MAA OMWATI DEGREE COLLEGE HASSANPUR (PALWAL)

Notes

BCA 3rd Sem

Introduction to Operating System.

SYLLABUS OF B.C.A. III & IV SEMESTER effective from 2013-14

BCA-201 : Introduction to Operating System

External Marks: 80
Internal Marks: 20

Time: 3 hours

Note: Examiner will be required to set NINE questions in all. Question Number 1 will consist of total 8 parts (short-answer type questions) covering the entire syllabus and will carry 16 marks. In addition to the compulsory question there will be four units i.e. Unit-I to Unit-IV. Examiner will set two questions from each Unit of the syllabus and each question will carry 16 marks. Student will be required to attempt FIVE questions in all. Question Number 1 will be compulsory. In addition to compulsory question, student will have to attempt four more questions selecting one question from each Unit.

UNIT – I

Fundamentals of Operating system: Introduction to Operating System, its need and operating System services, Early systems, Structures - Simple Batch, Multi programmed, timeshared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems.

Process Management: Process concept, Operation on processes, Cooperating Processes, Threads, and Inter-process Communication.

UNIT-II

CPU Scheduling: Basic concepts, Scheduling criteria, Scheduling algorithms : FCFS, SJF, Round Robin & Queue Algorithms.

Deadlocks: Deadlock characterization, Methods for handling deadlocks, Banker's Algorithm.

UNIT-III

Memory Management: Logical versus Physical address space, Swapping, Contiguous allocation, Paging, Segmentation.

Virtual Memory: Demand paging, Performance of demand paging, Page replacement, Page replacement algorithms, Thrashing.

UNIT-IV

File management: File system Structure, Allocation methods: Contiguous allocation, Linked allocation, Indexed allocation, Free space management: Bit vector, Linked list, Grouping, Counting.

Device Management: Disk structure, Disk scheduling: FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK.

Suggested Readings

1. Abraham Silberschatz, Peter B. Galvin, " Operating System Concepts", Addison-Wesley publishing. Co., 7th. Ed., 2004.
2. Nutt Gary, "Operating Systems", Addison Wesley Publication, 2000.
3. Andrew S. Tannenbaum, "Modern Operating Systems", Pearson Education Asia, Second Edition, 2001.
4. William Stallings, "Operating Systems, "Internals and Design Principles", 4th Edition, PH, 2001.
5. Ekta Walia, "Operating Systems Concepts", Khanna Publishes, New Delhi, 2002.

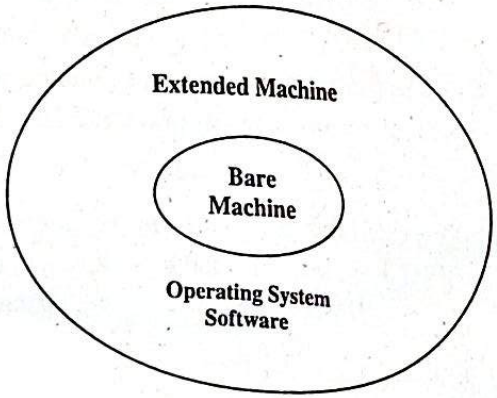
Note: Latest and additional good books may be suggested and added from time to time.

Q2.(a) Why operating system is called an Extended Machine and Resource Manager? Explain.
MDU BCA 2018, 2014

Ans. Operating System as an Extended Machine

A computer hardware is a bare machine. A bare machine is not the environment desired by most programmers. An operating system provides an environment for the execution of programs. It provides an environment where software and data make use of the hardware in the computer system.

An operating system is the most fundamental program that acts as an intermediate between a user of a computer and the computer hardware.



The user communicates with the operating system and does not directly interface with the hardware. The operating system effectively isolates the hardware from the user by insulating all the hardware.

Thus operating system can be regarded as a virtual machine standing between our programs and the bare hardware, thus providing a more convenient extended machine. Thus operating system seems to be as an extension of the bare machine and hence is an extended machine.

Operating System as a Resource Manager

An operating system is an essential component of a computer system. The operating system as a control program, controls and co-ordinates the use of hardware among various application programs of various users.
An operating system is a large collection of software, which manages resources of the computer system such as memory, processor, file system and input/output devices. It allocates the resources on the basis of the user's need and system capabilities.

As a Resource Manager/Resource allocator, the four major functions of the operating system are following:

1. Memory Management.
2. Processor Management.
3. Input/output and Peripheral Management.
4. File and software Management.

As a Resource Manager, the operating system must perform the following functions for each resource:

- ***Keep track of the resource.***
- ***Enforce policy that determines who gets what, when and how much.***

- *Allocate the resource to the job decided.*
- *Reclaim or deallocate the resource.*

Major Functions of Operating System as a Resource manager:

1. Memory Management Function

- *Keep track of main memory; what parts are in use and by whom? What parts are not in use i.e. free etc?*
- *In multiprogramming, decide which process gets memory, when it gets it, and how much.*
- *Allocate the memory when the process requests it and the policy decided above allows it.*
- *Deallocate the memory space, when the process no longer needs it or has been terminated.*

2. Processor (CPU) Management Function

- *Keep track of processor and the status of processes.*
- *Decide who will have a chance to use the processor; the job scheduler chooses from all the jobs submitted to the system. In multiprogramming, decide which process gets the processor, when, and how much.*
- *Allocate the CPU to a process.*
- *Reclaim/Deallocate the processor when process terminates or exceeds allowed amount of usage.*

3. Device Management Function

- *Keep track of all devices/peripherals.*
- *Decide which process gets the device, when and for how much time.*
- *Allocate the device and initiate the I/O operation.*
- *Deallocate the device. In most cases, the I/O terminates automatically.*

4. File Management Functions

- *Keep track of the file/information; its location, use, status etc.*
- *Decide who can use the resource; enforce protection requirements, and provide accessing routines.*
- *Allocate the resource (information/file) e.g. open a file.*
- *Deallocate the resource e.g. close a file.*

Q2.(c) What is Operating System? Discuss the services provided by an operating system.

MDU BCA 2016

OR

Define Operating System. What are the various services provided by an operating system to its users? Explain. **MDU BCA 2014**

OR

What do you understand by Operating System? What are the major functions performed by an Operating System.

MDU BCA 2012

Ans. Operating System

An operating system is a set of programs, which are used to manage the overall operations of a computer, in order to achieve maximum efficiency of the computer system.

It is a large collection of software, which manages resources of the computer system, such as memory, processor, file system and input/output devices.

It is the most important part of system software which schedules programs and controls the operations of the computer system. It is the program within a computer system which helps users to run their applications.

Computer system hardware is itself a complex system. If every programmer has to be concerned with how disk drive works, what could go wrong when reading a disk etc., writing a program is very difficult. To shield

programmers from the complexity of the hardware a layer of software on top of bare hardware is needed to manage all parts of the system. This layer of software is the operating system.

Thus ***"An Operating System is a set of programs that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware."***

Various services/ functions provided/performed by an Operating System

The operating system is designed with a mechanism to provide varied services/facilities to the users in a computer system.

