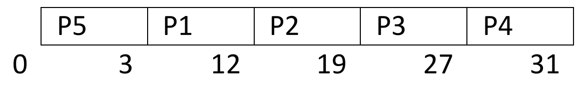
Process	burst time
P1	9
P3	7 x
P3	8 x
P4	4 x
P5	3

Time quantum = 3 ms

Priority P1	5
P2	7
P3	9
P4	10
P5	1

lower no. having the higher priority.

A: Gantt Chart:



TAT for P5=3-0=3

P1 = 12-0=12

P2 = 19-0=19

P3 = 27-0 = 27

P4 = 31-0=31

Avg. TAT = $\frac{3+12+19+27+31}{5}$ = 18.4 ms

WT for P5 = 0-0=0

P1=3-0=3

P2 = 12-0=12

P3 = 19-0=19

P4 = 27-0=27

Avg. WT = $\frac{0+3+12+19+27}{5}$

=12.2 ms

Time slice = 3 ms

	P1	P2	P3	P4	P5	P1	P2	P3	P4	P1	P2	P3
0	3	6	9	12	15	18	21	24	25	28	29	31

TAT for P1=28-0=28

P2=29-0=29

P3=31-0=31

P3=31-0=31

P4=25-0=25

P5=15-0=15

Avg. TAT = $\frac{28+29+31+25+15}{5}$

=25.6 ms

WT for P1 = (0-0)+(15-3)+(25-18)

= 12+7 = 19

P2 = (3-0)+(18-6)+(28-21)

=3+12+7

= 22 ms

P3 = (6-0)+(21-9)+(29-24)

= 6+12+15

23

P4 = (9-0)+24-12)

= 9+12=21

P5 = (12-0)=12

Avg. WT = $\frac{19+22+23+21+12}{5}$

= 19.4 ms

SEMAPHORE

- ⇒ It is a synchronization tool, denoted as 'S' which is an integer variable whose value can be changed and altered.
- ⇒ Its value indicates the status of shared resources, a process which needs the resource, will check the semaphore for determining the status fo the resource (available/unavailable).
- ⇒ If the value of the semaphore variable can be changed by two operations.
  - i) Wait (P)
  - ii) Signal (v)

Wait of S:-

- ⇒ Wait of S an atomic operation that waits for semaphore of the value of S is > 0, then decrement it by 1 or else wait on S.

```
If (S>0)        wait(s)
Then S          {
S=S-1           while (S≤0)
Else            {
Waitons         s---;
                }
                }
```

Signal of 'S':-

- ⇒ Signal of S is an atomic operation that increments 'S' by 1.

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

- ⇒ When one process modifies the value of the S, no other process can simultaneously modify the same S value.

  S are of 2 types:-
  - i) Binary S
  - ii) Counting S

i) Binary S:-

- ⇒ Binary S can take 2 values.
- ⇒ These values ranges between 0 to 1.
- ⇒ These S are used to acquire the lock.
- ⇒ The lock used by the binary S is termed as MUTEX lock.

```
do
{
Waiting(mutex);
// c.s
Signal(mutex);
//reminder section
{while(true);
```

ii) <u>Counting S:-</u>

⇒ The counting S is applicable for multiple instances of resource type.

⇒ Each process that wants to use the resource performs wait operation on the S.

⇒ When a process release the resource, it perform signal operation.

⇒ When the count of the S goes to zero all the resources are being used.

⇒ When the count > 0, he resources is unblocks

<u>Disadvantage in S:-</u>

⇒ The main disadvantage in S is busy waiting.

⇒ While a process is executing in critical section the other processes are requesting continuously looping, in the entry section by making the process busy wait.

⇒ Busy waiting cause wastage of CPU time, CPU cycle, m/m space etc. during busy waiting process spins in the entry section to get the lock. So this type of semaphore status is termed as spine lock.

<u>To overcome the busy waiting implementation of semaphore:-</u>

⇒ To overcome the problem occurred in busy waiting the wait () and signal() operation are modified.

⇒ When the process executes the wait operation, rather than staying in waiting condition, the process blocked itself. During this blocking it is associated with a waiting queue.

⇒ Similarly when a signal operation is occurred a process is removed from the queue and executes.

⇒ Each entry is a waiting queue has 2 data values.

i) Value of integer type.

ii) Pointer to the next record in the list

During this 2 operations will be performed

i) Block () →modification of wait ()

ii) Wake up ()→modification of signal()

<u>Questions:-</u>

1. Draw the states of process.
2. Draw the PCB.
3. What are the different types of scheduler?

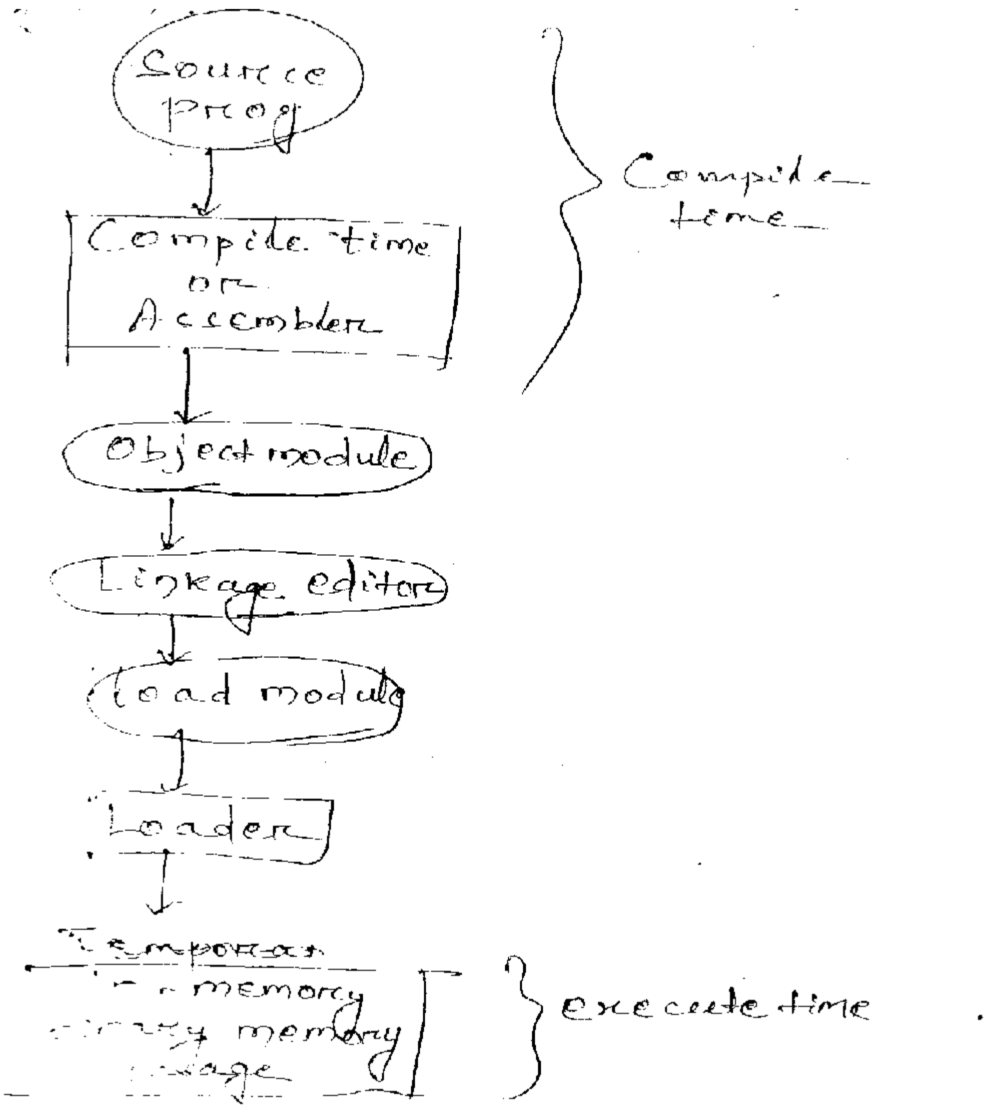
Chapter - 3

MEMORY MANAGEMENT

Resident Monitor:-

A small programme is called resident monitor, that, was created to transfer control from one job to another job.

Memory is a large array of words or bytes. It is a storage area or location to store the data and information being processed by the system.



⇒ An instruction is executed by covering two cycle.

i) Fetch cycle.
ii) Execution cycle.

⇒ The following are the steps for the execution of an instruction.

i) Fetch the instruction from the memory.
ii) Decode the instruction.
iii) Execute the instruction.
iv) Stores the result in memory.

⇒ The entire execution phase of a program divides into 3 parts.

i) Compilation phase
ii) Loading phase
iii) Run phase.

⇒ Compilation phase occurs by a component known as compiler which converts the source code to the object code or machine level code.

⇒ During the load phase the converted object codes is loaded into the memory by a component known as loader through a load module.

⇒ During the execution phase the instructions are being executed in the processor and returns the result to main memory.

⇒ Basically there are 2 kinds of memory.

i) Main memory
ii) Secondary memory.

⇒ Main memory is volatile in nature in which the instructions are stored temporarily.

⇒ Secondary memory is nonvolatile in nature in which instructions are store permanently.

Objective of M/M mgt. scheme:-

To give the protection of different processes from each other.

This scheme decides allocation strategy for different processes in memory.

This scheme gives idea regarding address, binding, local address, physical address, address translation etc.

It also implements some rules to avoid the wastage of memory space through paging.

It also implements the method to deallocate the memory space.

⇒ Memory will be divided into 2 parts

i) Operating system
ii) User space.

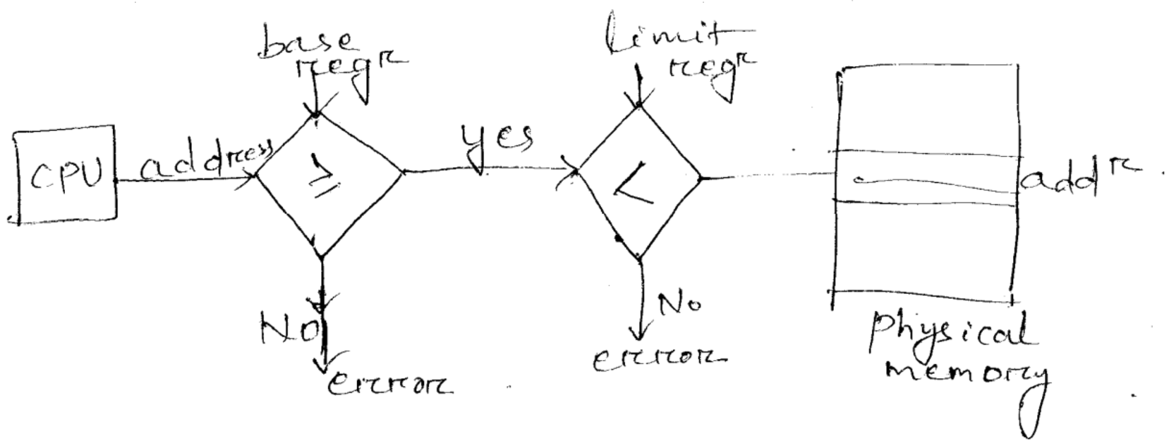
⇒ User space will be used to keep multiple no of processes in the memory.

⇒ When a process resides in memory to give the protection it will take the help of two registers.

i) Base register
ii) Limit register

⇒ Base register holds the smallest physical memory address (the memory location from where the process starts).

⇒ The limit register holds the size of the range.



Binding is a mapping from one address space to another address space.

To deal with an address binding there will be 2 types of memory or address space. I) logical, ii) Physical

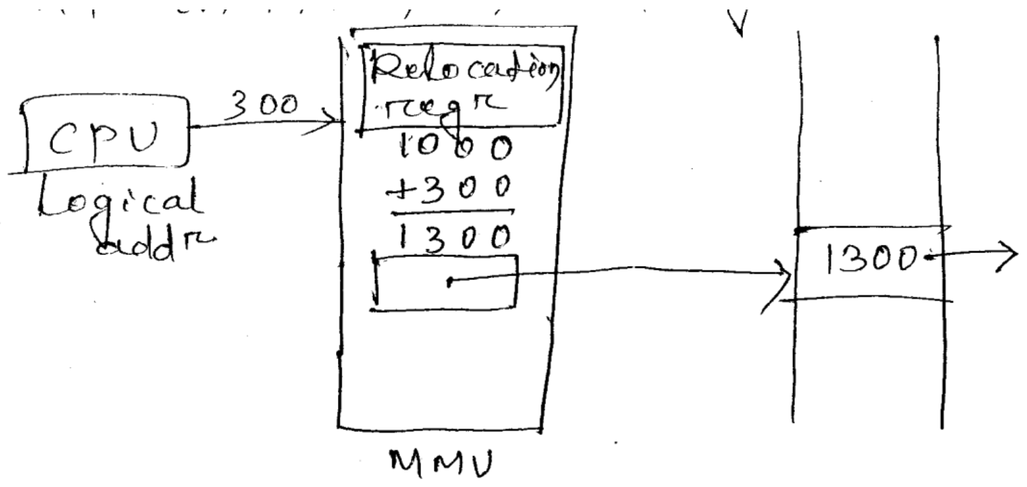
Logical address will be generated by the processor or CPU.

The set of all logical address is known as logical address space.

Physical address is the address refers to main memory location. It will be generated through a h/w unit named as memory mgt. unit (MMU) through a register called as memory address reg. (MAR).

The set of all physical address is known as physical address space.

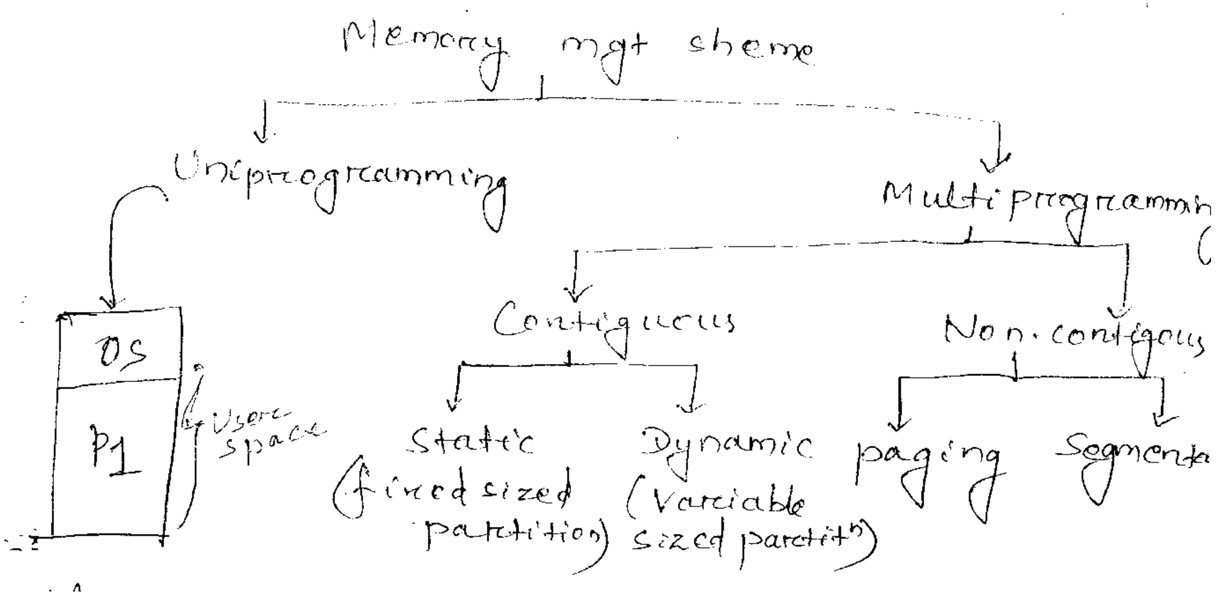
A register named as relocation register which relocate the address in the main memory.



Relocation means shifting the user program from one memory location to another memory location.

The relocation mechanism can be done in 2 way in relation of dynamic relation.

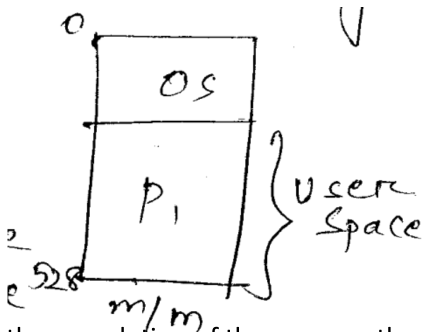
The logical address is related to an user mode open and physical address is related to a kernel mode operation.



Uniprogramming memory allocation (single partition):-

In uniprogramming system memory will be divided in to 2 parts.

i) OS
ii) User space



Each time only one process can be loaded in the user space after the completion of the process, the next process can be loaded.

Disadvantages:-

In an uniprogramming system a huge amount of wastage of memory space.

Multiprogramming memory allocation (Multi partition):-

⇒ In a multiprogramming system the user space will be divided into no. of blocks known as partition.

⇒ In a multiprogramming system the memory allocation may be either contiguous or non contiguous type.

⇒ The static contiguous memory allocation can be done before the execution of the process. This can be are of 2 types.

i) Fixed equal size partition.
ii) Fixed unequal size partition.

i) Fixed sized:-

⇒ In fixed sized equal partition the user space will be divided into equal size.

⇒ Here the 50 k user space has been divided into 5 nos. of partition having each size 10k.

Disadvantages:- in this type of allocation memory space may be wasted in each individual partition.

ii) Fixed unequal:-

⇒ In this partition the user space is divided into no. of unequal sized partition.

⇒ In this the user space has been divided into 5 partition are of unequal size.