The functions of an operating system may vary from one operating system to another. The main services provided by operating systems are the following:

1. User's friendly Interface
2. Resource manager
3. Accounting
4. Command Interpretation
5. Interrupt Handling
6. Protection
7. Fault Monitoring
8. Secondary storage management
9. Communication
10. Supervision

1. User's friendly Interface

Operating systems are designed and developed to optimize the man-machine capabilities. All its services intend to reduce the user's programming burden. The user need not write a complex set of commands to perform every task. User interacts with the computer through operating system.

2. Resource Manager

An operating system manages the resources of a computer system such as memory, CPU, file system and I/O devices etc. in order to attain an increased efficiency of a computer system. As a Resource Manager the four major functions of the operating system are following:

(a) Memory Management: Memory is an important resource that is managed by the operating system.

(b) Processor Management: Keeps the processor busy, scheduling of jobs permitting a number of users or application programs to share the CPU.

(c) Input/output and Peripheral Management: The speed of one I/O device differs from another and also from the speed of CPU. Operating system maintains a control in the sense that CPU is busy. Operating system allows the users to give input or get output by managing different I/O devices.

(d) File and software Management: The operating system allows the users to create, manipulate and organize files and directories, share data/files etc.

As a Resource Manager, the operating system performs the following functions for each resource:

(a) Keep track of the resource.

(b) Enforce policy that determines who gets which resource, when and how much/how long.

(c) Allocate the resource to the job decided.

(d) Reclaim or deallocate the resource.

3. Accounting

The accounting for the computer time used by a user is also prepared and maintained by the operating system in multi-user environment.

For accounting purposes logs of each user must be kept. It is also necessary to keep record of which user uses how much and what kinds of computer resources. The accounting data may be used for statistics or for the billing. It is also used to improve system efficiency.

4. Command Interpretation

The interpreter program residing in operating system facilitates the translation and execution of the commands given by the user in procedural language.

5. Interrupt Handling

An interrupt is a hardware/software generated change of flow within a system. The operating system directs an interrupt service routine and determines what action is to be taken.

6. Protection OR Security

Operating system keeps safe the important information for future and current use as well as protects from unauthorized access.

Protection involves ensuring that all access to system resources is controlled. In multi-process environment, it is possible that one process to interface with the other or with the operating system, so protection is required.

Security starts with each user having to authenticate to the system, usually by means of a password. External I/O devices must be also protected from invalid access attempts.

7. Fault monitoring OR Error detection

Error may occur in CPU, in I/O devices or in the memory hardware. The operating system constantly needs to be aware of possible errors. It should take the appropriate action to ensure correct and consistent computing.

If any kind of error occurs, the error handling routine in operating system terminate the particular process and immediately reports to the user to correct it.

8. Secondary storage management

Besides managing the main memory, operating system also performs the typical task of disk management. The operating system performs the following task with disk management:

- Disk scheduling.

- Storage allocation using compaction.
- Free-space management.

9. Communication

Data transfer between two processes is required for some time. The both processes are on the one computer or on different computer but connected through computer network. Operating system allows different running processes to communicate and handles deadlock situations.

10. Supervision

Operating system as a supervisor program:

- ***Controls the sharing of system hardware and software by several users.***
- ***Supplies common services for use by other software.***

Q2.(g) What do you mean by Simple Batch, Multi-Programmed, Real-time and Time-shared operating system. Explain in detail.

MDU BCA 2015

Ans. Simple Batch Operating System

In older days the computers were large systems run from a console. The common input devices were card readers and tape drives and output devices were line printers, tape drives and card punches.

The computer system did not directly interact with the users instead the computer users used to prepare a format that consisted of the programs, the data and some control information about the nature of the job and submitted it to the computer operator.

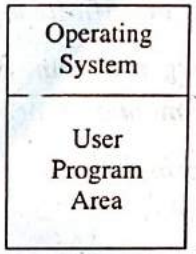
The job was usually in the form of punch cards. The process as a whole took a lot of time and was slow. To speed up the processing jobs with similar needs were batched together and were run through the computer as a group.

Thus, jobs with similar requirements were batched together and run through the computer as a group. The operating system handling a batch of programs is known as simple batch operating system.

In a batch processing system, a job is described by a sequence of control statements stored in a machine-readable form. This operating system relieves the user from the difficult task of loading the next program after the execution of the previous program is completed. It

must be remembered that a batch program operating system is a single program system because at a time only one program is being executed by the C.P.U.

Figure shows the memory layout for a simple batch system:



A simple batch operating system, thus normally Read a stream of separate jobs, each with its own control cards that predefine what the job does. When the job is complete, its output is usually printed.

In this operating system:

- ***Memory is usually divided into two parts. One part is fixed for containing operating system and the other part contains user programs to be executed. When one program is executed, another program is loaded into the same memory area.***
- ***The C.P.U. has to execute only one program at a time so the C.P.U. management also does not have problem.***

Disadvantages of simple batch operating system

Following are the disadvantages of simple batch operating system:

- *Simple batch operating system allows no interaction between users and executing programs.*
- *The turnaround time taken between job submission and job completion is very high. This time can be large from user standpoint.*
- *A job could enter an infinite loop.*
- *Difficult to debug program. A programmer cannot correct bugs the moment it occurs.*
- *Due to lack of protection scheme, one batch job can affect pending jobs.*

Multi-Programmed Operating System

Multiprogramming refers to a form of processing in which computer holds more than one program in the memory and executes them concurrently, i.e., it is a technique to execute number of programs simultaneously by a single processor.

In multiprogramming, numbers of processes reside in the main memory at a time and the operating system picks and begins executing one of the jobs. Any I/O wait by a process, will switch the CPU from that job to another job in the job pool, eliminating the CPU idle time. The operating system takes care of switching the CPU among the various user jobs. It also provides a suitable run-time environment and other support functions, so the jobs do not interfere with each other.

Multiprogrammed systems provide an environment in which the various system resources are utilized

effectively, but they do not provide for user interaction with the computer system.

Advantages of multi-programmed operating system

Following are the advantages of multi-programmed operating system:

- *CPU never sits idle, so it increases the CPU performance.*
- *Efficient memory utilization.*

Disadvantages of multi-programmed operating system

Following are the disadvantages of multi-programmed operating system:

- *CPU scheduling is required.*
- *To accommodate many jobs in memory, memory management is required.*
- *The user cannot interact with the job when it is being executed.*

Real time operating system

A real time operating system is one in which the input data has to be processed and the result produced within a given time period. Real time system is a special purpose operating system.

It is used in those environments where a large number of events external to computer systems, are accepted and processed in a short time or within specified time.

Real time operating system has well defined, fixed time constraints. In some real time tasks if the task is not completed within deadline time, results may be disastrous. Such tasks are Hard Real Time Tasks. In some other cases unsuccessful attempt requires rescheduling the task for completion. Such tasks may be termed as Soft Real Time Tasks.

For example, a computerized robot picking up something from a conveyor belt with the object moving. In cases, the robot is early or late, the object won't be there and thus resulting in an unsuccessful attempt though the robot moved to the right place.