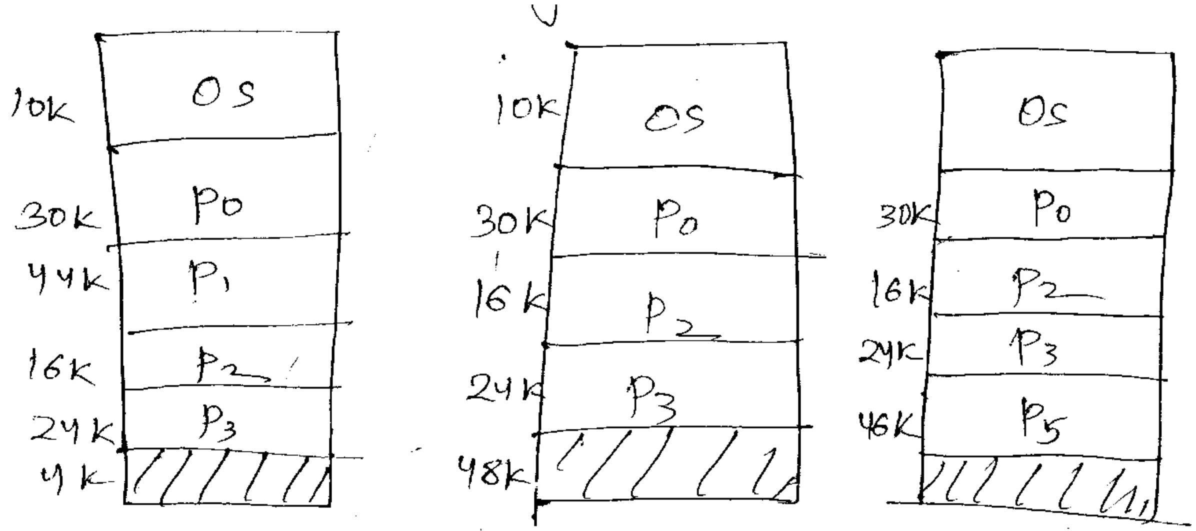
⇒ In this allocation scheme the wastage of memory space will be less in comparison to fixed equal sized partition. But still wastage of memory space is there.

Compaction:-

⇒ In external fragmentation several holes have been generated in each partition and the memory space present in the holes are not contiguous in nature. So it is unable to accommodate one of the process.

⇒ This problem can be overcome by a mechanism known as compaction.

⇒ Compaction means to move the processes in memory in such a way that the scattered piece of unused memory can be placed together so that any other process requiring the contiguous memory can use it.

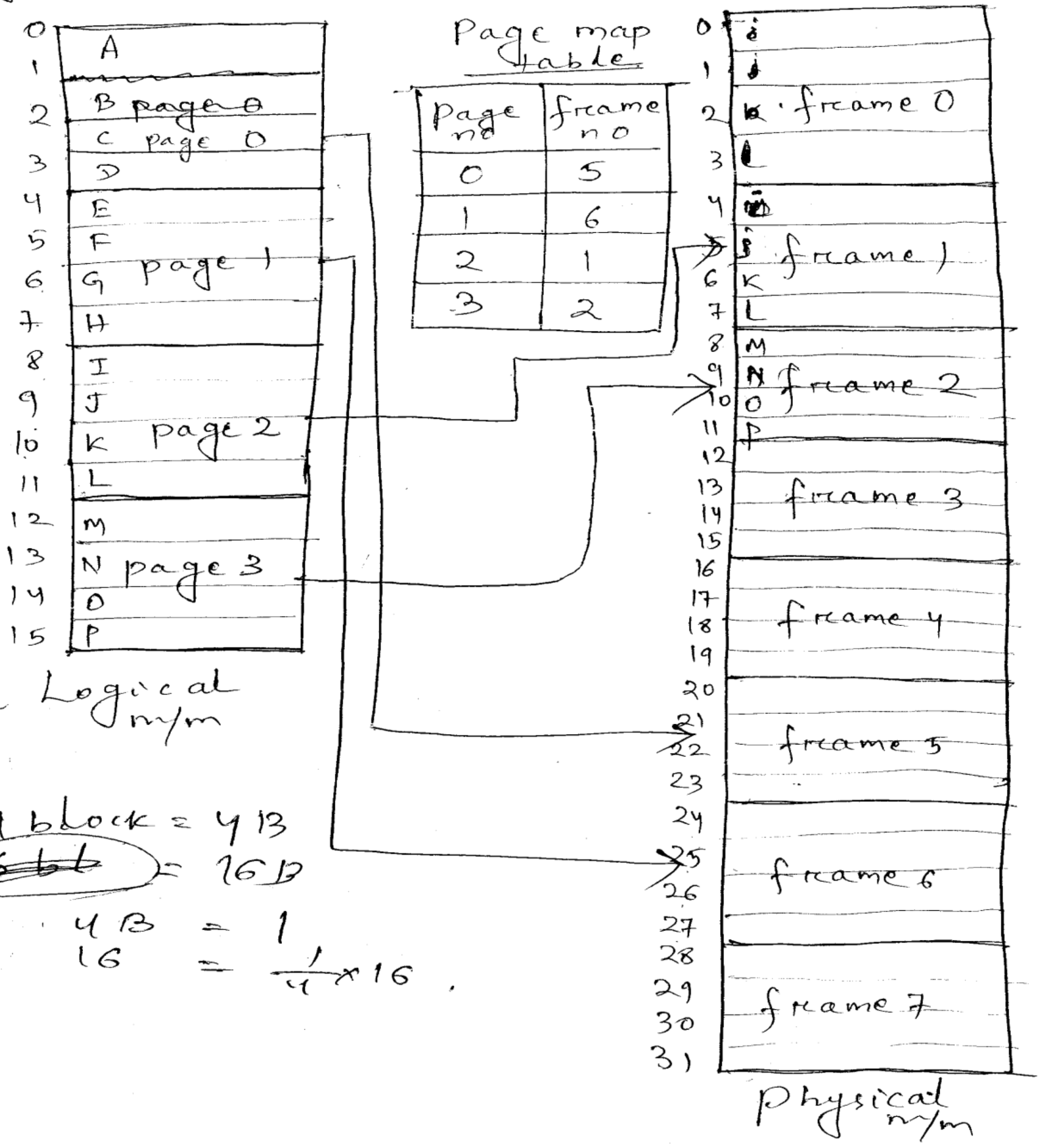


⇒ In the 1st diagram P1, P1, P2, P3 are executing.

⇒ In 2nd diagram P1 terminates. Then P2 and P3 will move upward. So the 44k of P1 is being compacted in the end of the memory with the 4K. so total unused space is now 48 k.

⇒ In 3rd diagram another process P5 arises whose process size is 46k. the availability of memory space is 48k. so it can allotted to the end partition of the memory.

⇒ A physical m/m of 32 bytes page size is 4B. Draw the dig.



Page no	frame no
0	5
1	6
2	1
3	2

1 block = 4B

~~16 bt~~ = 16B

4B = 1

16 = $\frac{1}{4} \times 16$.

Q: Translate the logical address 13 to physical address.

No. of pages = size of logical memory/size of each page

$2^3 = 2^n/2^2$

n = 5 bit

so 5 bit logical address is required

13=01101

Binary decimal

01 – 1 = offset

011 = 03 = page no

Physical address = (pageframe x page size) + offset

= (2x4)+1

=8+1 = 9

N = 13 will be related with 9.

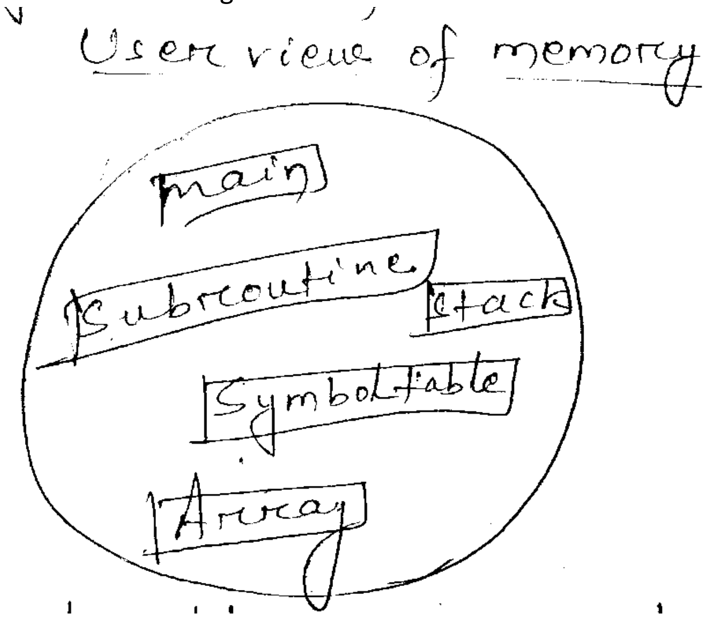
Paging:-

⇒ Paging is a memory mgt scheme in which the memory allocation is done in a contiguous manner.
⇒ In paging scheme there is no external fragmentation.
⇒ in this a process is divided into no. of pages
⇒ Logical memory or address is divided into fixed sized and blocks called as page.
⇒ Physical memory is divided into fixed sized blocks called as frames.
⇒ The objective of the paging technique is that whenever a process is executed, how to place the pages of process in the available of free frames in physical memory.
⇒ The logical address is translated/mapped into physical address by using a table named as page table which is content by the memory mgt. unit.
⇒ Every address generated by the CPU (logical address) has two parts i.e. page no. (P), page offset (d).
⇒ The page size should be same with the frame size.
⇒ The page size is defined by the hardware normally for easy mapping the page size is always a power of 2.
⇒ If the logical address's space size is 2^m, the page size equals to 2^n. Then the higher order (M-n) units of logical address is the page no. The lower order n units is the page offset.

Segmentation:-

⇒ It is a memory mgt. scheme that supports user view of memory.
⇒ User view of memory specifies how a uses program/process resides in memory.
⇒ A process/ program has been divided into of components known as segments.
⇒ Ex: AC compiler has following segment such as
   i) Main,
   ii) Subroutine
   iii) Stack
   iv) Symbol table
   v) Array etc.

⇒ The segments are of variable length



⇒ A logical address space is divided into no. of segments. Each segments has its name and length.

⇒ Each address generated through 2 quantities.

i) Segment name

ii) Segment offset

⇒ Logical address consist of 2 tuples.

i) Segment no. (S)

ii)

⇒ Log. Address is mapped into physical address through a table named as segmentation tale.

⇒ In segment table there are 3 entries.

i) Seg. no. – seg. No. specifies index to seg. Table

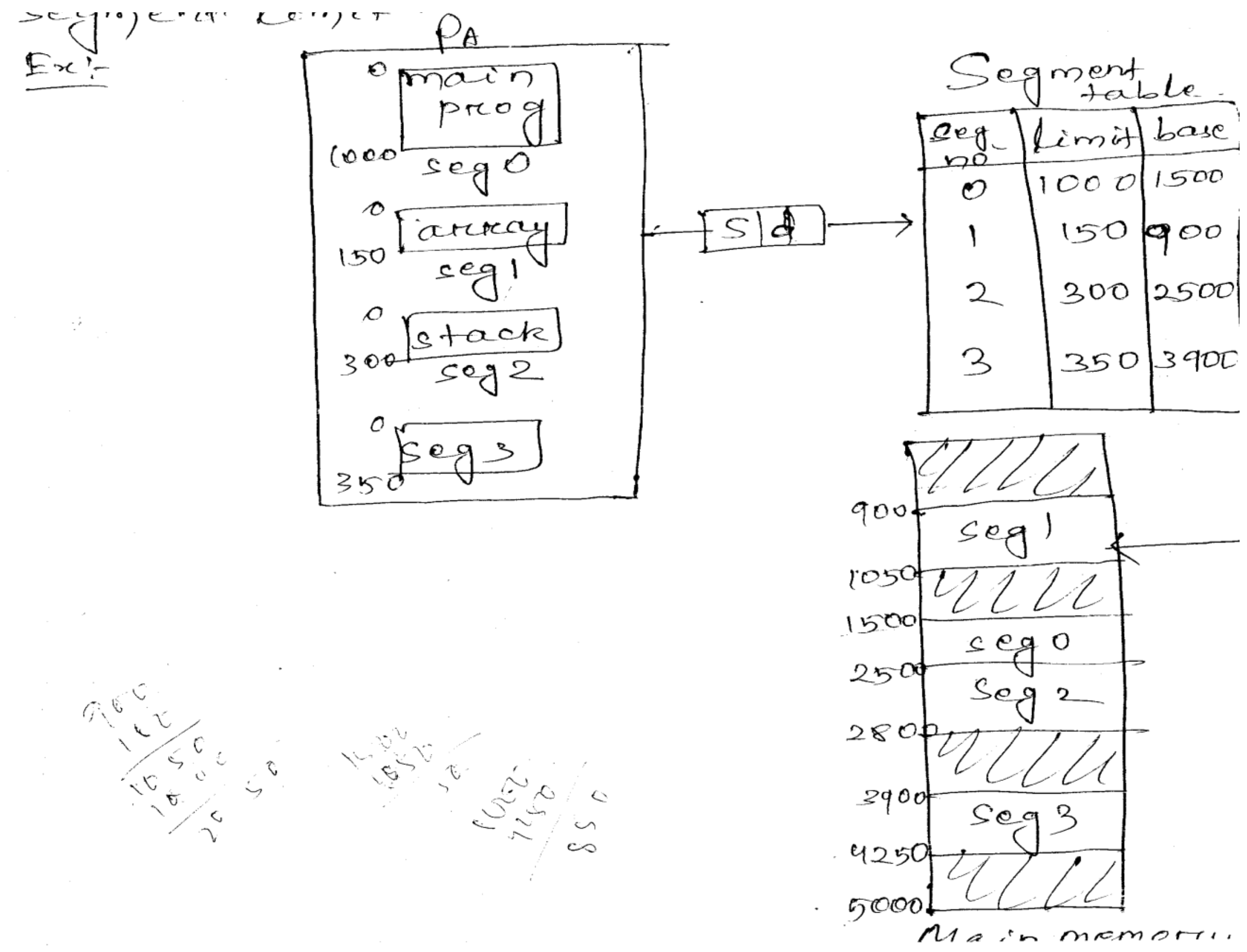
ii) Segment base – beginning address of seg. In phy. m/r

iii) Seg. Limit – specifies length of seg.

<u>Offset (d):-</u>

It must be within the range of 0 and segment limit.

Ex:-



Difference between paging and segmentation paging:-

Paging	Segmentation
⇒ The main memory is divided into no. of blocks i.e. known as frame to hold the pages.	⇒ The m/m is divided into no. of blocks to hold the seg.
⇒ Log. Memory is divided into no. of same sized blocks are known as pages.	⇒ Log. Memory is divided into no. of segments.
⇒ In paging internal fragmentation may occur.	⇒ In segmentation the external fragmentation may occur.
⇒ Address translation/ mapping occurs in paging through a table named as page table.	⇒ Address translation/map occurs through a tab named as segmentation table.
⇒ In page table there are only 2 entries. i.e. page no. and frame no.	⇒ In segmentation table there are 3 entries i.e. seg. No., seg. Limit, seg. Base.
⇒ In paging logical address is generated through two values i.e. page no., page offset.	⇒ Logical address is generate through 2 values i.e. seg. No and seg. Offset.
⇒ In paging the pages are of same size (normally)	⇒ The seg. Are of variable sized specify through
⇒ In paging the page size and frame size are same.	⇒ In segmentation the seg. Size and frame size are different.
⇒ Paging doesn't support the user view of memory.	⇒ Segmentation based on user view of memory.

Virtual memory:-

⇒ Virtual memory is a tech. which allows the execution of process even if the size of the log. M/m>p size of phy. m/m.

⇒ Vir. m/m mechanism cab be implemented through a concept known as swapping.

⇒ Advantage of virtual memory is the efficient memory unilisation.

⇒ It also makes the task of the programmer easier because not considering the size of phy. m/m.

Demand paging:-

⇒ It is application of virtual memory.

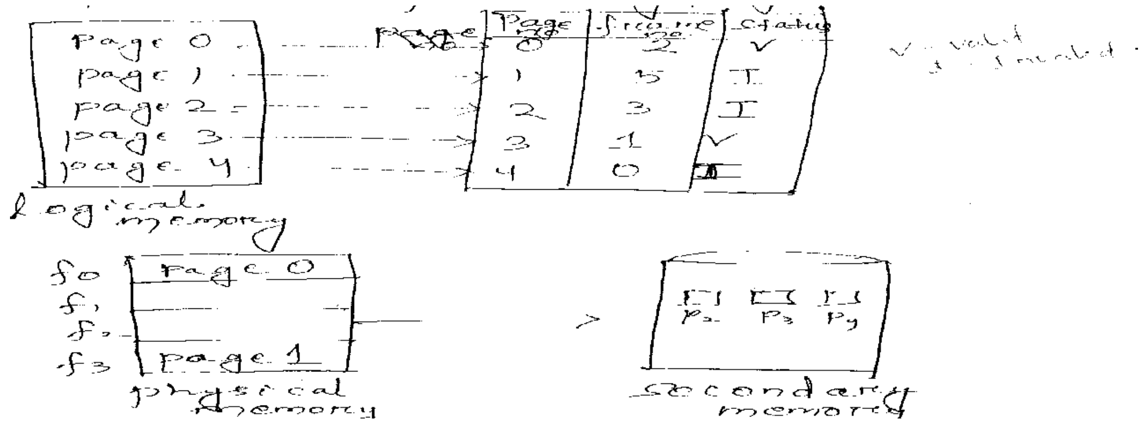
⇒ This is a combination of paging and swapping.

⇒ The criteria of the demand paging scheme is a page is not loaded to the main memory from the secondary memory until it is needed. So a page is loaded in the memory on demand.

Page fault:-

⇒ When the processor needs to execute a particular page and it is not available in the main memory. This situation is known as page fault.

⇒ When the page fault occurs the requested page should be loaded in the main memory in replacement of an existing page.



⇒ The selection of the page to be replaced is named as victim page.

⇒ The victim page will be replaced for the requested page.

⇒ The victim page is swapped out to the secondary memory and the requested page will be swapped into the main memory.

⇒ The status of the page (exist or not in the memory) is determined by a bit name as status bit which has 2 values.

i) Valid (V) – if requested page is present in main memory.

ii) Invalid (I) – If requested page is not present in the main memory.

⇒ To overcome the page fault occurrence some algorithms are implemented named as page replacement algorithm.

Page replacement algorithm:-

Following are the replacement algorithm. These algorithms will select the victim page which are to be replaced in against of the requested page i.e. it selects a page from main memory and replace according to the algorithm.

1. FIFO – First in first out.
2. MFU – Most Frequently used
3. LRU – Least recently used.
4. Optimal page replacement (OPT)
5. Belady's algorithm
6. LFU – Least frequently used.