Thus, real time operating system are very useful in places where a close watch is to be kept on the surroundings of the system and some corrective action is to be taken in case one specific thing happens.

Its main characteristics are:

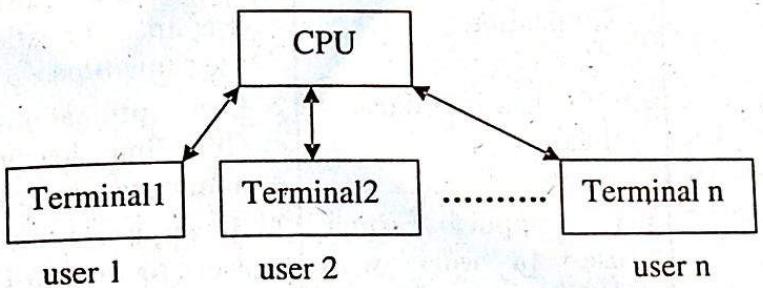
- *Fast Response time*
- *Feedback mechanism*
- *Reliability*
- *Predictability*

Such applications include:

- *Scientific experiments.*
- *Flight control*
- *Few Military applications.*
- *Industrial Control.*
- *Intensive Care Monitoring*

Time-sharing operating system

Time-sharing is a special case of multiprogramming operating system with a quick response time. It allows many users to simultaneously share the computer resources. It provides each user with a small portion of a time-shared computer.



In a time-shared system since each action or command take a very small fraction of time, only a little CPU time is needed for each user. As the CPU switches rapidly from one user to another user, each user is given impression that he has his own computer, whereas actually one computer is being shared among many users.

Most time sharing systems use time-slice (round robin) scheduling of CPU. Time sharing provides the advantages of quick response time. This leads to optimum utilization of resources. Since in time sharing systems, multiple programs co-exist in memory, hence memory management in such system provides for the protection and separation of user program.

Q4.(d) Explain the term Multi-Programming.
MDU BCA 2014

Ans. ***Multiprogramming is a technique to execute number of programs simultaneously by a single processor.***

In multiprogramming number of processes reside in main memory at a time and the operating system picks and begins executing one of the jobs in main memory.

Thus, multiprogramming refers to a form of processing in which computer holds more than one program in the memory and executes them concurrently.

In this, processor time is shared in such a manner that each program receives some attention for sometime, it seems as though all the programs are being run at the same time.

Multiprogramming operating systems as compared to Batch operating system are fairly better. Such operating systems are fairly sophisticated.

Multiprogramming increases CPU utilization by organizing a number of jobs such that CPU always has one to execute.

Advantages of multiprogramming

Following are the advantages of multiprogramming:

1. Increased CPU Utilization

Multiprogramming increases CPU utilization by organizing a number of jobs such that CPU always has one to execute.

2. Multiple Users

Multiprogramming also supports multiple users.

3. Increased Throughput

Throughput refers to the total number of programs executed over a fixed period of time. In multiprogramming, CPU is not waiting for I/O for the program it is executing, thus resulting in an increased throughput.

4. Shorter Turn Around Time

Turn around time for short jobs are greatly improved in multiprogramming.

5. Improved Memory Utilization

In multiprogramming, more than one programs reside in main memory. Thus memory is utilized to the optimum.

6. Increased Resource Utilization

In multiprogramming, a number of programs are actively competing for resources resulting in higher degree of resource utilization.

Q4.(b) Explain Parallel System. **MDU BCA 2016**

Ans. Multiprocessing operating system or the parallel system supports the use of more than one processor in close communication.

Thus, a system consisting of more than one processor that is tightly coupled (i.e., heavy sharing of resources like bus, clock, memory, I/O devices) is a parallel system.

A single CPU may be slow. So, a possible solution is to have many CPUs put parallel to execute a single given problem. This is parallel processing. These systems will increase the CPU throughput; reduce time required for job execution.

The advantages of such system are:

1. Increased Throughput

By increasing the number of processors, more work can be done/completed in a unit time.

2. Cost Savings

Such system shares the memory, buses, peripherals etc. Multiprocessor system thus saves money as compared to multiple single systems. Also, if a number of programs are to operate on the same data, it is cheaper to store that data on one single disk & shared by all processors than to have many copies of the same data.

3. Increased Reliability

In this system, the workload is distributed among several processors, which results in increased reliability. If one processor fails, its failure may slightly slow down the speed of the system but won't fail the system.

Q4.(c) Explain Distributed System. MDU BCA 2016

OR

What is distributed system? Explain the advantages of distributed system. What are the difficulties in implementing distributed system?

Ans. Distributed System

It comprises of a large central computer to which a large number of remote terminals are attached.

A distributed system is one that looks like an ordinary centralized operating system that runs on multiple independent CPUs.

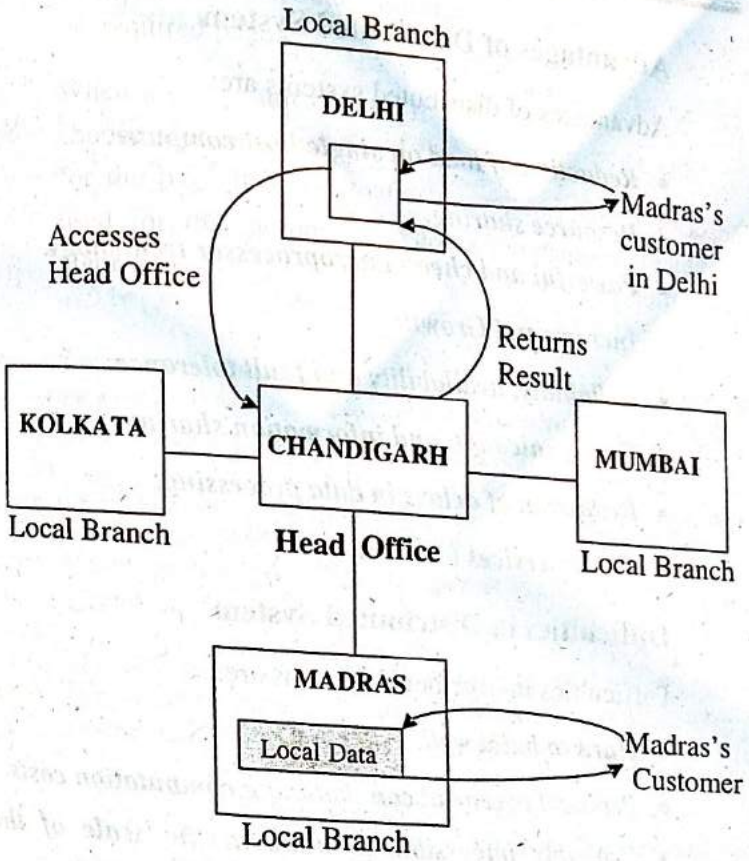
It is collection of processors that do not share memory. Each processor has its own local memory.

Distributed system runs on and controls the resources of multiple machines. It provides resource sharing across the boundaries of a single computer system. It looks to users like a single machine. It owns the whole network and makes it look like a virtual multiprocessor.

A distributed system implies a number of CPUs/computing systems put together by a high speed network.