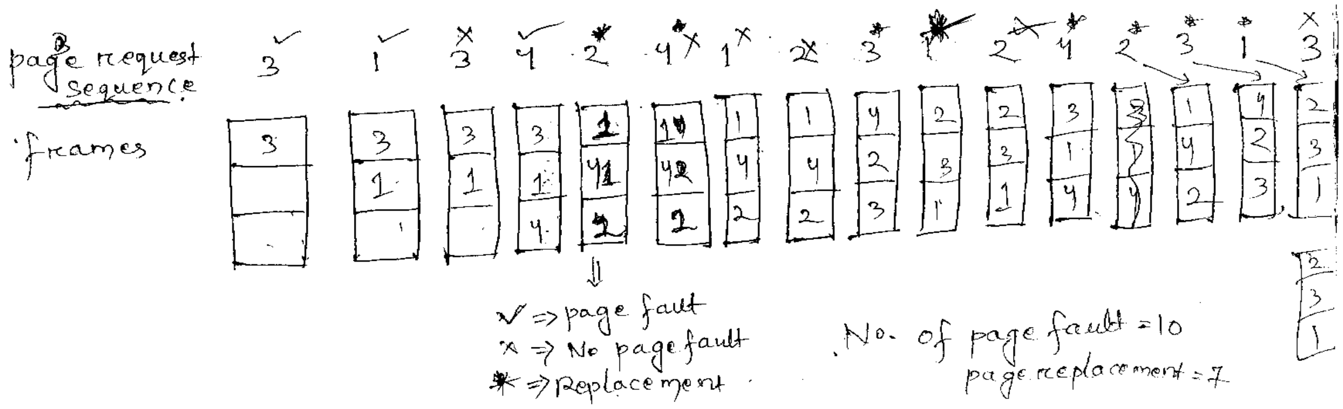
FIFO:-

⇒ It is the simplest algo. For the replacement of the page.
⇒ Replace the page that has been in memory longest.

Note:- i) calculate the page fault.

ii) Calculate the no. of replacement of pages.

iii) Calculate the page fault rate = (no. of page
iv) Several addresses is generated, when the pages are loades in the memory. These are specified as reference string.



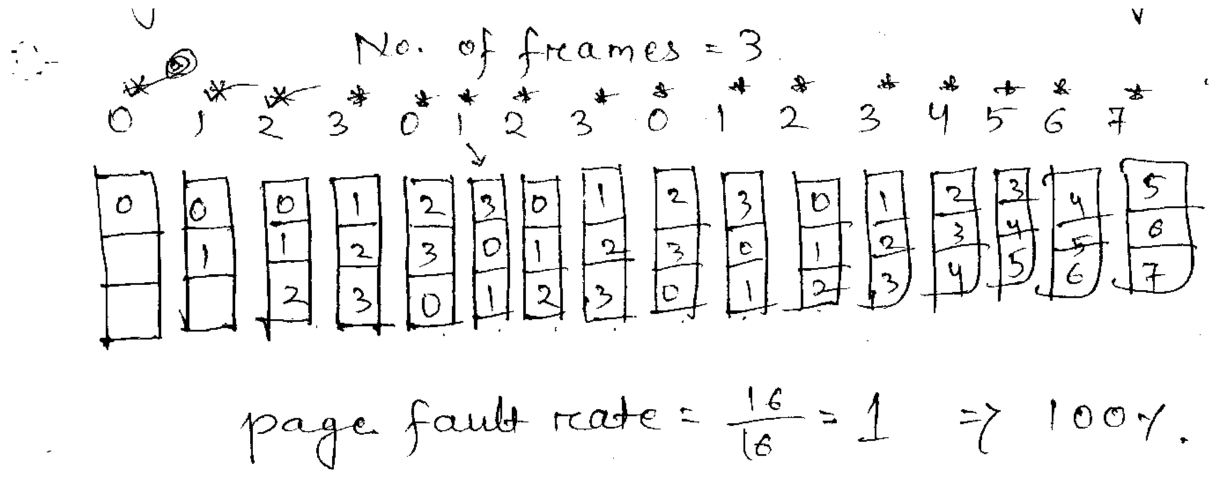
Note:- When the pages are field (page 2) is requested by the processor. In this time one of the page from the frame should be replaced. In FIFO the page 3 has 1st in. So it should be 1st out by making the other 2 pages upward and put the new requested page 2 in the last frame.

Calculate page fault rate:-

Page fault rate = 10/16=0.625

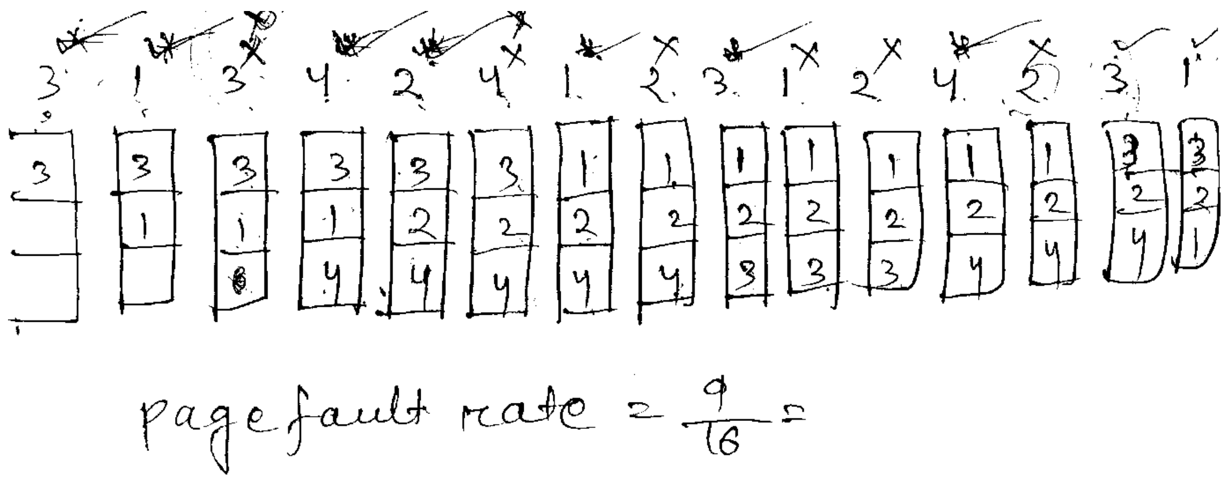
Note:- the algo having the least page fault rate can be the best algo.

Q: No. of frames = 3



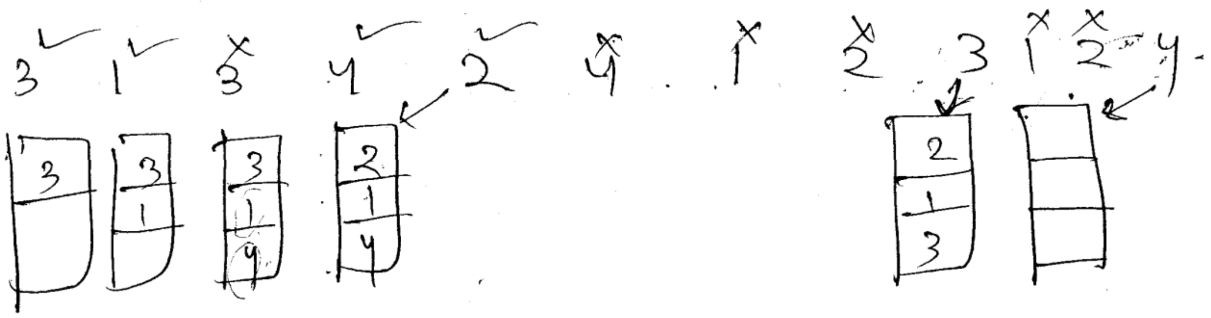
2) least recently used:-

- ⇒ Replace the page that have not been used for longest period of time.
- ⇒ Each time refer the string in back ward direction.



Optimal page replacement:-

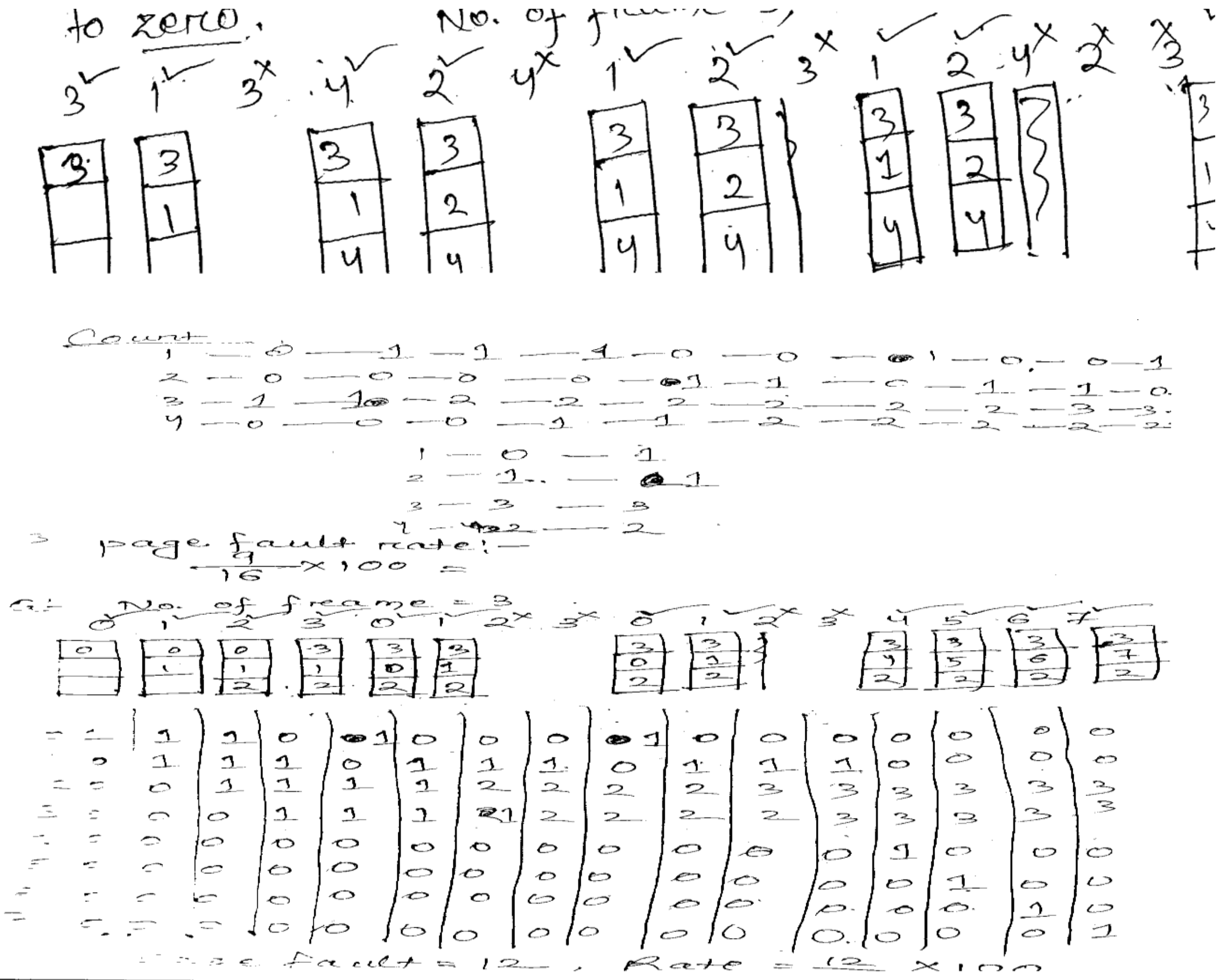
- ⇒ Replace the page that will not be used for longest period of time.
- ⇒ Each time refer the string in forward direction.
- ⇒ This algo. Shows the lowest page fault rate.



<u>LFU Least Frequently used):-</u>

Select a page for replacement if the page has not been used often in the past or replace the page that has smallest count.

Each time whenever a page fault will occur a page count has been maintained for each page. If the same occurrence or same count for the pages, then FIFO operation will be implement. Each time when one page is replaced, then the counter value for that page will be reset to zero.

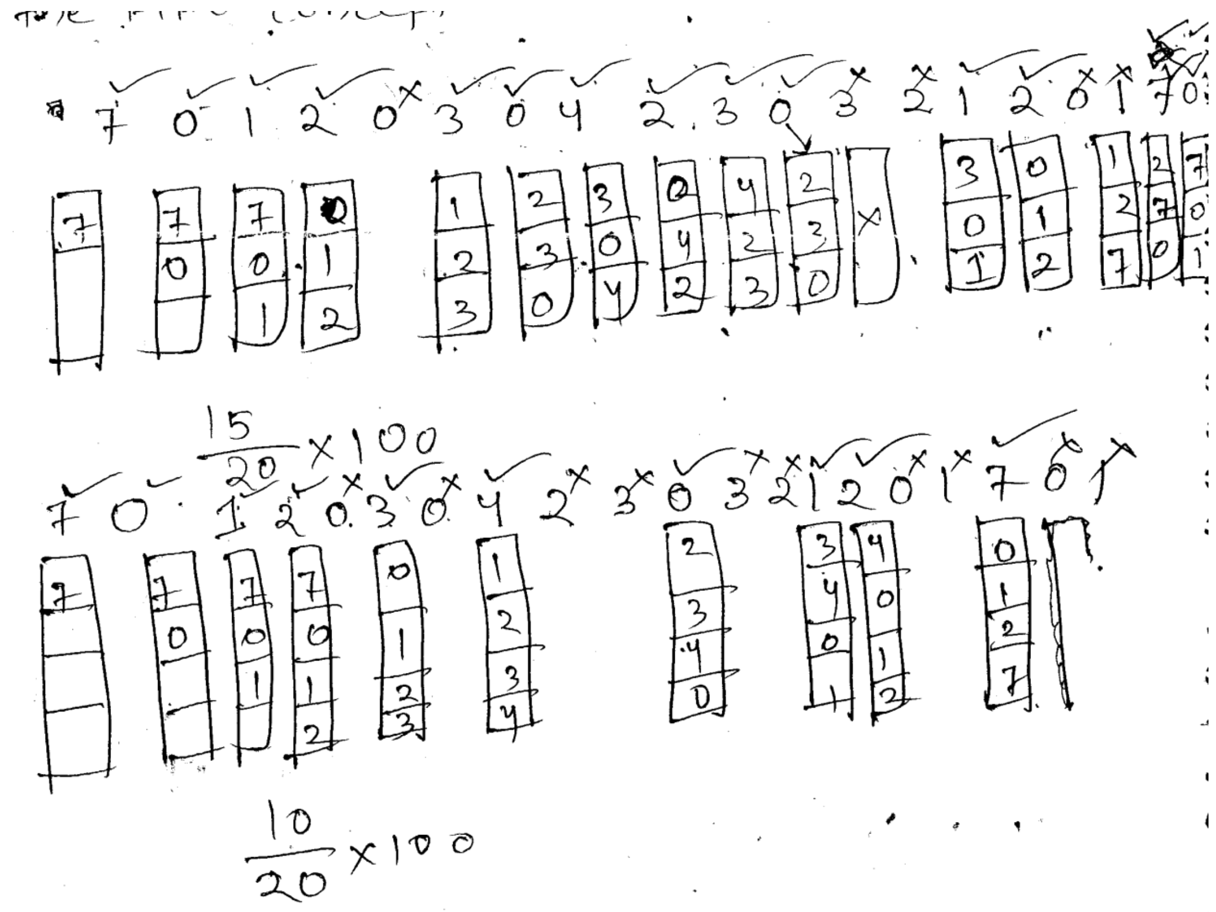


Belady’s Anamoly:-

If the no. of frames is increased, the page fault rate will be decreased. This unexpected phenomena is known as Belady’s anamoly.

Ex: sequence reference string = 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 1 7 0 1

i) No. of frames = 3
ii) Increase no. of frames = 4
iii) Manipulate the blady’s anamoly by using the FIFO concept.



Swapping:-

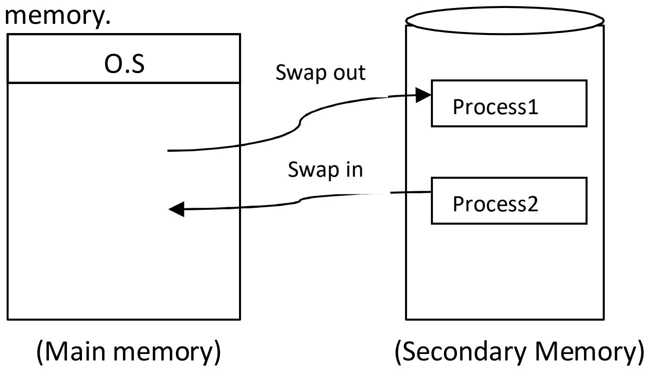
⇒ Swapping is a method, to improve the main memory utilization.

⇒ Suppose, CPU currently executing process number 5 and during the middle of execution, the process need to execute process number 9, then, the CPU switches to another hob and process 5, move to secondary storage.

⇒ Switches the process from main memory to secondary storage or disk is said as swap out.

⇒ Switches from secondary storage to main memory is known as swap in.

Ex:- The main memory consisting of 10 process, which, has the maximum capacity of the main memory.



Contiguous memory allocation:-

⇒ Contiguous memory allocation skim for efficient allocation of the process.

⇒ The memory is divided into 2 portion, i.e.
  i) Operating system
  ii) User

⇒ Operating system, keeps a table, which, is part of memory is available and which, are occupied. This is done by table.

⇒ Initially, the memory is treated as a single block, called hole.

⇒ The programme, enter a system that, are put in a queue.

⇒ If the hole is large, it is split in to two parts, i.e.
  i. Operating system.
  ii. Dynamic storage allocation problem,

⇒ Dynamic storage allocation problem is solved by
  i) First bit
  ii) Best bit
  iii) Worst bit

First bit:-
The hole, that, is large enough, is allocated.

Best bit:-
The smallest hole, that, is big enough.

Worst bit:-
The largest hole, available.

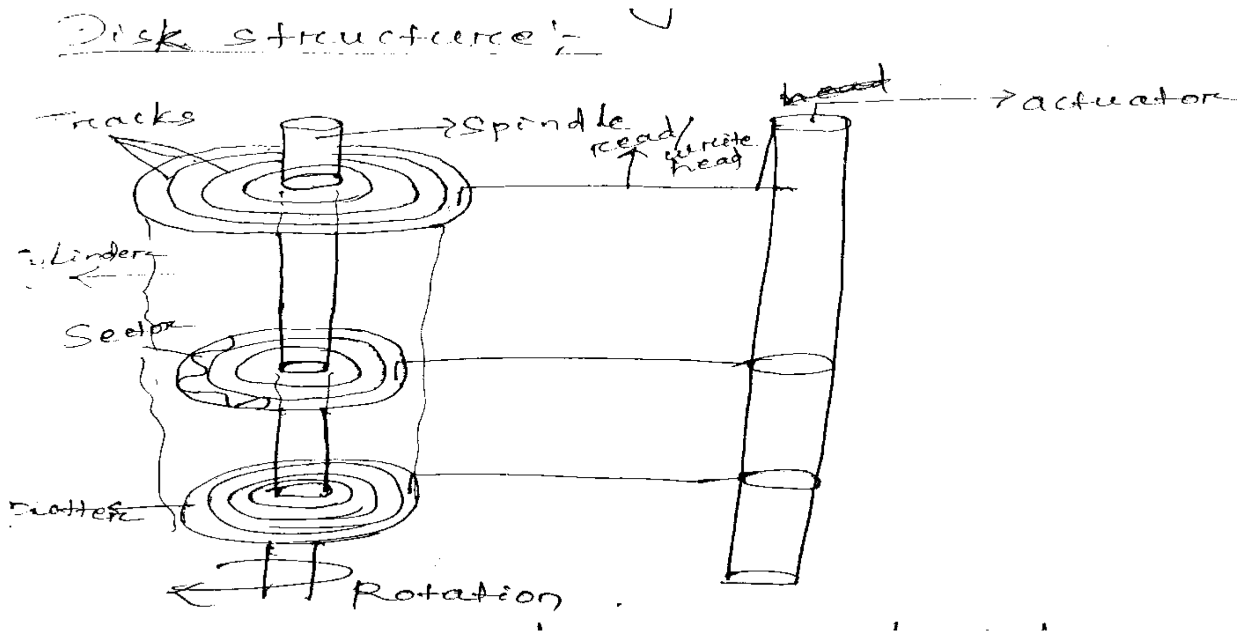
Questions:

1. What is paging?
2. What is page fault?
3. What is segmentation?
4. Difference between job scheduler and CPU scheduler.

CHAPTER-4

DEVICE MANAGEMENT

Disk structure :-



Disc provide a bulk of secondary storage on which the file and information are permanently stored.

Physical structure of a magnetic disc:-

Platter:- Each disc has a flat circular shape named as platter. The diameter of platter is from range 1.8^{11} to 2.25^{11} inches.
The information Are stored magnetically on platter.
Tracks:-
Surface of the platter are logically divided into circular tracks.
Sector:-
Tracks are subdivided into no. of sections termed as sector.
Each track may contain hundreds of sectors.
Cylinder:-
Set of tracks that are at one composition makes up a cylinder.
There may be thousands of concentric cylinders in disc drive.
Read/write head:-
This is present just above each surface of every platter.
Disc arm:-

The heads are attached to a disc arm that moves all the heads as a unit.

Note:- The storage capacity of a normal disk drive is measured by GB.

Seek time:-

The time required to reach the desired track by the read and write head is a seek time.

There are 2 components to calculate seek time.

i) Internal start up time
ii) The time taken to traverse the cylinder

Ts = m + n + s

Where Ts = Estimated seek time

n = no. of tracks traverses

m = constant

s = start up time.

Rotational delays time:-

The time required to reach the desired sector by write/ read head is called rotational delay time.

Generally the average of rotational delay is between 100 to 200 ms.

Disk scheduling algorithm:-

Whenever a process request an i/p, o/p operations from the basic it will issue a system call to the OS.

The request have been processed according to some sequence. This sequence has been made by various algo. Named as disk scheduling algo.

Disk sch. Algo. Schedules all the request properly and in some order.

This algo. Is implemented in a multi programming system.

Following are the disk scheduling algo.

i) FCFS
ii) SSTF (Shortest seek time first)
iii) SCAN
iv) C-SCAN (Circular scan)
v) Look

i) FCFS:- (First come First serve)

This algo. Is a simplest disk scheduling algo.

⇒ It doesn't provide fastest service.

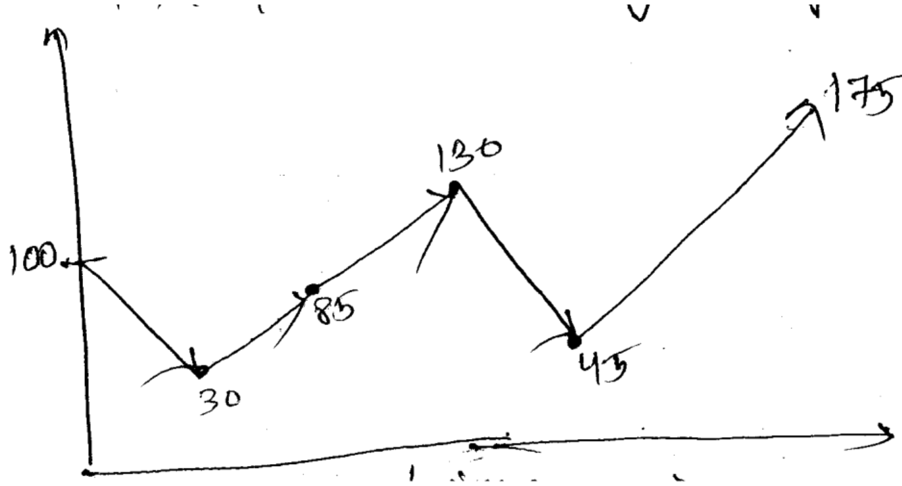
Q: the i/p, o/p pending request are 30, 85,130, 45, 175

Total no. of cylinder = 0 to 199.

Initially the disk head is at cylinder no. 100.

A: Steps to be followed:-

1) Generate the disk head movement.
   100-30
   30-85
   85-130
   130-45
   45-175
2) Draw the disk scheduling diagram



3) Calculate the total head movement.
   I100-30I + I30-85I + I85-130I + I130-45I + I45-175I
   = 375 cylinder.
4) Average head movements:
   3385/5 = 77
   In FCFS the avg. head movement is very high.

ii) <u>SSTF:-</u>
   This algo. Request chooses the pending request closest to the current head position. It will select the request which is having minimum seek time from the current head position.

Q: Pending are 30,90,115,45,130,175

Total no. of cylinder = 200

Initially the disk is at 100

A:- <u>1) Generate the head movements:-</u>

100-90

90-115

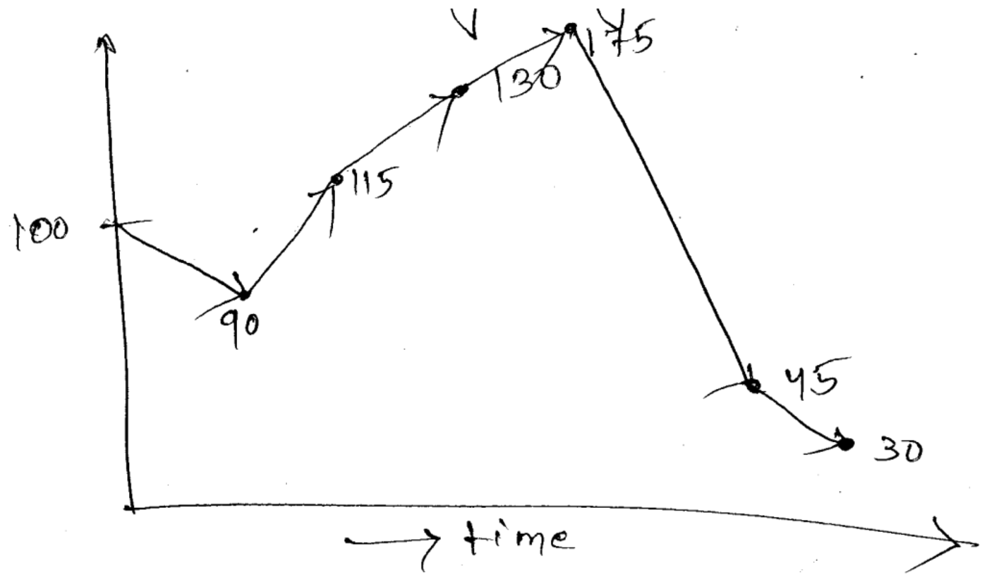
115-130

130-175

175-45

45-30

<u>2) Disc scheduling dg:</u>



<u>3) Total head movements:</u>

I100-90I + I90-115I+I115-130I+I130-175I+I175-45I+I45-30I

= 240

Avg. = 240/6 = 40

iii) <u>Scan Algo:-</u>

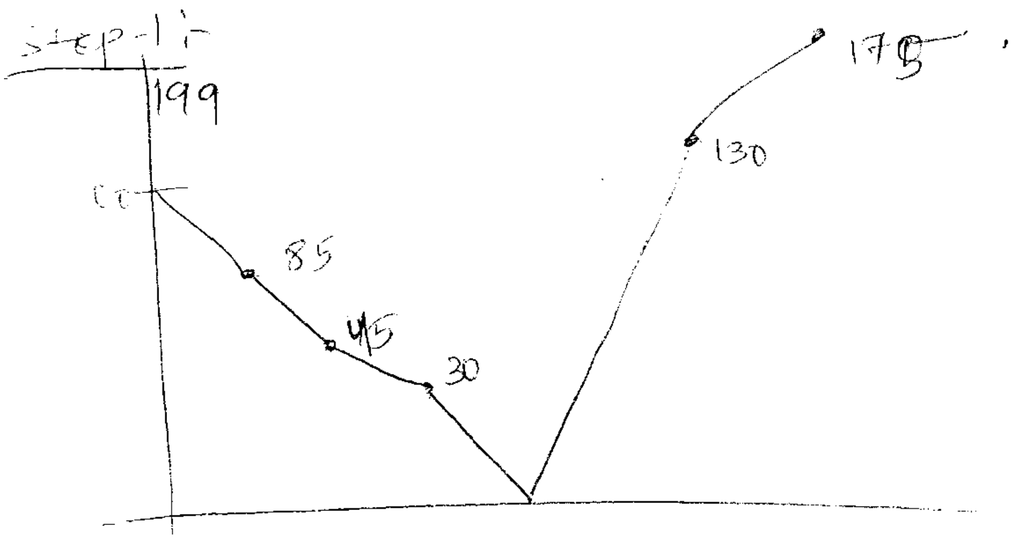
The disk are start at one end of the disk and move towards the other end by servicing the request. When it gets to the other end of the disk the head movement is reversed and servicing continues.

Pending request = 85 30 130 45 175

Total no. of cylinders = 200

Initially the head is at cylinder no. 100

Step:-1



Step:-2

Generate the head movement

100-85

85-45

45-30

30-0

0-130

130-175

Step:-3

Calculate the total head movement

I100-85I+I85-45I+I45-30I+I30-0I+I0-130I+I130-175I

= ( )

Step:-4

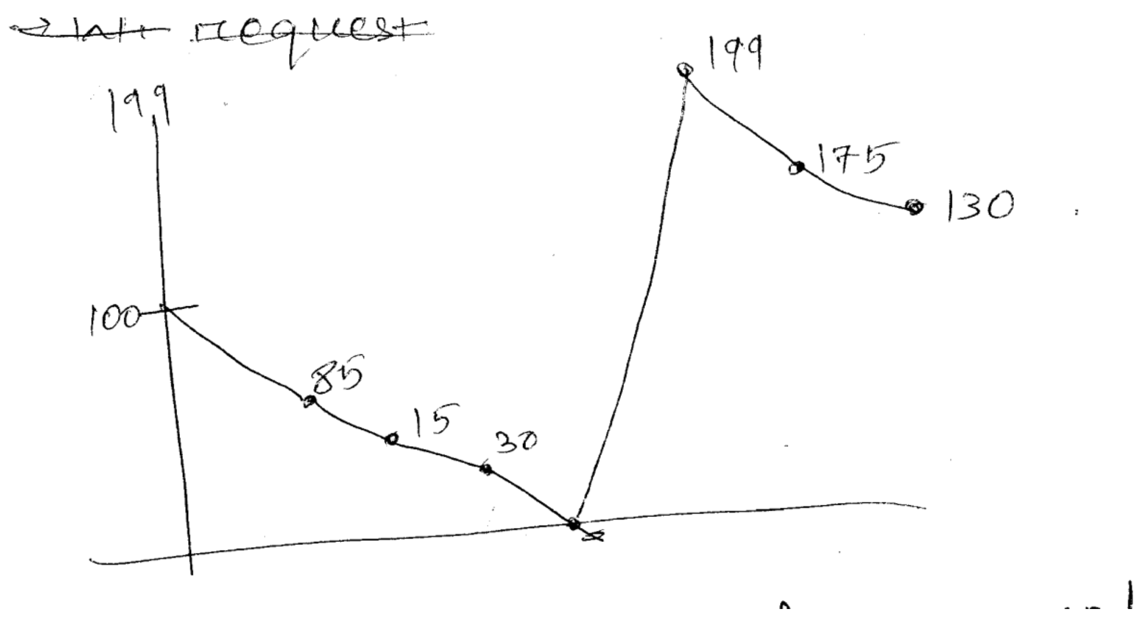
Calculate the avg. head movement

= ( )/5 = 45.83

C-Scan algo.:-

⇒ The head moves from one end of the disk to the other by servicing the request.

⇒ When it reaches to the other end, immediately it returns to the beginning of the disk.



Note:- When it is moving from one end to other end immediately it will not service any request. Here when the head moves from 0 to 199 it will not service any request.

Head movements =

100-85

85-15

15-30

30-0

0-199

199-175

Total head movement:-

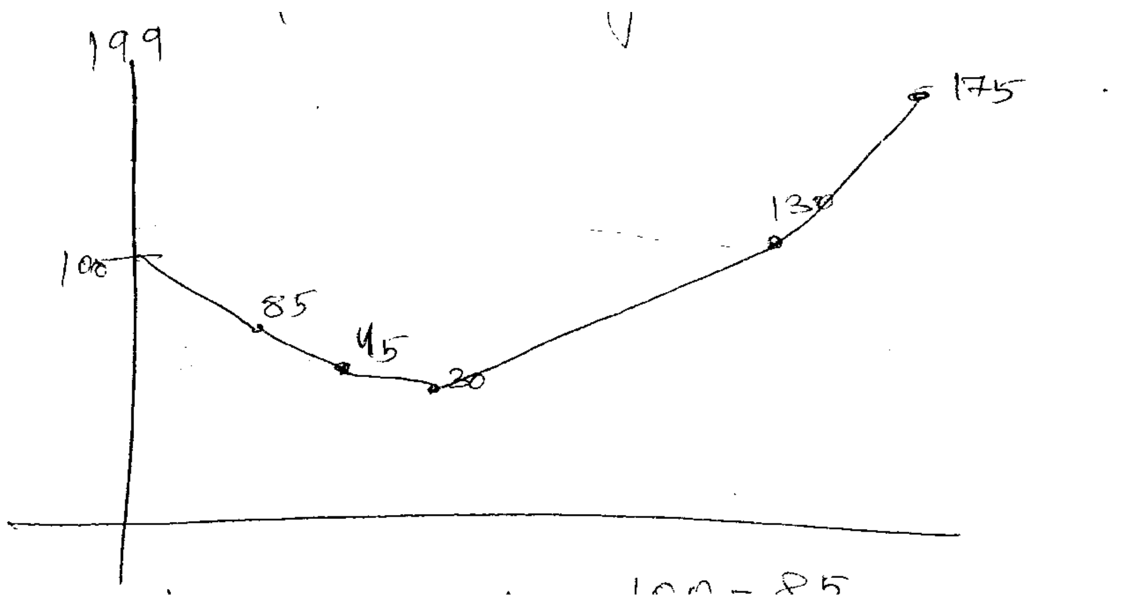
I100-85I+I85-15I+I15-30I+I30-0I+I0-190I+I199-175I+I175-130I = 368

Avg. head movement :-

= 368/5 = 73.6

<u>Look scheduling algo:-</u>

The disk arm start at first i/p and o/p request on the disk and move towards the last i/p and o/p request on the other end. Servicing the request. When it gets the last i/o request, the head movement is reversed and servicing continues.



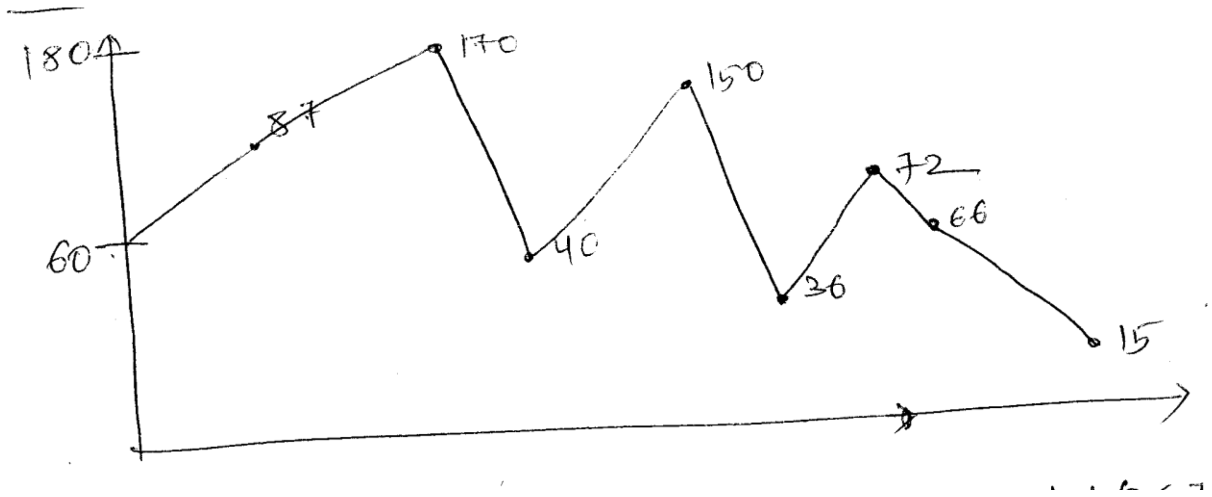
Q: Perform all the disk scheduling algo. For the following sequence of i/o request.