The use of multiple processors is invisible to the user i.e. the users are not aware of where their programs are being run or where there files are residing. These are handled automatically and efficiently by the operating system.

For example, a Bank having its branches throughout the country. The bank maintains/handles the local data at the local branch and copy of the data of all the branches at the Head Office.



With distributed processing, related data at all locations/terminals is synchronized, updated simultaneously.

This operating system allows programs to run on several processors at the same time, thus requiring more complex processor scheduling algorithms in order to achieve maximum utilization of CPU's time.

The distributed operating system allows the user to access remote resources in the same manner as local resources as they do.

Multiprocessing	Multiprogramming
3. Multiprocessing refers to processing of more than one process by multiple CPUs at one time.	3. Multiprogramming refers to keeping several programs in main memory at the same time and execute them concurrently by the same computer.
4. It allows parallel processing.	4. There is nonparallel processing, here context switching takes place.
5. Time taken to process the jobs is less.	5. Time taken to process the jobs is more.
6. Such systems are more expensive.	6. Such systems are less expensive.

Q5.(a) Explain Process. **MDU BCA 2017**

OR

What short note on process. **MDU BCA 2013**

Ans. ***A program in execution is called a process. A process can also be defined as an instance of a program in execution.***

Process is the entity that can be assigned to and executed on a processor. It is the unit of work in a modern time-sharing operating system. A process is more than the program code. It includes the current activity as represented by the value of the program counter (PC) and the contents of the processor's registers.

e.g. (i) In UNIX operating system, shell or command interpreter is also a process which performs the task of listing to whatever is typed on a terminal. As we run a command, shell runs that command as a separate process.

(ii) In a time-shared system, computer memory stores multiple users' programs. Each program is given a fraction of CPU time. As CPU begins executing a program, a process is created. When one process stops running after taking its share of CPU time, another process starts.

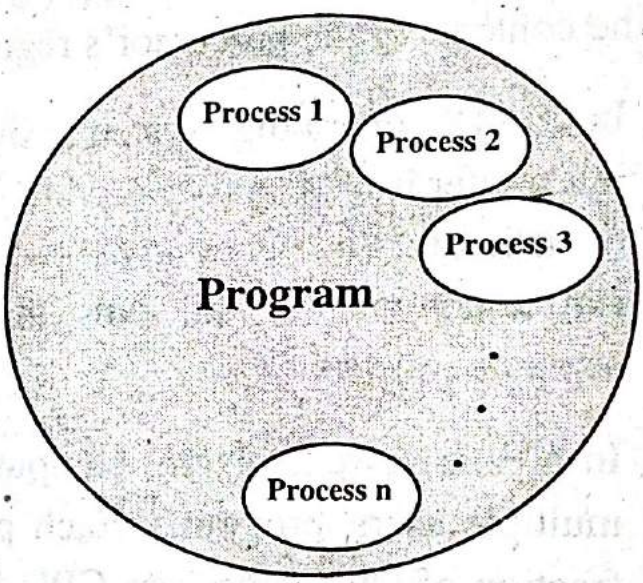
A 'process' is not the same as 'program', it is infact a program in execution. A program is a passive entity. For example, a program file on the disk. A process on the other hand is an active entity. A process is a smallest unit of work that is scheduled by operating system. A

process is **job** of batch system or **task** of time-sharing system.

The term 'process' was first used by the designers of the MULTICS in 1960's, since then, the term process, used somewhat interchangeably with 'task' or 'job'.

A process is running piece of code. It is that entity to which processors are assigned. It contains the program code and its current activity. A process is a running program with which a particular set of data is associated.

A program may consist of several processes.



Thus there may be a situation when more processes may be associated with the same program but they are considered for separate execution processes.

Process consists of three components:

- ***An executable program.***
- ***Associated data needed by the program and***
- ***Execution context of the program.***

Q5.(b) Explain the term operation on process.

MDU BCA 2014

OR

What are the operations defined on a process? Explain.

Ans. Operations on a Process

Main operations that can be performed on a process are:

1. Process creation
2. Process execution
3. Process termination

1. Process creation

First the process is 'created'. Creating a process involves the following operations:

- Name the process
- Insert it in the system's list of known processes
- Create the process control block (PCB) etc.

A process may generate one or more new processes during execution. The creating process is called the parent process and the created process is called the child process. Only one parent is needed to create a child. Each of the children processes may create other processes. Thus such creation forms a hierarchical process structure that is a tree of processes.

There are some common events that lead to the creation of a process. In a batch environment, a process is created in response to the submission of a job. In an interactive

environment a process is created when a new user attempts to logon. It is the operating system which is responsible for the creation of the new process. On behalf of an application also the operating system can create a process. As a user requests for a file to be printed, the operating system creates a process that will manage the printing.

When a process is created, it requires some parameters. These are priority, level of privilege, requirement of memory, access right, memory protection etc. Process will need certain resources, such as CPU time, memory, files and I/O devices to complete the operation. When process creates a subprocess, that subprocess may obtain its resources directly from the operating system, otherwise it uses the resources of parent process.

2. Program execution

After a process is created, process is ready for execution. Depending on operating system policy, the newly created process may inherit its resources from its parent or its own new resources from the operating system. Some initialization data may also be passed from the parent to the child process.

When a process creates a new process, two possibilities exist in terms of execution. These are:

- ***The parent continues to execute concurrently with its children.***
- ***The parent waits until some or all of its children have terminated.***

3. Process termination

Process termination implies that the process executes its last statement and asks the operating system to delete it or exit. As a result, the process's resources are de-allocated by operating system. DELETE system call is used for terminating a process.

Following are the reasons for terminating the child process by parent process:

- ***Child has exceeded its usage of some of the resources that it has been allocated.***
- ***The task given to the child is no longer required.***
- ***Operating system does not allow a child to continue if its parent is exiting.***

The process termination can be done in following conditions:

- ***Normal exit,***
- ***Error exit,***
- ***Fatal error,***
- ***Killed by another process.***

Other operations

In addition to the above operations of the process, other different operations performed on a process are the following:

- Destroy a process

Q6.(e) Explain the term co-operating processes.

MDU BCA 2014, 2013

Ans. Processes executing concurrently in the operating system can be classified into two types based on their interaction with the other processes. The two types are independent process and co-operating process.

A process is independent if it cannot affect or be affected by the other processes executing in the system. A process that does not share data with any other process is independent.

A process is co-operating if it can affect or be affected by the other processes executing in the system.

A co-operating process is one which shares some information or communicates with other processes in order to speed up the computation time.

A co-operating process can affect the state of other processes by sharing memory, sending signals etc.

Advantages of co-operating processes

Co-operating processes have the following advantages:

1. Information sharing
2. Computational speed up
3. Modularity
4. Convenience

1. Information sharing

Several users may be interested in the same piece of information. For example, a shared file. Operating

system provides an environment to allow concurrent access to such information.

2. Computational speed up

Some problems can be solved quickly by dividing the process into smaller/subtasks which can be executed in parallel on several processors.

3. Modularity

Modularity implies dividing the system functions into separate processes or threads. Thus the solution of a problem is structured in a modular function. These parts with well defined interfaces run in parallel.