87, 170, 40, 150, 36, 72, 66, 15

Total no. of cylinder 181

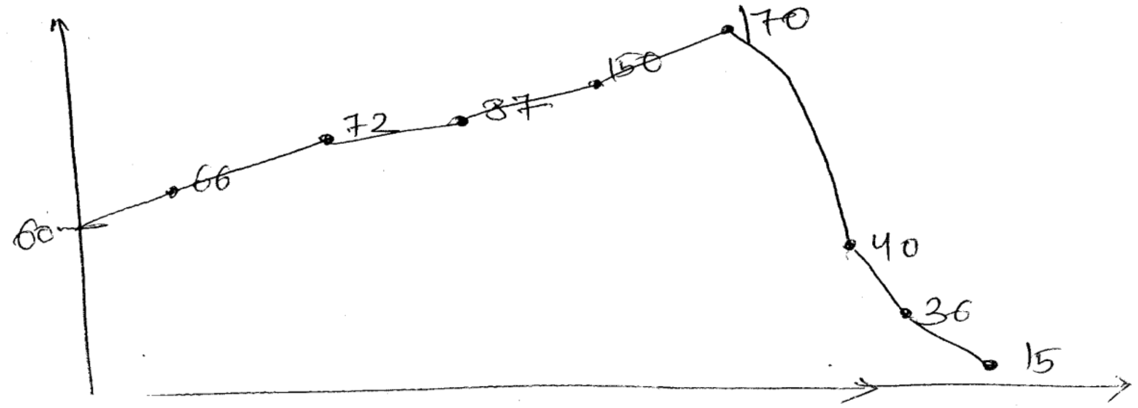
Initially disk head is at 60

A:- <u>FCFS</u>:-



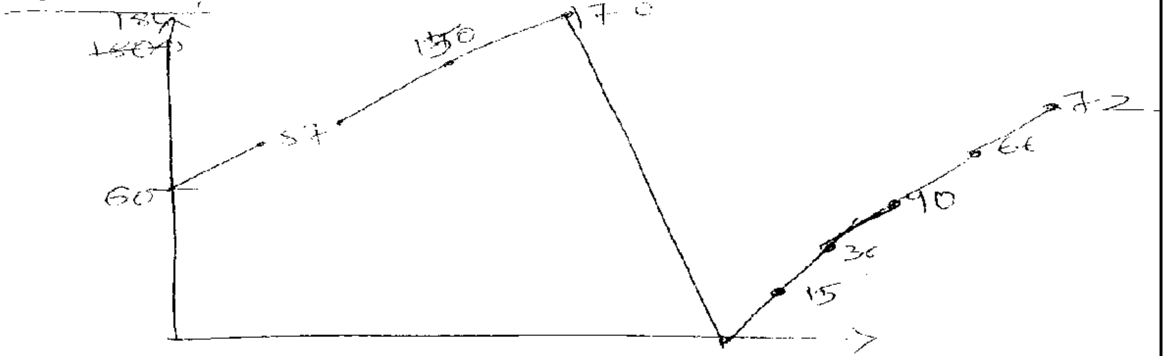
I60-87I+I87-170I+I170-40I+I150-36I+I36-72I+I72-66+I66-15I = 557
AVG. = 557/8 = 69.625

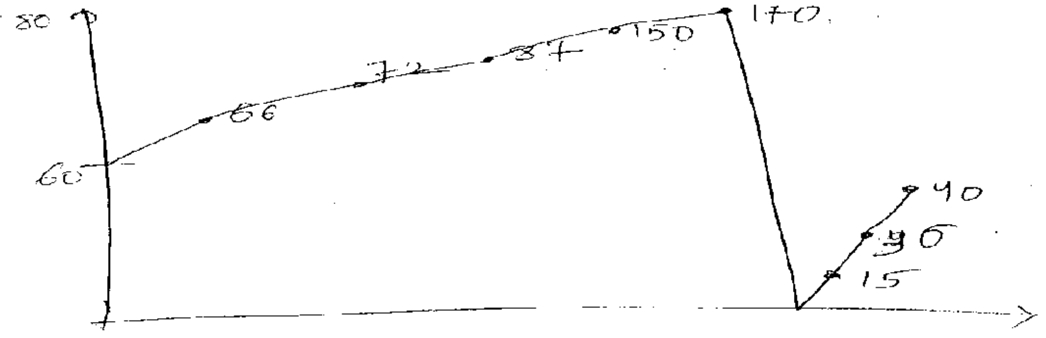
SSTF :-



$\frac{265}{5} = 33.125$

[illegible]





$\Rightarrow$ 326

Avg = 90.75

	CPU burst	priority	Arrival time	
P1	109	3	0	0
P2	10	1	1 3	
P3	21	3	22	
P4	54	2	3 1	

Gantt Chart:- (Priority)

P1	P2	P3	P4	P1	P3	
0	1	2	3	8	17	18

TAT =

P1 = 17-0 = 17

P2 = 2-1 = 1

P3 = 18-2=16

P4 = 8-3=5

Avg. TAT = 17+1+16+5/4

=39/4 = 9.75

WT for P1 = (0-0)+(8-1)=9

P2=(1-1)=0

P3=(2-2)+(17-3)=14

P4+(3-3)=0

Avg. WT = 9+0+0+14/4 = 23/4 = 5.75

RR:-

P1	P	P3	P4	P1	P3	P4	P1	P4	P1	P4	P1	P1	P4	P1	P1	P1	P1	P1
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

TAT for P1 = (18-0)=18

P2 = (2-1)=1

P3=(6-2)=4

P4=(13-3)=10

Avg. TAT = 18+1+4+10/4

= 33/4 = 8.25

WT for P1= (0-0)≠0+(4-1)+ (7-5)+ (9-8)+(11-10)+(13-12)+(14-13)+(15-14)+16-15

P2 = (1-1) = 0

P3 = (2-2)+(5.3) = 0+2=2

P4 =

P1 = (4-1)+(7-5)+(9-8)+(11-10)+(13-12)+(14-13)+(15-14)+(16-15)+(17-16)+(18-17)

=3+2+1+1+1+1+1+1+1+1

Device management technique:-

There are 3 techniques for device management, i.e.

i) Dedicated.
ii) Virtual
iii) Shared

Dedicated:-
Dedicated device is allocated a job for several user at the same time.
Ex.:-Printer, card reader and disk.

Virtual:-
Some dedicated device are converted to shared device, through, the technique, known as virtual device.

Shared:-
This device can shared, several processes at the same time.

I/O traffic controller:-

I/O traffic controller, control all the device track or channel.

The traffic controller maintain all the status information.

The traffic controller, attend 3 questions, i.e.

i) Is there a path, available to server or I/O request?
ii) If more, than, one path available.
iii) If no path currently available, when, one will be free.

In order to answer these question, I/O traffic controller use one of the following data base, i.e.,

i) Unit control block (UCB)
ii) Central Unit Control Block (CUCB)
iii) Channel control Block (CCB)

UCB	CUCB	CCB
⇒ Device unit identification. ⇒ Status of device. ⇒ List of control unit, connected to the device. ⇒ List of processes, waiting for this device.	⇒ It is control unit identification ⇒ List of device connected to the control unit. ⇒ List of channel connected to the control unit ⇒ Waiting for the control.	⇒ Channel identification. ⇒ Status of the channel. ⇒ List of control unit connected to the channel. ⇒ List of the process, waiting for the channel

I/O scheduler:-

If there are more I/O request, pending, then, available path is necessary to choose, which, I/O request is satisfied, first.

Then, the process of scheduling, is applied here, and it is known as I/O scheduler.

I/O device handler:-

I/O device handler, perform, the error handling technic in I/O scheduler.

Spooling:-

A spool is a buffer, that, hold the OIP.

Ex:- Printer.

Although, a printer can search only one job at a time. Several application are wish to print their OIP, currently, without having their OIP, mixed together.

The operating system, solve this problem, using separate disk and it is known as, spooling.

Spooling is managed by system process.

Race condition:-

Race condition is a situation, where, several process, access and manipulate, same data, concurrently and the execution depend on a particular order.

System calls:-

It is a process, that, provide, interface between user and operating system.

System calls are 5 types, i.e.

i) Process control
ii) File management
iii) Device management
iv) Information maintenance
v) Communication

Questions:-

1. Physical structure of magnetic disk.

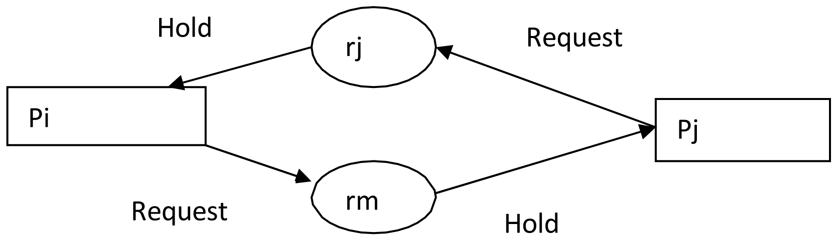
CHAPTER -5

DEADLOCKS

⇒ Deadlock means a lock having no key.

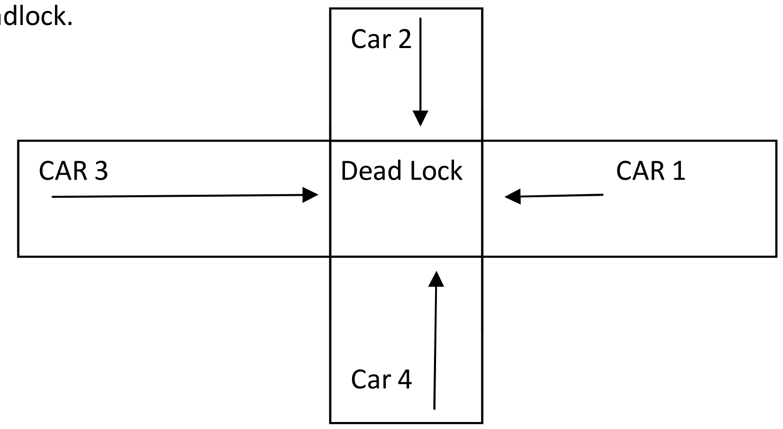
⇒ In multiprogramming system multiple no. of processes may compete for the finite no. of resources.

⇒ During this competition several process may move to a waiting condition (indefinite blocking) due to unavailability of resources. So none of the process will be executed. This unavoidable situation for which multiple no. of processes blocked is called deadlock.



⇒ In the above diagram Pi is holding the resource rj and requesting the resource rm (which is hold by Pj).

⇒ Pj is holding the resource rm and requesting for rj (which is hold by Pi). So neither the process pi and pj being executed. They will move to an waiting condition. This situation is known as deadlock.



Here we assume car = process.

Street = resource.

Each process (car) wants to allocate the resource (a section of street) which is hold by another process. So neither of the car can proceed further, since the resource need by each of them is occupied by other. So all the cars move to an indefinite blocking condition. This situation is named as dead lock.

Types of resources:-

There are 2 types of resources.

i.e. Physical (Printer, scanner, memory space, tape driver etc.)

logical (semaphores, monitors etc.)

<u>steps for allocation of resource to a process:-</u>

1) <u>Request:-</u>process request for a resource through a system call. If the resource is not available it will wait.
2) <u>Use:-</u> After getting the resource, the process can make use of it by performing the execution.
3) <u>Release:-</u> After the completion of the task the resource is not required by that process, in that it should be released.

<u>Preempt able resource:-</u>

⇒ A resource is a preemptable one if it can be taken away from the process without causing any effect. Ex: memory.

<u>Necessary condition for deadlock:-</u>

1) Mutual exclusion
2) Hold and wait
3) No preemption
4) Circular wait.

1. <u>Mutual exclusion:-</u>
   ⇒ The resource involved are non-sharable.
   ⇒ At least one resource must be held in a non sharable mode. i.e. only one process at a time have the exclusive access to that resource.
   ⇒ If another process request that resource, the requesting process must wait until the resources has been released.
2. <u>Hold and wait condition:-</u>
   ⇒ A requesting process holding a resource and requesting for another resource.
   ⇒ There must exist a process i.e. holding a resource already allocate to it while waiting for another resource that are currently held by other processes.
3. <u>No preemption:-</u>
   ⇒ Resources already allocated to a process can not be preempted.
   ⇒ Resources can't be removed from the processes temporarily by the process holding it.
4. <u>Circular wait:-</u>
   ⇒ The process in the system form a circular list or chain where each process in the list is waiting for a resource held by the next process in the least.

<u>Process synchronization:-</u>

⇒ A cooperating process is one that can be affect or be affected by other processes executing in the system.

⇒ These processes are sharing their address spaces.

⇒ Process synchronization deals with various mechanism to ensure orderly execution of co-operating processes.

⇒ The basic purpose of process synchronization is to maintain data consistency.

Q:- Define race condition.

A: The race condition is a situation in which more than on process access a shared resource concurrently and the result depends on the order of execution.

Ex:

```
Int count = 10

P1 (00000)

{

 do something;

Count ++;

}

P2(…..)

{

Do something ;

Count--- ;

}
```

P1	P2	Operation
Do	Do	Initialize the count
Load count		P1 starts executing
Count ++		Count is being incremented.
	Load count	P2 starts executing
	Count ---	P2 count is being decremented
Save count		In memory 11
	Save count	In memory 9

In the above example the following sequence shows a race condition.

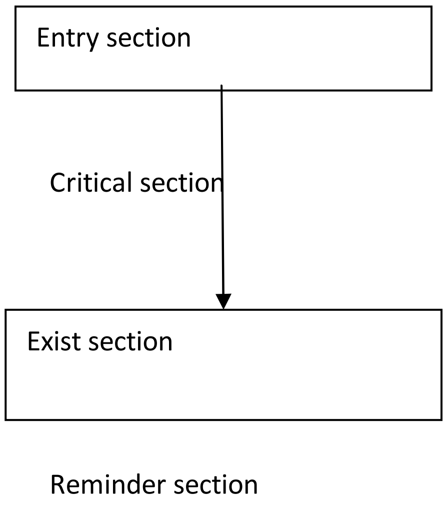
i) 2 processes run concurrently.

ii) Bothe the process the shared variable count at the same time.

iii) Finally the computation result depends on the order of execution of save instruction.

Critical section:- (C.S)

- ⇒ A section of code in which only one process may be executing at a given time is called critical section.
- ⇒ In C.s. a process may changes the common variable, update the table, writing into a file etc.
- ⇒ When one process is executing in its critical section, no other process is allowed to enter into that.
- ⇒ The process execution in C.s. is mutually exclusive in time.
- ⇒ The C.S. code must executed very quickly.
- ⇒ A cooperating process performs execution through 3 section i.e.
  - i) Entry section
  - ii) Exist section
  - iii) Reminder section



Entry section:-

- ⇒ Each process must request to enter into critical section. The section of code performing the request is known as entry section.

Exist section:-

- ⇒ The entry section is followed by a section after using the shared variable is known as exit section.

Reminder section:-

- ⇒ The remaining code of the process (not associates with the shared variable) is present in the reminder section.

Resource allocation graph (RAG):-

- ⇒ A diagrammatic represent to det. The existence of deadlock in the system by using a graph named as RAG.

⇒ It is a directed graph.

⇒ RAG consists of several no. of nodes and edges.

⇒ It contains i) Process node P Circle ii) Resource node R Square.

⇒ The bullet symbol within the resource is known as instances of that resource.

⇒ Instance of resources means identical resources of same type.

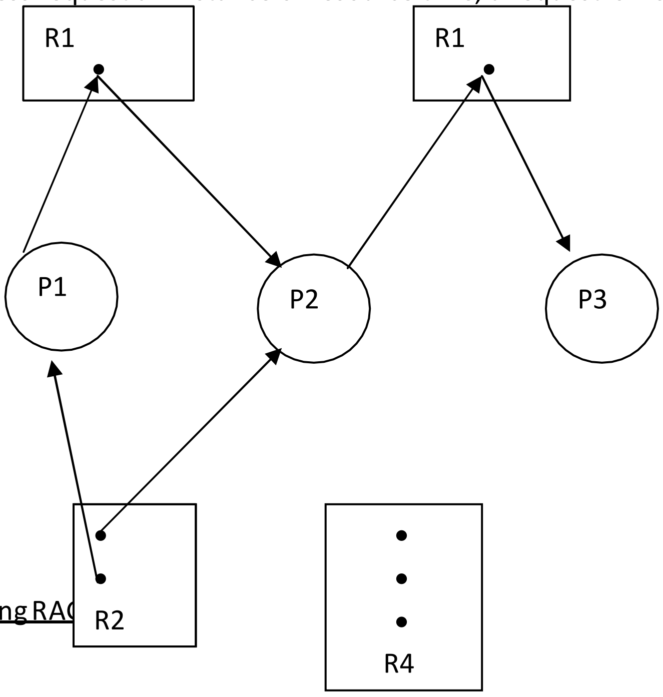
⇒ There exist 2 kinds of edges.

i) Allocation / assignment edge

ii) Request edge.

⇒ Allocation edge is directed from resource to process releases a resource allocate to it.

⇒ When a process request an instance of resource time, a request is inserted in RAG.



Information regarding RA

Step-1

⇒ Information about processes, resources and edges.

P = {P1,P2,P3}

R = {R1,R2,R3,R4}

E={P1 to R1, R1 to P2, P2 to R3, R3 to R3

⇒ 2: Information about instance of resource type.

R1=1

R2=2

R3=1

R4=3

⇒ 3: Information about the status of processes and resources.

i) P1 is requesting an instance of resource R1 and holding the instance of resource R2.

ii) P2 is requesting for R3 by holding R2.

iii) P3 is holding the instance of resource R3.

Reorganization of deadlock from RAG:-

⇒ Deadlock in the system equals to cycle in the graph.

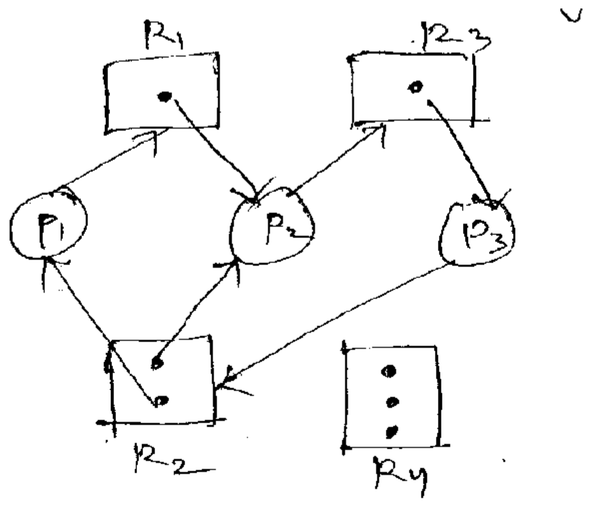
⇒ If the graph contain no cycle, then no process is in deadlocked.

⇒ If there is a cycle, then

i) If resources type have multiple instances, then deadlock may exist.

ii) If each resources type has one instance, then deadlock has occurred.

Resource allocation graph with a deadlock:-



Two cycles in the above cycles are:-

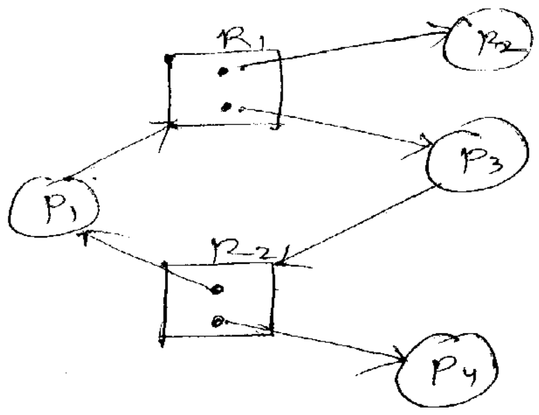
i) P1R1P2R3P3R2P1

ii) P2R3P3R2P2

i) The no. of process present in deadlock is 3 i.e. P1, P2, P3.

ii) No of processes present in deadlock is 2 i.e. P2, P3.