4. Convenience

An individual user may have many tasks to work on at one time. For example the user may be editing, printing and compiling in parallel.

Q6.(a) Explain threads and their uses.

MDU BCA 2017, 2016

OR

Explain threads in detail.

MDU BCA 2015, 2013

Ans. Thread

A thread is light weight process which is very useful for the efficient utilization of the CPU.

Threads are very much similar to process but unlike the nature of the processes threads are not independent of one another. A thread belongs to only one process and they exist only internal to a process. Threads are very much useful in multiprocessing environment whereas processes are very much useful in multiprogramming environment. Threads are not independent of one another.

A thread is a basic unit of CPU utilization, it consists of:

- program counter
- register set
- stack space

The operating system does not need to create a new memory map for a new thread as it does for a process. Threads provide speed and sharing to related threads. A thread shares with its peer threads its:

- code section
- data section
- operating system resources

In a multiple threaded task, while one server thread is blocked and waiting, a second thread in the same task can run.

Uses/Advantages of threads

Threads have the following uses/advantages:

1. ***A thread is very useful for the efficient utilization of the CPU.***
2. ***With a multithreaded application, the scheduler can schedule different threads to run in parallel on different cores or processors.***
3. ***Threading also makes certain types of programming easy. The same global and static variables can be read and written among all threads in a process.***
4. ***A thread takes less time to terminate.***
5. ***Switching between threads within the same process is much faster.***
6. ***Creation of a new thread within the process is quick.***
7. ***Efficient communication between threads is possible.***
8. ***Cooperation of multiple threads within the same process enables higher throughput and improved performance.***

Q6.(c) What do you mean by inter process communication? Explain with examples.

IGU BCA 2018

OR

Explain inter-process communication.

MDU BCA 2016, 2015

Ans. Inter-process communication

In concurrent environment processes may need to communicate with other co-existing processes. This is called Inter Process Communication (IPC).

For example, if a user_process wants to read from a file, it must tell the file_process what it wants; then the file_process tells the disk_process to read the desired block.

A mechanism through which data is shared among the processes in the system is referred to as Inter-process communication. A set of functions is required for the communication of a process with each other. Communication between processes is needed in a well-structured way, not using interrupts.

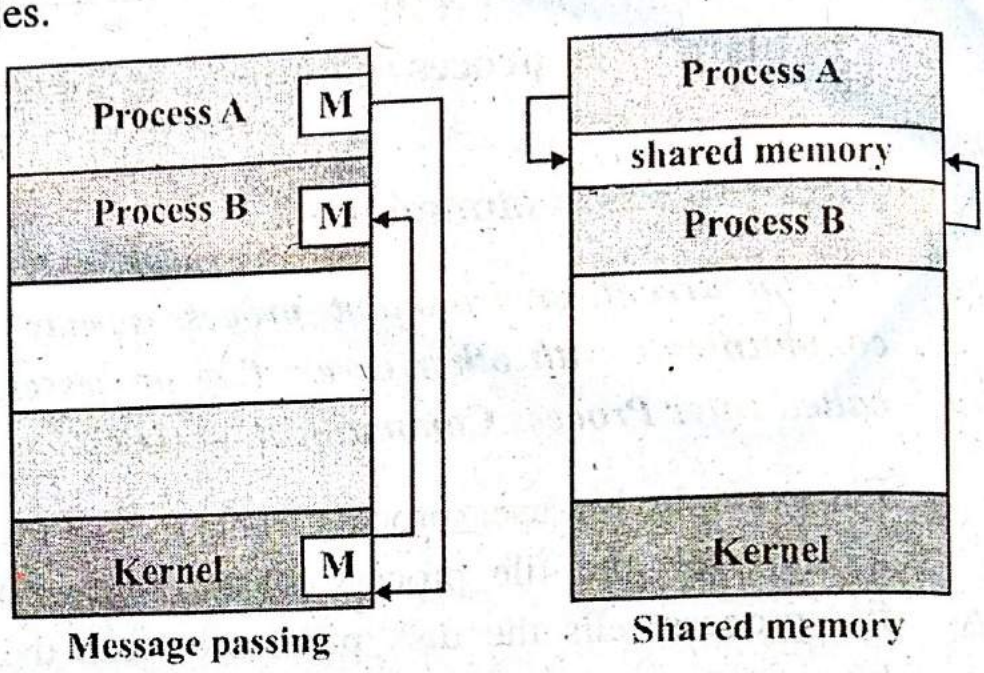
Two main communication schemes/techniques to implement inter-process communication are:

1. Shared memory
2. Message passing

1. Shared Memory System

In this, processes communicate via some shared variables. The processes exchange information through the use of these shared variables.

Here the operating system provides only the memory, other things like communication etc. have to be done by the programmer. Signals & semaphores perform specific roles.



2. Message Passing System

In the message system, messages are exchanged among the processes. Here communication etc. is done by operating system itself. Also communication occurs without the need to resort to shared variables.

This message passing facility provides two operations:

- Send (message)
- Receive (message)

In order to communicate, processes P1 and P2 must Send and Receive messages from each other. Messages sent by a process can be of any of these varieties:

- Fixed sized.
- Variable sized.
- Typed messages.

Q7.(b) What is scheduling? Discuss in detail the different levels of scheduling. MDU BCA 2013

Ans. Scheduling

Scheduling is a fundamental operating system function that controls the order in which the work to be done is completed.

The primary objective of scheduling is to optimize system performance in accordance with criteria. To maximize CPU utilization, some process should always be running. A scheduler selects a process from among the ready processes to execute on the CPU.

The three levels for scheduling are:

1. Long term scheduling
2. Medium term scheduling
3. Short term scheduling

1. Long term scheduling

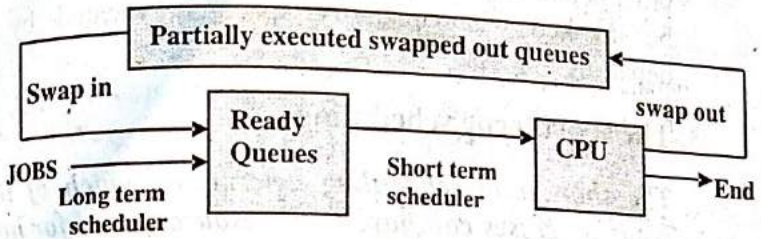
Long term scheduling is performed when a new process is created. This is a decision to add a new process to the set of processes that are currently active.

Long term scheduling determines which programs are admitted to the system for execution and when, and which ones should be exited.

It is also called job scheduling. There are always a number of jobs to be executed by the CPU. These jobs are stored in large storage devices like disk for later processing.

The long-term scheduler selects jobs from the secondary storage pool and loads them into memory. These processes in the memory are ready for execution.

The short-term scheduler selects jobs from the ready queue for assigning one of them to the CPU.



2. Medium term scheduling

Medium term scheduling determines when processes are to be suspended and resumed.

During execution, a process may be suspended for some I/O operations. This process can't progress towards completion until the related suspending condition is fulfilled. So it is beneficial to remove them from main memory to make space for other processes.