Resource allocation graph with a cycle but no deadlock:-



Note:- No cycle = No deadlock

Cycle = mayor may not deadlock.

General structure of a process in critical section:-

```
do

{entry section

}

//critical section

}

Exit section

Reminder section

} while (true);
```

Solution to the critical section problem

To overcome the problem of critical section 3 condition should satisfy.

i.e.

i) Mutual exclusion

ii) progress

iii) Bounded waiting.

i) Mutual Exclusion:-Only one process will be executed in the critical section at a time.
ii) Progress:- The process can request to enter into the critical section is 2 condition.
    a) If no process is currently executing in the critical section.
    b) The processes shouldn't be part of reminder section.
iii) Bounded waiting:- There exist a bound or limit on the no. of times that other processes are allowed to enter in the critical section (through request can be made by the processes several times, but finite time can access the critical section.

Peterson's solution:-

```
{
Flag[i] = true
Turn = j;
While (flag[j] && turn ==j);
Critical section
Flag [i] = FALSE;
Reminder section
} while (TRUE);
```

Solution for Pj:-

```
 Do{

Flag[j] = TRUE;

Turn = I;

While (flag[i]&&turn==i);


Critical section

Flag [j] – FALSE;

Reminder section

} while (TRUE);
```

All the 3 conditions mutual exclusion, progress and bounded waiting is preserved in peterson's solution. So it can be considered as a best solution to the critical solution problem.

SEMAPHORE

⇒ It is a synchronization tool, denoted as 'S' which is an integer variable whose value can be changed and altered.

⇒ Its value indicates the status of shared resources, a process which needs the resource, will check the semaphore for determining the status fo the resource (available/unavailable).

⇒ If the value of the semaphore variable can be changed by two operations.

iii) Wait (P)

iv) Signal (v)

Wait of S:-

⇒ Wait of S an atomic operation that waits for semaphore of the value of S is > 0, then decrement it by 1 or else wait on S.

```
If (S>0)        wait(s)
Then S          {
S=S-1           while (S≤0)
Else            {
Waitons         s---;
                }
                }
```

Signal of 'S':-

⇒ Signal of S is an atomic operation that increments 'S' by 1.

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

⇒ When one process modifies the value of the S, no other process can simultaneously modify the same S value.

S are of 2 types:-

iii) Binary S

iv) Counting S

iii) Binary S:-

⇒ Binary S can take 2 values.

⇒ These values ranges between 0 to 1.

⇒ These S are used to acquire the lock.

⇒ The lock used by the binary S is termed as MUTEX lock.

```
do
{
Waiting(mutex);
// c.s
Signal(mutex);
//reminder section
{while(true);
```

iv) Counting S:-

⇒ The counting S is applicable for multiple instances of resource type.

⇒ Each process that wants to use the resource performs wait operation on the S.

⇒ When a process release the resource, it perform signal operation.

⇒ When the count of the S goes to zero all the resources are being used.

⇒ When the count > 0, he resources is unblocks

Disadvantage in S:-

⇒ The main disadvantage in S is busy waiting.

⇒ While a process is executing in critical section the other processes are requesting continuously looping, in the entry section by making the process busy wait.

⇒ Busy waiting cause wastage of CPU time, CPU cycle, m/m space etc. during busy waiting process spins in the entry section to get the lock. So this type of semaphore status is termed as spine lock.

To overcome the busy waiting implementation of semaphore:-

- ⇒ To overcome the problem occurred in busywaiting the wait () and signal() operation are modified.
- ⇒ When the process executes the wait operation, rather than staying in waiting condition, the process blocked itself. During this blocking it is associated with a waiting queue.
- ⇒ Similarly when a signal operation is occurred a process is removed from the queue and executes.
- ⇒ Each entry is a waiting queue has 2 data values.
  - iii) Value of integer type.
  - iv) Pointer to the next record in the list

During this 2 operations will be performed

- iii) Block () →modification of wait ()
- iv) Wake up ()→modification of signal()

i) Reader-writer problem:-

A database is to be shared by the concurrent processes. 2 operation made in a database. i.e.

a) Read
b) Write operation

Read operation will be done by reader processes
Write operation will be done by writer process.

Reader:-

This process only read the database, can't update.

Writer:-

This process will both read and write the database.

A common synchronization problem arise when a resource can be used sharable by any no of readers and exclusively by one writer at a time.

- ⇒ This synchronization problem is known as reader – writer problem.
- ⇒ These problems are of 2 categories.
  - i) First reader writer problem
  - ii) Second reader writer problem

i) First reader writer problem:-

No reader will be kept waiting unless a writer has obtained the permission to sue the shared object.

- ⇒ No reader should be wait for other readers to finish because a writer is waiting.

ii) Second reader writer problem:-

Once a writer is ready then that writer has to perform the writer operation as soon as possible.

⇒ If writer is waiting to access the object no new reader may start reading.

⇒ In this problem 2 semaphore will be used.
i.e. a) Writ, b) Mutex which is initialized to 1
An another variable will be used reader which is initialized to zero.

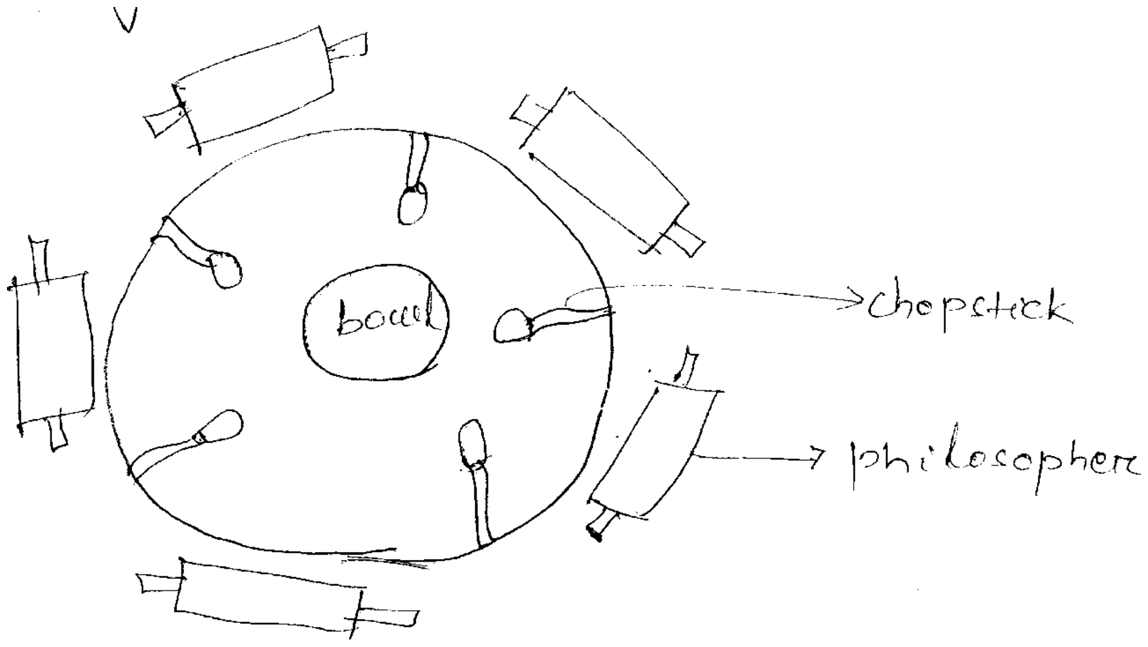
```
Reader                              Writer
P(mutex)                            P(writ)
Reader = reader+1                   -----
If (reader==1)                      Do writing
P(wrt);                             V(wrt);
V(mutex);
-----
-----
Do reading
P(mutex);
Reader = reader-1;
If (reader==0)
V(wrt);
V(mutex);
```

⇒ During the first reader writer problem the writer may starve.

⇒ During second reader writer problem the reader may starve.

⇒ The reader writer problem and solution is generalized by using a lock and it is known as reader writer lock.

iii) Dining philosopher:-



In this problem there are 5 processes, philosopher→process using 2 shared variables, chopstick→shared variable to take the rice from the bowl present in the center of the table.

2 operations occurs i.e. i) thinking, ii) eating.

Whenever a philosopher is thinking he doesn’t interact with his neighbors. Each time the philosopher gets hungry and he tries to pick up the chopsticks.

A philosopher may pick up only one chopstick at a time.

When an hungry philosopher has both his chopsticks he eats and after finishing eating he can releases chopsticks and can start thinking again.

```
Do
{
Wait (chopstick[i]);
Wait(chopstick[(i+1)%5]);
-------
//eat
Signal(chopstick[i]);
Signal(chopstick[(i+1)%5]);
---
Think;
-----
} while (true);
```

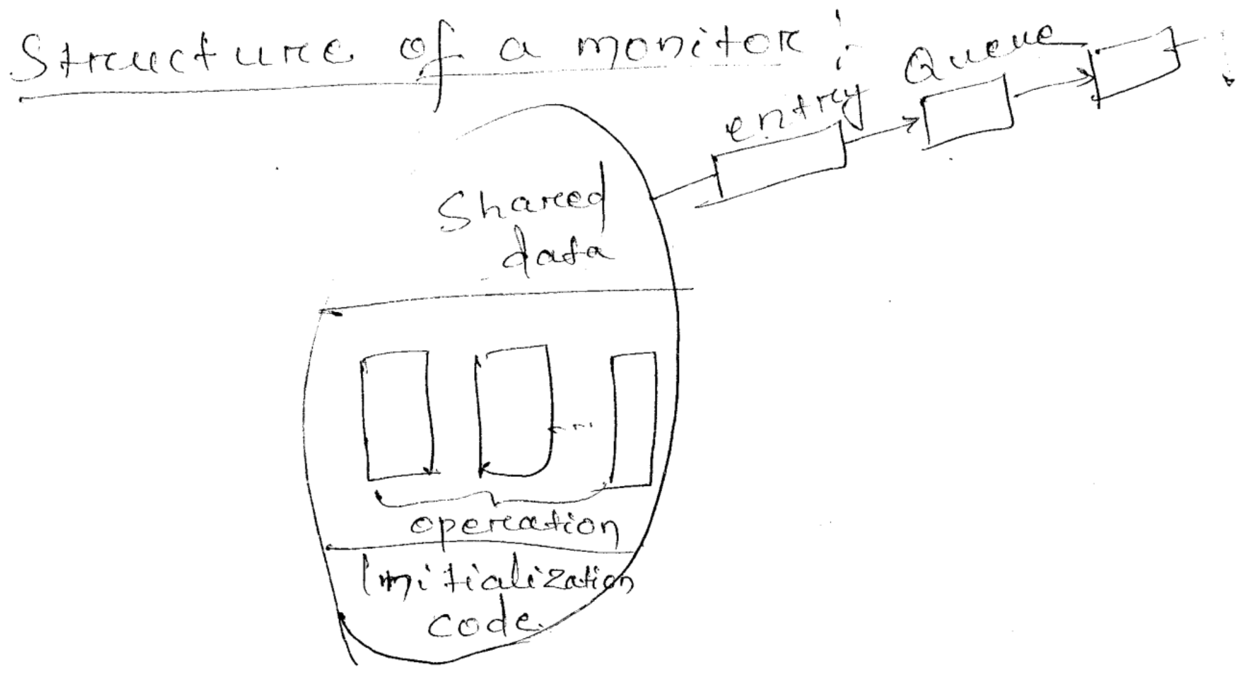
- ⇒ A philosopher tries to take a chopsticks by executing wait operation.
- ⇒ He releases his chopstick by executing signal operation.
- ⇒ No 2 neighbors can eat simultaneously because by this, the deadlock can occur.
- ⇒ All 5 philosopher become hungry simultaneously and each grabs their left chopstick.
- ⇒ Following might be an expected solution for dining philosopher.
  - i) Allow almost 4 philosopher to be sitting simultaneously at the table.
  - ii) Allow the philosopher o pick up the chopsticks when both are available.
  - iii) One odd philosopher pick up first his left chopsticks and then his right chopsticks.
  - iv) One even philosopher picks up his right chopsticks and then his left chopsticks.

<u>Monitor:-</u>

- ⇒ During the use of several synchronization model various types of error are generated to solve critical section problem.
- ⇒ To deal with such error a high level synchronization model is developed named as monitor type.

```
Monitor monitor name
{
//shared variable declaration;
Producer P(---);
{
---
}
Procedure P2(---);
{ ---
}
Procedure Pn(---);
{ ---
}
Initialization code (---_
{---}
```

Structure of a monitor:-



⇒ Each monitor having 3 components, i.e.
   a) Sharable data
   b) Producers operating on data
   c) Initialization code

⇒ With in the monitor only one process can be active at a time.

⇒ The queues are used for the execution of the processes and these are known as producers.

⇒ Initialization code contains some variables named as condition variable.

⇒ Based on condition variable 2 operations can be performed i.e. a) wait, b) signal.

⇒ A monitor type represents a set of programme defined operations. These operations are mutual exclusive within the monitor.

⇒ A procedure must be defined with in the monitor and can access only those variable those are declared with in the monitor.

Deadlock Prevention:-

There are 4 conditions to be required.

⇒ Mutual exclusion

⇒ Hold and wait

⇒ No preemption

⇒ Circular wait

Deadlock occur in the system .

⇒ Deadlock prevention gives the mechanism at least one of the condition should not occur in the system so that deadlock can be avoided.

1) Mutual exclusion:-

⇒ The resources in the system should be sharable.

⇒ Sharable resource doesn't have mutual exclusion property. So it can avoid the occurrence of deadlock in the system.
Ex.:-Read only file is a sharable resource.

⇒ If several processes attempt to open an read only file, they can access the file at the same time.
Note:- Every resources of the system cannot be made a sharable type.

2) Hold and wait:-
Whenever a process requires resources it doesn't hold any other resource. By this the deadlock can be prevented.

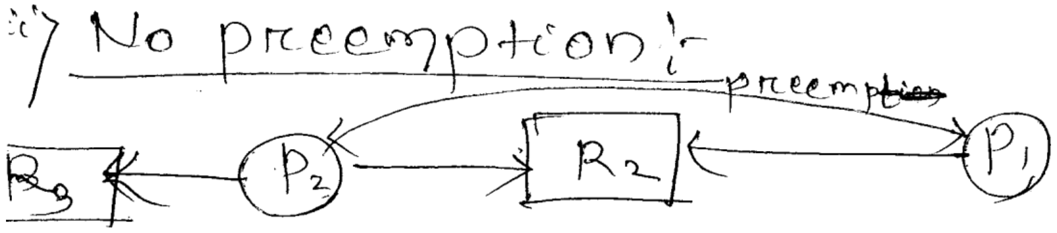
⇒ Following 2 rules can be used to prevent hold and wait condition.

i) Each process request and allocated all the required resources before its execution.
ii) A process should request a resource when it has no allocated resource and use the resource when additional resource will be requested all the holding requested should be released.

Note:- in the above situation, the deadlock can be some means prevented, but following 2 problems may arise.

i) CPU utilization will be very low.
ii) Some processes may move to a starvation situation.

i) No preemption:-



To prevent deadlock from the system, make the resources permutable type.

For making the resource preemptive type. Following are the conditions.

i) If the process request some resources, then it should check first whether the resource is available or not, if the resource is available then allocate to that required process.
If the resources are available, then check whether they are allocated to some other processes which are in waiting state.
In this situation preempt the resource from process and allocate them to the first requesting process.
If the resource is neither hold or available then the process moves to an waiting condition.
During the waiting situation it should release some of the allocated resource. After the availability of the resources the waiting process can be restarted.

iv) circular wait:-

to prevent the deadlock circular condition shouldn't occur in the system.

Let R-{R1,R2,R3, --- Rn}

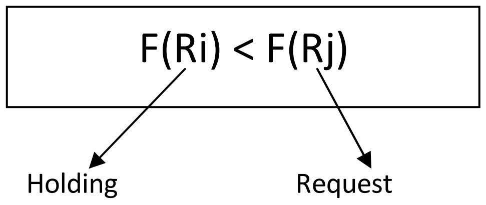
R is a set of resources.

In this each resource type should be assigned with a numeric integer value and it should be in an order.

⇒ Each process should request some of the resources in an increasing order of their numerical value.

⇒ Once a process holding some resources it can allocate or request new resources only if no. of all holding resources is less than the numerical value of the requesting resources.

⇒



Deadlock avoidance:-

⇒ Deadlock can be avoided by considering the future request by the process.

⇒ For the deadlock avoidance 2 approaches should be followed.
1) Never start a process if it demands tends to deadlock.
2) Don't allocate any additional resource if this allocation lead to deadlock.

safe state:-

⇒ A state is said to be safe, it system can allocate the resources to each processes by following some sequence/ order without causing deadlock in the system.

⇒ A system is said to be safe if there only exist a safe sequence (fig-1)

⇒ Safe state = safe sequence = no deadlock

⇒ Unsafe state = unsafe sequence = deadlock

⇒ There are 2 deadlock avoidance algorithm.
i) Resource request algorithm
ii) Banker's algorithm.

i) Resource request algorithm:-

⇒ In this algo. Deadlock can be avoided if the system having single instance of each resource type.

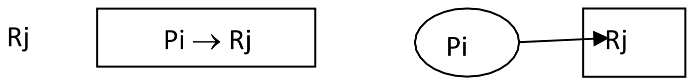
⇒ There are exist 3 edges.
i) Resource request algorithm.
ii) Banker's algorithm.

i) Resource request algo:-

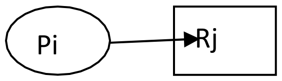
⇒ In this algo. Deadlock can be avoided if the system having single instance of each resource type.

⇒ There are exist 3 edges.
  i) request edge
  ii) Allocation edge
  iii) Claim edge.

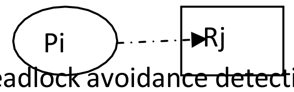
i) <u>Request edge:-</u> it indicates a process Pi requesting for resource Rj.



ii) <u>Allocation edge:-</u>it indicates a resource Rj is allocated to a process Pi.