Thus, these suspended processes should be removed from the main memory and saved on secondary storage devices in order to utilize the space for other processes. These processes can be reintroduced into memory and continued where it was left earlier at some later time.

This shifting of suspended process from main memory to a secondary device is known as swapping out or rolling out.

Medium term scheduling is a part of the swapping function. This is a decision to add a process to those that are atleast partially in main memory and therefore available for execution. It is the medium term scheduler which is responsible for swapping in and swapping out the process(es).

As soon as the suspending condition is fulfilled, the medium term scheduler gets activated to swap in the process and make it ready for CPU. This scheduler may be invoked when memory space is vacated by a departing process.

3. The short term Scheduling

The short term scheduling determines which of the ready processes can have CPU resources, and for how long.

The short-term scheduler allocates processes in the ready queue to CPU for immediate processing. The goal of the short-term scheduler is to maximize CPU utilization.

The short term scheduler is more frequent as compared to other schedulers. Both long term scheduler and medium term scheduler prepare workload for the short-term scheduler.

The short-term scheduler must be very fast to improve system performance.

Thus, short term scheduling is the actual decision of which ready process to execute next.

Q7.(c) What do you mean by scheduling? Explain various scheduling algorithms with example.

MDU BCA 2016

OR

What is CPU scheduling? Explain the scheduling criteria. Also explain the scheduling algorithms in detail. **MDU BCA 2015**

OR

Discuss the following process scheduling algorithms with the help of examples:

1. **FCFS (First-Come-First-Served)** **MDU BCA 2013**
2. **SJF (Shortest-Job-First)** **MDU BCA 2013**
3. **RR (Round Robin)** **MDU BCA 2013**
4. **Priority based scheduling**
5. **Multiple-level-Queue (MLQ) scheduling** **MDU BCA 2013**
6. **Multiple level queues with Feedback scheduling (MLQFB)**

Ans. ***Scheduling is a fundamental operating system function that controls the order in which the work to be done is completed.***

There are several scheduling algorithms, which decide the processes in the ready queue to be allocated to the CPU.

A scheduling is said to be non-pre-emptive if once a process has been assigned to the CPU, the CPU can not be taken away from that process. A scheduling is said to be pre-emptive if the CPU can be taken away in between execution.

Scheduling Criteria

To choose a particular CPU scheduling algorithm for a particular situation, properties of the various algorithms must be considered. The characteristics used for comparison can make a substantial difference in the determination of the best algorithm. The scheduling criteria are listed below:

1. CPU Utilization
2. Throughput
3. Response time
4. Turnaround time
5. Waiting time

1. CPU Utilization

The main aim is to keep the CPU as busy as possible. The load on the system affects the level of utilization that can be achieved. CPU utilization may range from 0% to 100%. On large and expensive system, i.e., time shared system, CPU utilization may be the primary consideration.

2. Throughput

Throughput refers to the amount of work completed in a unit of time. In other words throughput is the number of jobs or processes executed in a unit of time.

The scheduling algorithm must aim at maximizing the number of jobs processed per time unit.

3. Response time

Response time is the amount of time it takes to start responding the request. A scheduler must aim at minimizing response time for interactive users.

4. Turnaround time

Turnaround time is the time between the moment of submission of a job/process and the time of its completion. Thus how long it takes to execute a process, is also an important criterion.

5. Waiting time

It is the amount of time a job spends in waiting for resource allocation when several jobs are competing in multiprogramming system. The aim should be to minimize the waiting time.

Scheduling Algorithms

Main scheduling algorithms are:

1. First-Come-First-Served (FCFS) scheduling
2. Shortest-Job-First (SJF) scheduling
3. Round Robin Scheduling
4. Priority based scheduling
5. Multiple-level-Queue (MLQ) scheduling
6. Multiple level queues with Feedback scheduling (MLQFB)

1. First-Come-First-Served (FCFS) scheduling

This is one of the simplest scheduling algorithms. A process that requests for the CPU first is allocated the CPU first. Hence the name first come first served or FIFO i.e. first-in-first-out.

In it a FIFO (First-in-First-out) queue is maintained. The processes are assigned to CPU in the order of their arrival.

When the CPU is free, it is allocated to the process which is at front of the queue. This scheduling is non pre-emptive i.e. once a process has the CPU, it runs to completion. Consider the following example of two processes.

Process	Execution time
P1	18
P2	2

If the processes arrive in the order P1, P2.

Waiting time for Process P1 = 0 unit time.

Waiting time for Process P2 = 18 units time.

Average waiting time (0+18)/2 = 9 units time.

However, if the same processes arrive in P2, P1 order

Average waiting time = (0+2)/2 = 1 units time.

Thus this example shows that short jobs may be delayed in FCFS scheduling algorithm.

If processes have same CPU time, FCFS is used.

Advantages of First-come-First-served algorithm

Advantages of first-come-first-served algorithm are:

- *It is simple to understand.*
- *It is suitable for batch systems.*

Limitations of First-come-First-served algorithm

Limitations of first-come-first-served algorithm are:

- *Non pre-emption usually results in poor performance.*

- *Short jobs may suffer long delays when CPU has been allocated to longer jobs.*
- *This algorithm is not good for time sharing systems where it is important that each user get a share of the CPU at regular intervals.*

2. Shortest-Job-First (SJF) scheduling

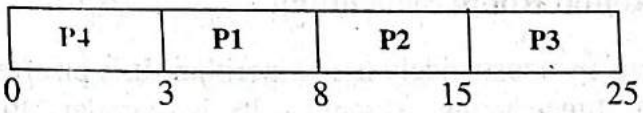
In this scheduling algorithm a process is selected on the basis of its shortest run time.

Among the processes in the ready queue CPU is always assigned to the process with least CPU burst time.

Consider the following example:

Process	Execution time
P1	5
P2	7
P3	10
P4	3

Using shortest job first scheduling, these processes would be in the P4, P1, P2, P3 order.



Waiting time for Process P4 = 0 unit time.

Waiting time for Process P1 = 3 units time.

Waiting time for Process P2 = 8 unit time.

Waiting time for Process P3 = 15 units time.

Average waiting time= (0+3+8+15)/4 = 26/4 = 6.5 units time.

Using the FCFS scheduling, the average waiting time will be = (0+5+12+22)/4 =39/4 = 9.75 units of time.

Thus SJF scheduling minimises the average waiting time of a given set of processes.

If there are two processes with same CPU burst requirement, one which arrived first will be served first.

This scheduling works good when the exact future execution times of jobs or processes are known at the time of scheduling.

Advantages of Shortest-Job-First scheduling

Advantages of shortest-job-first algorithm are:

- *It reduces the average waiting time to the minimum.*
- *Good for non-interactive systems.*

Limitations of Shortest-Job-First scheduling

Limitations of shortest-job-first algorithm are:

- *In this scheduling, estimating future process behavior is not always possible.*

3. Round Robin Scheduling

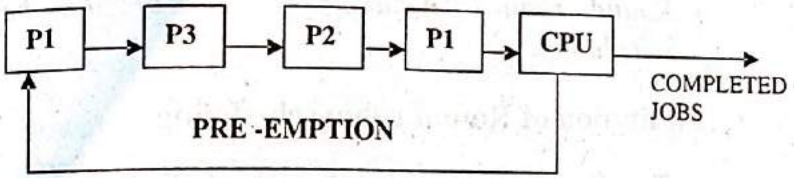
It is the most widely used algorithm. It is primarily used in time-sharing systems. It is similar to FCFS scheduling, but pre-emption is added to switch between processes. This aims at providing equal opportunity to all programs.

Pre-emption takes place after a fixed interval of time called time quantum or time slice.

Each process is given a limited small amount of CPU time in milliseconds. As one time slice is over, the CPU

is given to the next job waiting in the ready queue. A process may either complete its execution in a single time slice or may require more CPU time slices.

Thus the pre-empted process which requires more CPU time for its completion is placed at the back of the ready queue.



Small process may be executed in a single time-slice while long processes may require several time slices.

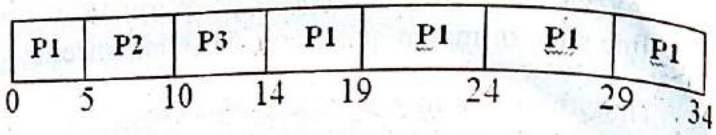
For example there are 3 processes: P1, P2, P3 which require the following CPU time.

Process	Execution time
P1	25
P2	5
P3	4

Let a time slice be of 5 units of time. Job P1 gets the first 5 units of time. After the first time slice is over, it is pre-empted and the CPU is given to the next job P2.

Since P2 requires only 5 units of time, it is completed in its time slice. Then CPU switches to P3, which needs only 4 units of time. P3 terminates before its time slice ends. Now each process has received one time slice. The CPU is again given to P1 for an additional time-slice.

Thus the resulting round robin scheduling is:



Advantage of Round robin scheduling

- *Round robin scheduling is one of the best scheduling.*

Limitation of Round robin scheduling

- *Too large or too small size of time quantum degrades the performance.*

4. Priority based scheduling

In this scheduling, each process is assigned a priority. The CPU is given to the process in the ready queue with the highest priority.

If two or more processes have equal priority they are scheduled in FCFS manner.

Priority scheduling may be pre-emptive or non pre-emptive. In priority based pre-emptive scheduling, pre-emption is done on the basis of priority. As a process arrives in the ready queue, its priority is compared with the priority of the process running currently.

If the priority of the newly arrived process is more than the priority of the currently running process, CPU will be allocated to the new process immediately.

In priority based non-pre-emptive scheduling, higher priority job is added to the front of the ready queue. The priority of the new process will be considered only after the execution of the currently running process.

Limitations of Priority based scheduling

Limitations of Priority based scheduling are:

- *A major problem that arises in this scheduling is indefinite blocking or starvation of a low priority process by a high priority process. This means that there is no certainty of completion of a low priority process within a finite time.*
- *Pre-emptive priority scheduling leads to starvation.*

However, a solution to this problem is provided by developing a mechanism of aging priority.

Aging is a technique in which the low priority of processes waiting for a long time is gradually increased. Thus the older processes attain high priority and can be completed in a finite time.

5. Multiple-level-Queue (MLQ) scheduling

This scheduling algorithm is designed for situations in which processes can be classified into different groups.

For example, a common division is made between interactive processes and batch processes.

Thus a multi queue-scheduling algorithm partitions the ready queue into separate queues. Processes are permanently assigned to a particular queue. Each queue has its own scheduling algorithm. The interactive queue is scheduled by a round-robin algorithm while batch queue follows FCFS.

Advantage of Multiple-level-Queue scheduling

Advantage of Multiple-level-Queue scheduling are:

- *Low scheduling overhead.*

Disadvantages of Multiple-level-Queue scheduling

Disadvantage of Multiple-level-Queue scheduling are:

- *Inflexible.*
- *Processes can never change their queues and low priority jobs may have to starve for the CPU if the other higher priority queues never become empty.*

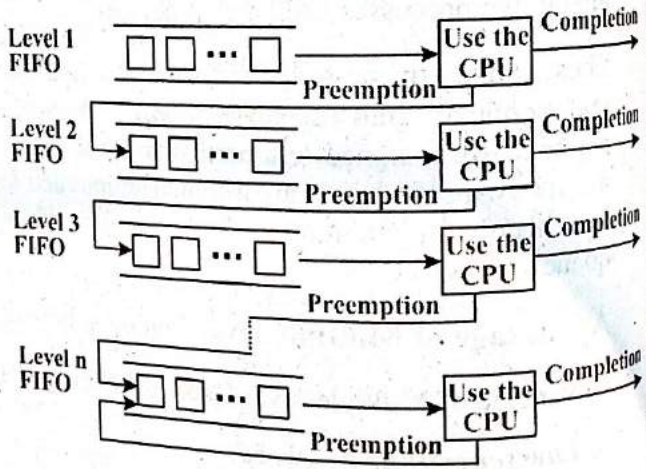
6. Multiple Level Queues with Feedback Scheduling

It is an enhancement of multiple level queue scheduling where processes can move between queues.

MLQ is used in the form of multiple level queues with feedback. Each process may start at the top-level queue. If a process is completed within a given time slice, it terminates after having the high priority consideration.

Processes that need more than one time slice may be reassigned by the operating system to the next queue (lower priority queue). If the process is still not completed it is moved to yet another lower level queue.

e.g. Consider a multilevel feedback queue scheduler with queues 1 to n.



Let each job in queue1 is given a time quantum.If a job doesn't finish within this time it is moved to the tail of queue 2.

When queue 1 is empty the jobs of queue 2 are given their quantums.

If it still does not complete it is put in queue 3 and so on. The jobs in queue 3 are run FCFS only when queues 1 and 2 are empty and so on.

Advantage of Multiple Level Queues with Feedback Scheduling

Advantage of Multiple Level Queues with Feedback Scheduling are:

- *Since it allows a process to move between queues, they do not have to wait for long.*
- *Prevents starvation.*
- *It is more flexible.*

Disadvantages of Multiple Level Queues with Feedback Scheduling

Disadvantage of Multiple Level Queues with Feedback Scheduling are:

- *Moving the processes among the queues increases CPU overhead.*
- *The most complex scheduling algorithm.*

Q10.(a) What is deadlock? What are the various conditions that result in deadlock?
MDU BCA 2013

Ans. Deadlock

Deadlocks can occur in concurrent environments. Deadlocks occur if each process holds one resource and requests the other.

A Deadlock is 'A situation where a group of processes is permanently blocked as a result of each process:

- ***having acquired a subset of resources needed for its completion &***
- ***having to wait for release of the remaining resources held by others in the same group.***

Thus making it impossible for any of the deadlocked processes to proceed.'

In multiprogramming environment several processes compete for a fixed number of resources.

A process requests for a resource and if resource is available, it is granted; and if the resources are not available at that time it enters a wait state. It may never gain access to the resource(s) since they are being held by other waiting processes.

Dijkestra was one of the first to describe the problem of deadlocks.