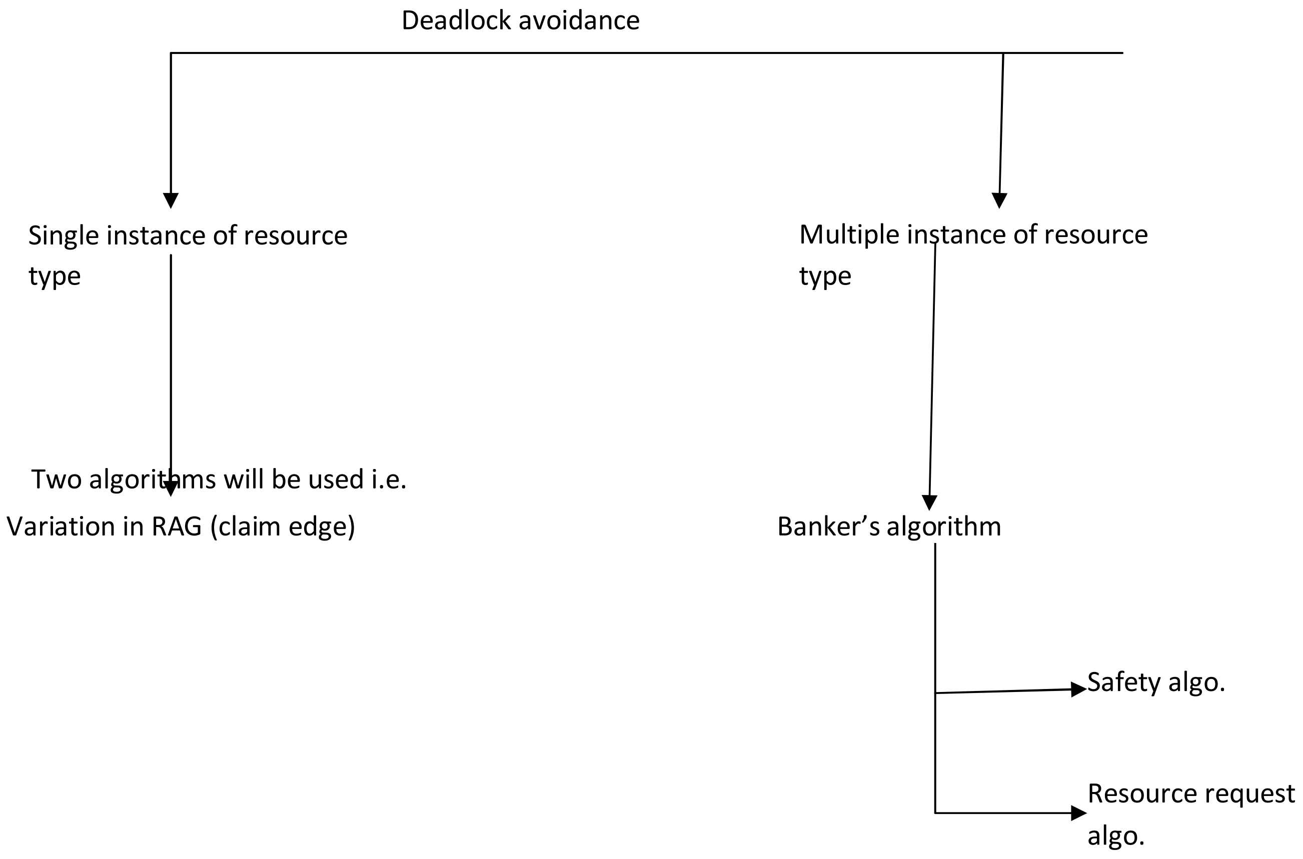


iii) <u>Claim edge:-</u> it indicates a process Pi may request the resource Rj in near future.



⇒ In deadlock avoidance detecting a cycle in the graph requires $0(n^2)$ operation.

⇒


Deadlock avoidance
Single instance of resource type
Multiple instance of resource type
Two algorithms will be used i.e.
Variation in RAG (claim edge)
Banker's algorithm
Safety algo.
Resource request algo.

ii) Banker's algorithm:-

⇒ Following data structure should be maintained, N = no. of processes.
M = no. of resources
Available = it is a vector of length m, it indicates no. of available resource in each time.
Max = It is a n x m matrix. It indicates maximum demand of each process.
Allocation = it is a n x m matrix. It indicates no. of resources of each type currently allocate to each process.
Need = m x n matrix. It indicates remaining need of resources for each process.
Need = max – allocation

⇒ Two algo will be used in banker's algo.

i) Safety algo
ii) Resource request algo.

i) Safety algorithm:-

⇒ Work – it is the vector of length m.

⇒ Finish – vector of length n. initialization work = available. Finish [i] = false for I = 0 to n – 1.

⇒ For find an I, s.t. both. Finish [i] == false and needi ≤ 1 work. If no such I exist, go to step 4.

⇒ Work = work + allocation I . finish [i] = true, go to step-2.

⇒ Finish[i] == true, when the system is in save state.

Steps to solve banker's algorithm:-

Steps-1:-calculate the need matrix. Need = max – allocation.

Step-2:- calculate the order of execution of processes (safe sequence).

i) Check need ≤available, if no. then the process must wait, otherwise the process will execute and after execution it should release the resources.
ii) Update the available, available = available + allocation

Step-3:-continue the process (1st iteration) to get a sequence.

Step-4:- if any process doesn't get a chance to execute in the 1st interaction, then continue to make them execute.

Note:- After performing the above operation if there is a same sequence, then the process is in safe state.

Q:

Process	Max			Allocation			Available			Need		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0	[3	3	2]	7	4	48
P1	3	2	2	2	0	0				1	2	2
P2	9	0	2	3	0	2				6	0	0
P3	2	2	2	2	1	1				0	1	1
P4	4	3	3	0	0	2				4	3	1

i) Calculate the need matrix.
ii) Is the system is in safe sate, if yes find the safe sequence.

A:- Step-1:- Calculate the need matrix

Need = max – allocation

Step-2:- Calculate the safe sequence.

Make 3 columns. Assume available = 1

Finish	i	work
False	0	[7 4 3] ≤ [3 3 2] No, P0 wait----

False →true	1	[1 2 2] ≤ [3 3 2]
		Yes, P1 will execute after execution it will release the resource but updating the availability
		Available = available + allocation
		Available = [3 3 2] + [2 0 0] = [5 3 2]

False	2	[6 0 0] ≤ [5 3 2] No, P2 will wait.
False →true	3	[0 1 1] ≤ [5 3 2] yes. P3 will execute Available = [5 3 2] + [2 1 1] = [7 4 3]
False →true	4	[4 3 1] ≤ [7 4 3] yes. P4 will execute. Available = [7 4 3] + [0 0 2] = [7 4 5]

Those processes have been left in the 1st interaction, again hey will continue the same process.

Finish	I	work
P0false→true	0	[7 4 3] ≤ [7 4 5]
P1 true	1	yes P0 will execute available = [745]+[010]=[755]
P2 False→true	2	[600]≤[755] Yes, P2 will execute Available = [755]+[302]=[10 5 7]
True	3	available = [10 5 7]
True	4	available = [10 5 7]

The order of execution of process =
<P1,P3,P4,P0,P2>
Safe sequence = <P1,P3,P4,P0,P2>

ii) <u>resource request algorithm:-</u>

Let request be the request vector for pi, request [j] = k. means Pi wants k instances of resources of Rj type.

1) If request i≤0 need, goto step-2 else error.
2) If request i≤ available, go to step-3 else pi must wait because resources are not available.
3) If the above two condition satisfy then available = available – requesti
   Need I = allocation I + request I
   Needi = need I + request I

   After the resource allocation if the process in a safe sequence, then the system is in safe state.

Q:-

Process	max	allocation	available
P0	7 5 3	0 1 0	[3 3 2]
P1	3 2 2	2 0 0	
P2	9 0 2	3 0 2	
P3	2 2 2	2 1 1	
P4	4 3 3	0 0 2	

what would happen in P1 request one additional resource of instance a type and 2 additional resource of instance C-type.

A: P1 is requesting [1 0 2]

P1 allocation will be updated as

[2 0 0]+[1 0 2] = [3 0 2]

Available will be updated as

[3 3 2] – [1 0 2] = [2 3 0]

The new snapshot is

Process	max	allocation	available	need
P0	7 5 3	0 1 0	[2 3 0]	7 4
P1	3 2 2	3 0 2		0 2
P2	9 0 2	3 0 2		6 0 0
P3	2 2 2	2 1 1		0 1 1
P4	4 3 3	0 0 2		4 3 1

Need = need – request
= [1 2 2] – [1 0 2]
= [0 2 0]

Finish	I	work
False	0	[7 4 3] ≤ [2 3 0] No, it will wait
False→true	1	[020] ≤[230] Yes, it will execute. Available = available + allocation ‘ = available = [230]+[302]=[532]
False	2	[600] ≤[532] No, it will wait.
False	3	[011] ≤[532] No, it will execute available = [532]+[211]=[743]
false	4	[431] ≤[230]

Deadlock detection:-

Deadlock can be detected in a system by following two approaches.

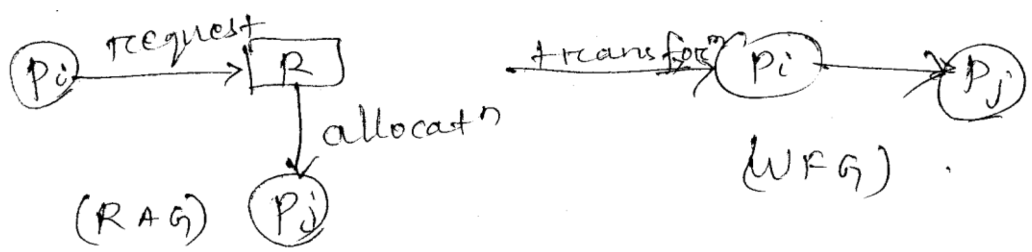
i) Single instance of resource type.
ii) Multiple instance of resource type.

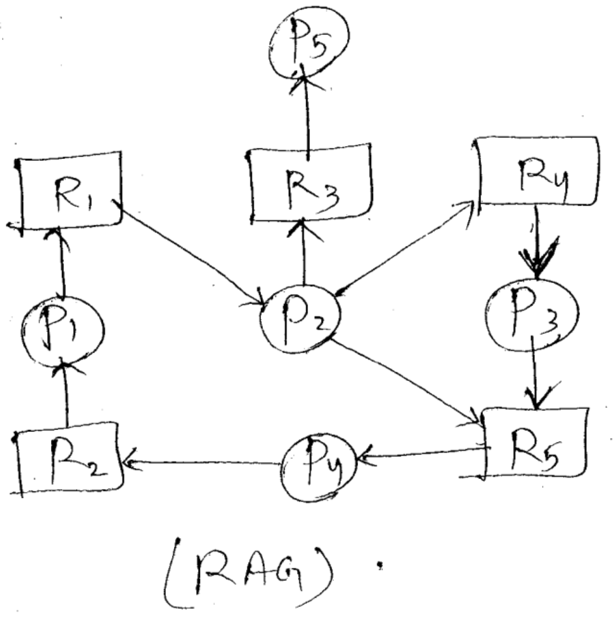
The deadlock detection algo. Determines whether the deadlock has occurred in the system or not.

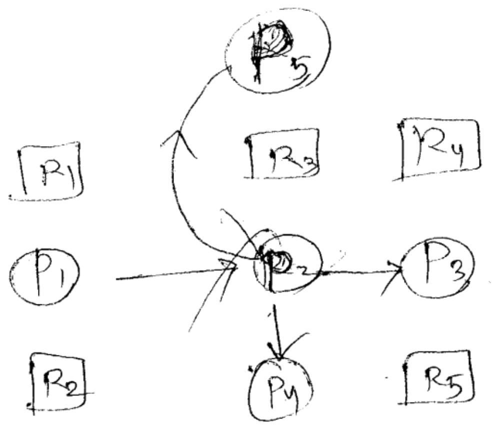
Single instance of resource type:-

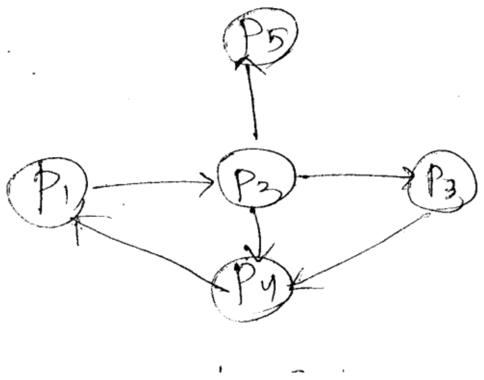
For single instance of resources deadlock can be detected in terms of a graph known as WFG for (wait for graph).

⇒ This graph is modified from of RAG.
⇒ In this graph there exist only the dependency among the processes.
⇒ Remove the resources by collapsing the corresponding edges from the RAG tends to WFG.
RAG→WFG









Multiple instance of resources type algorithm:-

⇒ This is a modified representation of bankers algo.

1) Available
2) Allocation
3) Request :- it is a nxm matrix. It indicates the current request of each process.

⇒ Step-1:- let work and finish are 2 vectors of length n & m. initialize work = available for I = 0 to n-1 if allocation ≠0, then finish [i] = false, others wise finish [i] = true.

⇒ Step-2:- find an index I such that both

i) Finish [i] == false
ii) Request I ≤ work

If no such I exist goto step-4

⇒ Step-3:- work = work + allocation I
Finish [i] = True, goto step-2

⇒ Step-4:- If finish [i] == false, then the system is in deadlock state.

⇒ Note:- this algorithm requires 0(mxn^2) operation to detect the deadlock state.

Q:-

Allocation	request	available/work
P0 010	000	000
P1 200	202	
P2 303	000	
P3 211	100	
P4 002	200	

Steps to solve:-

1) Check request ≤ work, if yes, the process will execute and make the updating of work, work= work + allocation.
2) If request ≤ work the process will wait and wait for the next interaction for execution.

A:-

Finish	I	work
False→	0	[000] ≤[000]

True		yes, process will execute Work = [010]
False	1	[202] ≤[010] No, process will wait
false→true	2	[000] ≤[010] yes, process will execute. Work = [010]+[303]=[313]
False → true	3	[100] ≤ [313] Yes work = [313]+[211]]=[524]
False → true	4	[200] ≤ [524] Yes work = [524]+[000]=[526]
False → true	1	[202] ≤ [526] Yes work = [526]+[200]=[726]

Safe sequence = <P0, P2, P3, P4, P1>

<u>Deadlock recovery:-</u>

the deadlock can be recovered from the system in 2 ways.

1) Process termination
2) Resource preemption.

1) <u>Process termination:-</u>
   ⇒ To eliminate the deadlock from the system we should abort some processes.
   ⇒ This can be done in 2 ways.
      i) Aborting all the deadlock process at the same time.
      ii) Abort one process at a time until the deadlock cycle is eliminated.
2) <u>Resource preemption:-</u>
   ⇒ To eliminate deadlock successively preempting some resources from the processes and give those resources to other processes until the deadlock cycle is broken.
      Note:- Because of the resource preemption 3 issues can be arise.
      i) Starvation
      ii) Releback
      iii) Selecting a vector

<u>Q:</u>

Process	allocation	max	available
	ABCD		
P0	0012	[0012]	[1520]
P1	1000	[1750]	
P2	1354	[2356]	
P3	0632	[0652]	
P4	0014	[0656]	

Answers the following qu. Using bankers algo.

i) What is the content of need matrix?
ii) If a request from process P1 arrives for 0420 can request be granted immediately.

<u>A:-</u>

<u>Need (max-allocation)</u>

[0000]

[0750]
[1002]
[0020]
[0642]

Finish	I	work
False →	0	[0000]≤[1520]
True		yes, P0 will execute
		Available = available+allocation
		=[1520]+[0012]=[1532]
False →	1	[0750]≤[1532]
		No., P1 will wait
False →	2	[1002]≤[1532]
True		yes, Available = [1532]+[1354]=[2886]
False →	3	[0020]≤[2886]
True		yes, Available = [2886]+[0632]=[2 14 11 8]
False →	4	[0643]≤[2 14 11 8]
True		yes, Available = [2 14 11 8]+[0014]=[2 14 12 12]
False →	1	[0750]≤[2 14 12 12]
True		yes, Available = [2 14 12 12]+[1000]=[12 14 12 12]

Safe state
,P0,P2,P3,P4,P1>

iii) Yes request will be granted
[0420] ≤[0750]
[0420] ≤[3141212]
Available = available-request
= [3 14 12 12]
- [0 4 2 0]
[3 10 10 12]
Allocation = allocation + request
= [1000]
+ [0420]
[1420]
Need = need – request
= [0750]
[0420]
[0330]
Safe state = <P0,P2,P3,P4,P1>
p1 0420 available = available+alllocation = [15 20]-0420]

Questions:

1. What is virtual memory?
2. What is cache memory?
3. Write characteristics of memory.

CHAPTER-6

FILE MANAGEMENT

A file is a collection of related information i.e. stored on secondary storage. Attributes of a file:-

i) Name of the file
ii) Identifier
iii) Type
iv) Location
v) Size of the file
vi) Protection
vii) Time, data, user identification.

Operation on files:-

i) Creating a file [f open()]
ii) Writing a file [f write()]
iii) Reading a file [F read ()]
iv) Repositioning with in a file [f seek ()]
v) Deleting a file
vi) Truncating a file

Locking on a file:-

⇒ File locks allows one process to lock a file and prevent other processed to access that file.
⇒ File lock are useful when a file is shared among diff. processes.
⇒ Two types of file locks
  i) Shared lock
  ii) Exclusive lock

i) Shared:-
Shared lock is otherwise known as read lock. Several process can acquire this lock concurrently.
Ex. This lock is implemented during the reading of a file.

ii) Exclusive:-
It is otherwise known as write lock only one process at a time can acquire this exclusive lock on a file.
Ex. During the write operation this lock is implemented.

Note:- OS may provide other locks named as mandatory, advisory file locking.

i) If a lock is mandatory then once a process acquires an exclusive lock. The OS will prevent any other process from accessing the lock file.
ii) If the lock is advisory, then OS will not prevent any other process from accessing the locked file.
iii) If the locking system is mandatory the OS ensures locking integrity.
iv) If a lock is advisory then it is upto the software developers to ensure that locks are appropriately acquired and released.
Ex: Mandatory lock windows OS. Advisory lock in unix OS.

<u>Access methods:-</u>

The information in the file can be accessed in following ways.

i) Sequential access method
ii) Direct access method.

i) In sequential access method information in the file is process in order one record after another
   Ex. Editors, compilers etc.
ii) Direct access method is otherwise named as relative access method. In this a file is made up of fixed length logical records that allow prog. To read and write records rapidly in no particular order.
   For direct access there is no restriction in the order of reading and writing.
   Ex: A file stored the information regarding air line reservation.

<u>Directory Structure:-</u>

The logical structure of a directory can be of following types.

i) Single level directory
ii) Two level directory
iii) Three structured directories.
iv) A cyclic graph directories.
v) General graph directory.

<u>Note:-</u>

Path name of a directory can be of 2 types

i) Absolute path
ii) Relative path

i) An absolute path and name begins at the root and follows a path down to the specification file giving the directory names on the path.
ii) A relative path defines path from the current directory.

Ex:-

Root
↓
Student
↓
Gec
↓
Dir-1
↓
File 1

Absolute path

root/student ----/file/

Relative

Dire1/File1

OS. has developed a logical storage chit called file.

⇒ The file system consist of 2 parts.

i) File
ii) Directory.

File :- Each file stored related data.

Directory:- It store the information about all the system .

File concept

- ⇒ A file is a collection of similar record with a common name.
- ⇒ A file has a structure according to its type.
  1) Text file:- A text file is a sequence of character organized into line.
  2) Source file:- A source file is a sequence of function.
  3) Object file:- It is sequence of by organize in 10 block.
  4) Executable file:- It is a series of pule to execute short programme.

File attributes

Name

It is a string of character file name is the only information which kept in human redebule form.

Identifier

This is a unique tag usually a no identify the file with in the file system.

Type

The name of the file split into two part.

1) Name
2) Extersion

The type is depend on extension of the file.

EXT (executable file)

OBJ (Object file)

Srs (Source file)

Txt (Text file)

This information is a pointer to device and to locate the location of file on the device.

SIZE:-

The current size of the file.

Ex. In bytes, kilo bytes and block

Time and date.

This is the information for the last modification for the file.

File operation

Operating system provide system call to create, write, read, delete, reposition truncate file.

Create:-

- ⇒ This step needed for create a file.
- ⇒ Space in the file system most be found in the file.
- ⇒ Entry for the file most be wait.

Write

To write a file a system.

- ⇒ Is specified of the name of file and information.

Read

To read a file system called is specify the name of the file and the location.

Delete

To delete a file, we search the appropriate directory where the file is exit.

Truncate

When a user wants to crage the content of file but keep its attribute with it.

Reposition

The file is reposition with the help of system call by finding the current file position.

- ⇒ Reposition with in the file doesn't need any actual. IO and this operation is called file seek.

File access method:-

File store information which information on the file can be access in several way.

1) Direct
2) Index or other
3) Sequential

Direct

File are known as logical record and store the information.

⇒ The direct access method is based on ckt model i.e. disk allow the random access method.

Ex. Suppose a CD consist of 10 song currently we are in sony no. 3 suppose we want to missing song no.9 then we directly go to song no.9 without restriction.

Index or other

This method contain a index of various block to find a record we most such the index file and then go to the desired file.

Sequential

This is the simplest method to access the file.

Ex. Suppose a file consist of 100 lines are were in 45 line and we want to access 75 line then we sequentially access 75 line then we sequentially access the no. from 45 ----------- 75, 75.

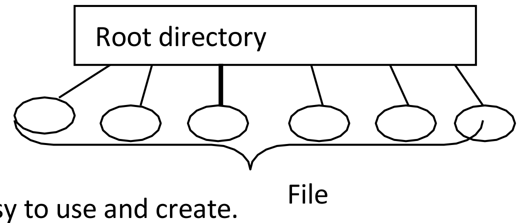
File system reliability:-

Reliability means to keep the file save from physical damage or data loss.

⇒ The data loss is happened due to system crash during file modification.

⇒ To save the file we need.

1. Backup:- copy the entire file system into a media (CD, Pen drive at a regular interval).
2. Raid disk (Redundant array of independent disk):- it is use to store the data of more than on disk.
3. Assembler:- an assembler is a program that store the index file of the computer.
4. Directory structure:- A directory is a collection of file which store millions of instruction it is of 3 types.
   a) Single:-this is the simplest directory structure all file are store under a single directory and the directory is known as root directory.

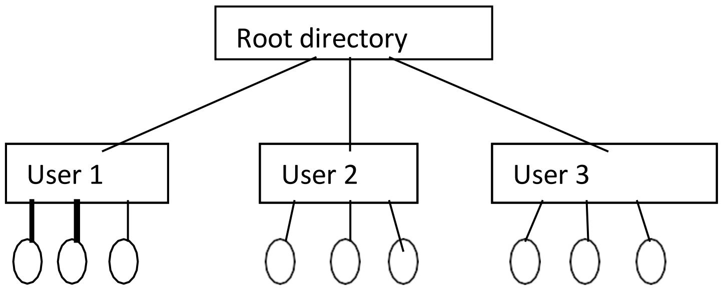


   Advantage:-easy to use and create.
   Disadvantage:- it is not use in multi user system that means when one a user name A then other user can not create the file with the same name A.

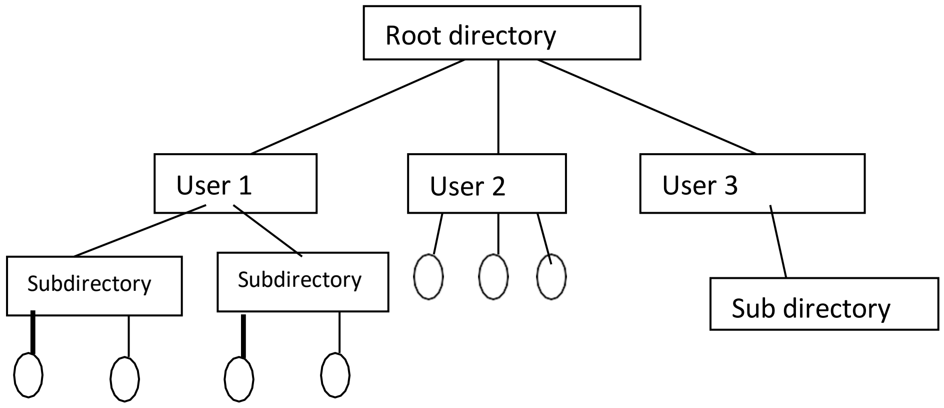
   b) Double:- to over come the disadvantages signal directory we use, two lable directory.

      ⇒ In two label directory structure each file is stored on its own user directory.

      ⇒ In two label directory we use 2 directory – 1) MFD (Master file directory)
      2) UFD (user file directory).



c) Tree:-



It is the most common directory structure in now a days.

⇒ file allocation:- there are 3 type of file allocation method.

1) Continuous
2) Link
3) Index

Continuous

⇒ In this method of a file allocation table is used.
⇒ File occupies a set of continuous block on the disk.
⇒ File allocation table contain file length starting block address and length of the file.
⇒ This method is suitable for sequential file.

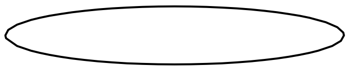
Advantage

Is avoid external fragmentation.

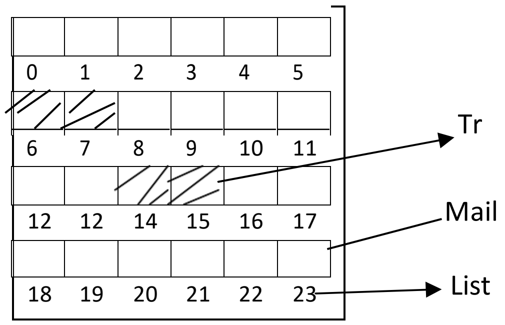
Disadvantage:-

It's accesing time is more.

File	Start	Length



Count	0	2
Tr	14	3
Max	19	6
List	28	4



linked:-

⇒ In this method is block contain a pointer to next 3 block.

⇒ Here we also use the file allocation table that contained file name, start block and end block.

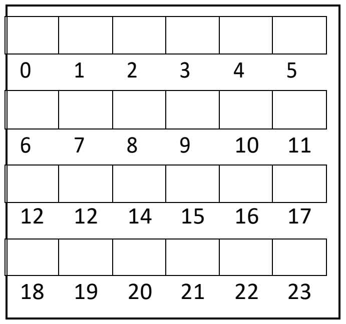
Advantage

No fragmentation occur.

Disadvantage

Pointer takes some memory.

File	Start	End
Jee	9	2
Dkl	10	4
Ddl	25	3

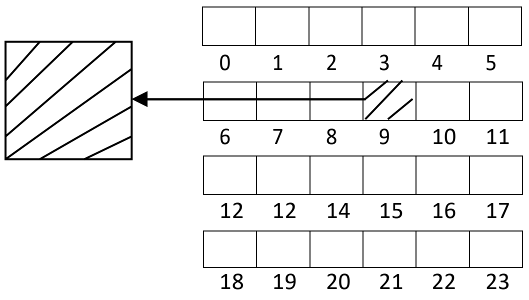


Index:-

In this method file allocation table contain a single entry for each file.

In index file method the block contained a index table that means all the linked block are stored on that index table.

File	Index File
CSE 2nd in WP DKL	9



Secondary storage management:-

⇒ Secondary storage device is used for storing all data and program.

⇒ Size of main memory is small to store all data and program.

⇒ It also lost the data when power OS loss. For this reason secondary storage device is used. Therefore proper management of disk is needed.

⇒ The O.S. is the responsibility to manage the secondary storage with the following activity. The O.S. maintain a true disk space list.

⇒ The free space list record the free disk block during the creation of file.

⇒ The O.S. 1st search the free space and compare that memory require by the file. If it is sufficient then the O.S. create that file.

⇒ There are 4 technic for free space.

1) Bit vector
2) Linked list
3) Counting
4) Grouping

Bit Vector:-

⇒ In this free space list it is implemented by a bit map or bit vector. Each block represented by 1.

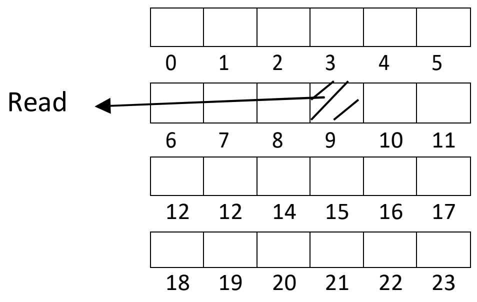
⇒ If the block is free the bit is 1 if the block is allocated then the bit is 0.
Ex. Let in a disk block. In a block number 2,3,4,5,8,9,10,11,12,13,17,18,25,28 & 27 and rest block are allocated. Then the bit vector is 001111001111100011000000.

Linked list:-

It is created by O.S. that store the free block address.

⇒ During any file creation the O.S. 1st search the 1st node.
Exl: Let in a disk block number 2,3,4,5,8,9,10,12,15,18,20,25,26,27 are free.
Then the linked list is.

File index



Counting:-

⇒ When a space is allocate through continuous memory allocation algorithm several continuous block may be allocated.

Grouping

⇒ There is a master block which stores the address of tree block.

Storage allocation

⇒ It means to allocate the secondary storage by the process.

Disk scheduling

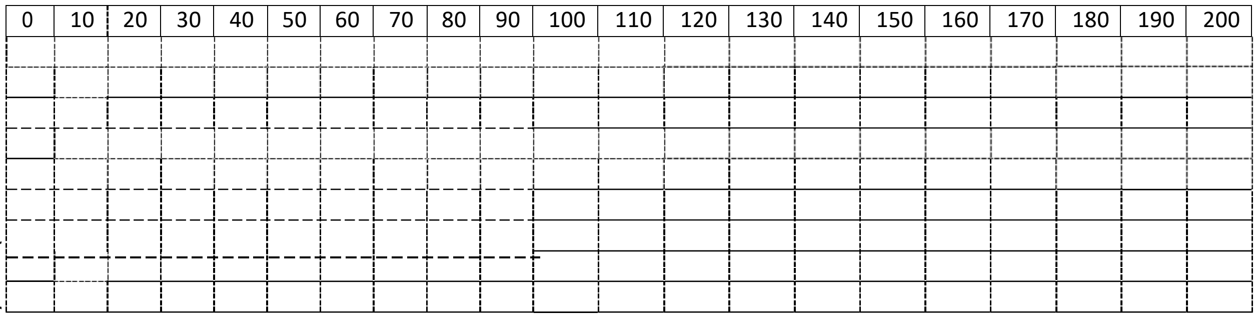
⇒ To create a file inside the block in the cylinder is known as disk scheduling.
⇒ Different disk scheduling technic are FLFS, SSTF, SSCan
⇒ The main purpose of

Seek time

Seek time is the time for the disk to move the head from one sector to desired sector.

Ex. A disk queue with the request for the I/O on cylinder number 98,183,37,122,14,124,65,67.

⇒ The head initially at cylinder 53.



Seek tim

There fc

File sha

⇒ File sharing is very desirable for user to want to reduce the effort.

1. *Multiple user*
2. *Remote file system*
3. *Client*
4. *Distributed system*

File protector

⇒ Where information kept in comp the want to keep it safe for damage (Reliable)
⇒ i.e. protection.

Types of access

Read-to read from a file

Write:-

Execute:- load the file into memory and execute it.

Append :- write new information at end of the file.

Delete = delete the file

List – list the name and attribute of the file.

Access control

⇒ the most common approach to protect the file is to identified to user. There 3 types of user.

1. Owner – the user book create the file.
2. Group – A set of user to need to shair the file and need to access all the file.
3. Universe – all other users in the system in the universe who use the system.

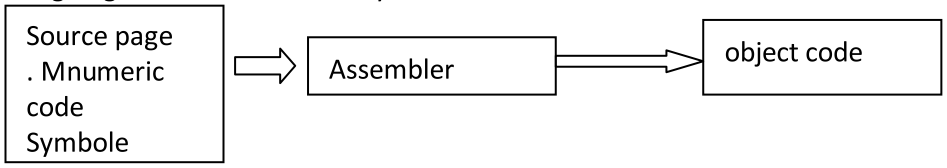
System programming

Assembler:- an assembler is a program that takes basic computer instruction and converts them into a patterns of bits that the computer's processor can use to perform its basic operation. Some people call the instruction assembler language or assembly language.

Function of Assembler

⇒ Translating uneronic option under to their machine language equivalent.

⇒ Assigning machine address to symbolic level.



Compiler

⇒ A compiler is computer program that transform source code written in a programming language into another computer language (The target language often having a binary form known as object code).

⇒ The most common reason for converting a source code is to create an executable program.

⇒ Generally computer known as translator.

Function of compiler

⇒ Compiler formulate the program at a time.

⇒ Its check the error.

Difference between compiler and interpreter

⇒ Compiler formulate the high level instruction into machine language. But the interpreter formulate the high level instruction into an intermediate code.

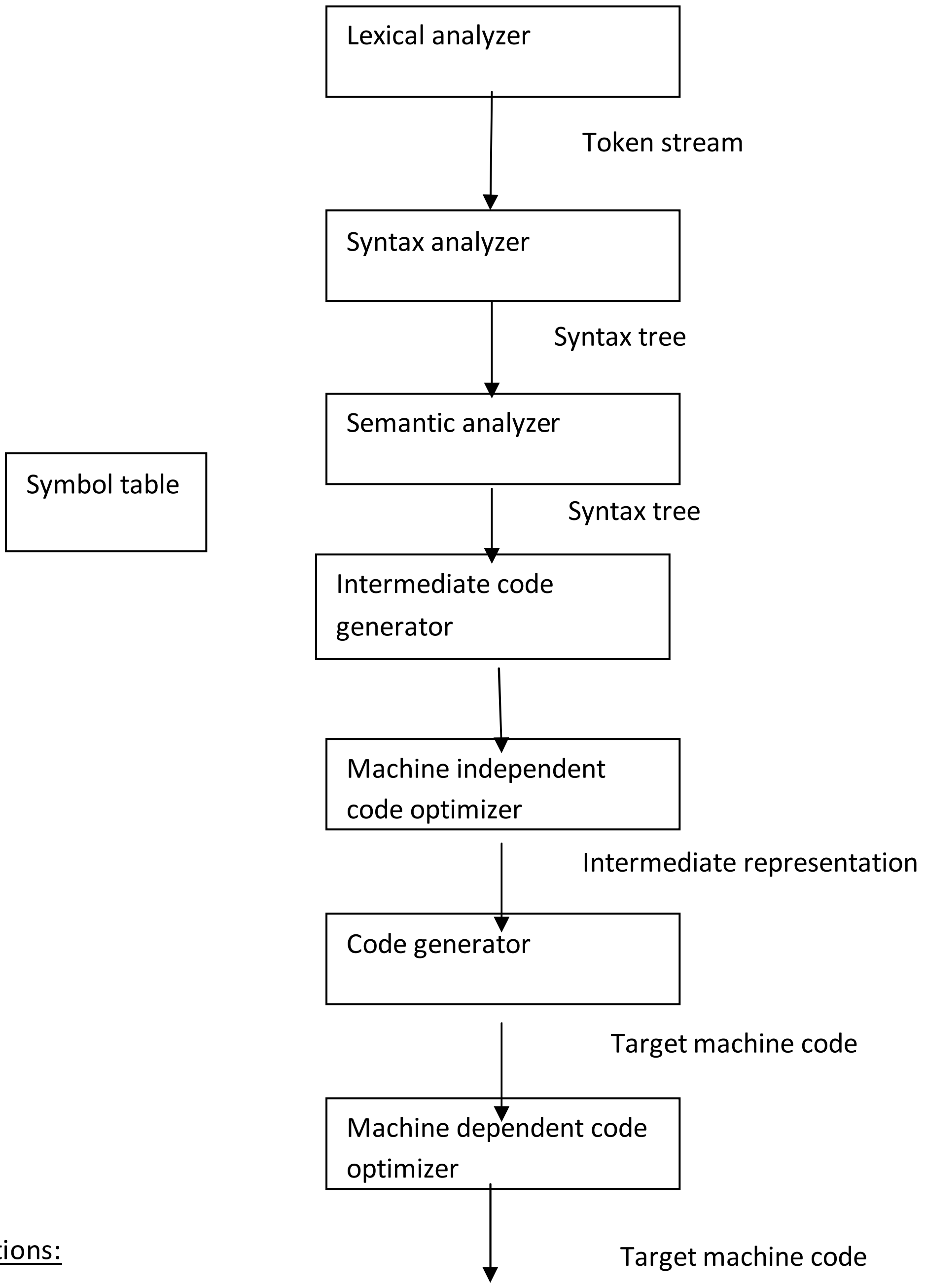
⇒ Compiler execute the entire program at a time.

⇒ But the interpreter execute each and every line individually.

⇒ Compiler reports the list of errors that are caused during the present of execution but the interpreter quit translating soon after, finding an error the progression of the other line of the program will be done after refining the error.

⇒ Autonomous executable file is generated by the compiler while interpreter is compulsory for an interpreter program.

7 phases of compiler



Questions:

1. What is DMA?
2. What is I/O channel architecture?