Conditions causing deadlocks

1. Mutual exclusion

A resource is assigned to exactly one process. If another process requests the resource it is delayed until the release of that resource.

2. Hold and wait

There must be one process that is having one resource and waiting for another resource currently held by another process.

3. No Pre-emption

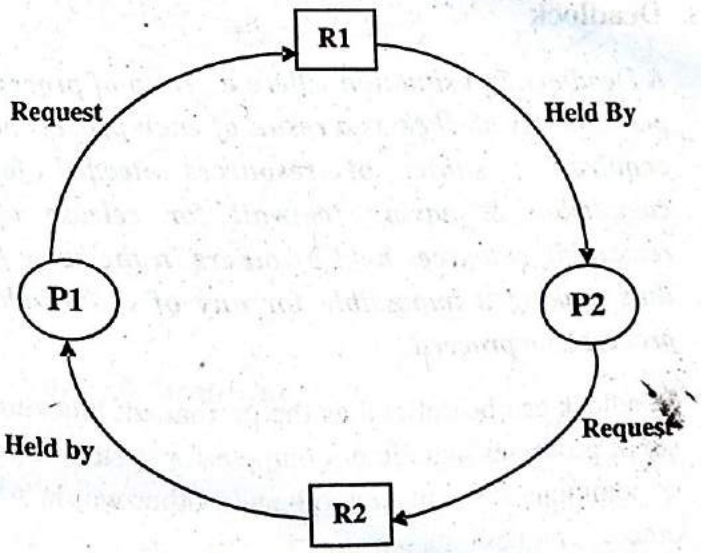
Resources once granted can't be forcibly taken away from a process. They will be properly released by the processes holding them.

4. Circular Wait

There must be a circular chain of two or more processes each of which requires a resource held by another member of the chain.

In the figure, the processes are shown as circles and resources are shown as squares.

The simultaneous existence of these conditions causes deadlock.



Q10.(b) What do you mean by deadlocks? Explain how you handle deadlocks. IGU BCA 2018

OR

What do you mean by deadlock? Explain deadlock prevention, avoidance and detection. MDU BCA 2017

OR

What is deadlock? What are the various strategies to deal with deadlock? Explain. MDU BCA 2015

OR

What do you mean by deadlock? Explain deadlock prevention, avoidance and detection. MDU BCA 2014, 2013

OR

What are the three fundamental approaches used for handling deadlocks?

Ans. Deadlock

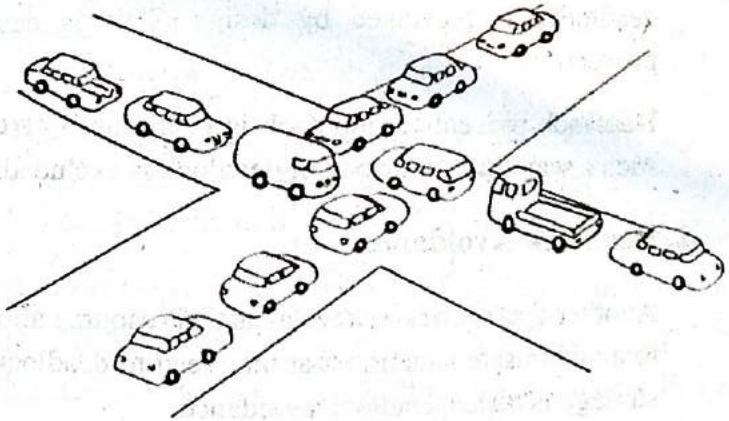
A Deadlock is a situation where a group of processes is permanently blocked as a result of each process having acquired a subset of resources needed for its completion & having to wait for release of the remaining resources held by others in the same group thus making it impossible for any of the deadlocked processes to proceed.

Deadlock can be defined as the permanent blocking of a set of processes that either compete for system resources or communicate with each other. In other words, when a process request resources, if the resources are not available at that time, the process enters a wait state.

Waiting process may never again change state because resources that have requested are held by other waiting process. This situation is called deadlock.

In multiprogramming environment several processes compete for a fixed number of resources. A process requests for a resource and if resource is available, it is granted; and if the resources are not available at that time it enters a wait state. It may never gain access to the resource(s) since they are being held by other waiting processes.

A common example of deadlock is the traffic deadlock. Figure shows a situation in which four cars have arrived at a four-way stop intersection at approximately the same time.



Deadlock handling

Deadlocks are undesirable features, which affect the progress of processes. There is no single method to deal with deadlocks. There are several approaches to solve the deadlock problem.

These are:

1. Deadlock prevention
2. Deadlock avoidance
3. Deadlock detection and recovery

1. Deadlock prevention

Deadlock prevention is a set of methods for ensuring that atleast one of necessary conditions cannot hold.

For deadlock to occur, each of the four necessary conditions must hold. By ensuring that atleast one of these conditions cannot hold, one can prevent the occurrence of a deadlock.

Thus, one way to handle deadlocks is to ensure that at least one of the four necessary conditions causing deadlocks is prevented by design. This is deadlock prevention.

Deadlock prevention approach is to design a system in such a way that possibility of deadlock is excluded.

2. Deadlock Avoidance

Another approach is to have a careful resource allocation to avoid unsafe situation that may lead to deadlock. This strategy is called deadlock avoidance.

3. Deadlock detection and recovery

This approach is to grant resources freely but occasionally examine the system state for deadlock and take remedial action when required. This is called deadlock detection and recovery.

Now we examine the approaches related to each of the four conditions:

1. Deadlock Prevention

Deadlock prevention is to deny at least one of the four necessary conditions for deadlocks.

Mutual exclusion

In general, this condition can not be disallowed.

Since mutual exclusion is difficult to prevent so one or more of the remaining three conditions will be prevented.

Hold-and-wait

This condition can be eliminated by requiring a process to release all resources held by it whenever it requests a resource that is not available.

This means that whenever a process requests a non-available resource, it does not hold any other resource.
We can implement this method in two ways:

(i) Process requests all needed resources at one time before starting execution.

e.g. Let a process is to copy data from a drive to the disk file, sort the disk file and then print the result to a printer. Since all resources must be requested at one time, the process initially requests for the tape drive, disk file and the printer. It will hold all these during entire execution, though it needs the printer only at the end.

(ii) *Another way to prevent hold and wait condition is that the process requests resources incrementally releasing all other resources it currently holds. In other words a process requests resources only when the process has none.*

e.g. It initially requests only the tape drive and disk file. It copies from the tape drive to the disk, then releases both. The process must then again request the disk file and the printer. After copying the disk file to the printer, it releases these two resources and terminates.

Disadvantages of Hold-and-wait

- *Resource utilization may be low, as many of the resources allocated may be unused for a long period.*
- *Starvation may occur. A process requiring several popular resources may have to wait indefinitely, because at least one of the resources that it needs may be allocated to some other process.*

No Pre-emption

This condition can be prevented by allowing preemption.

This can be done in the following ways:

- *A process holding certain resources and requesting for another resource must release its original resources. If necessary, request them again together with the additional resource.*
- *Alternatively, if a process requests a resource, which is currently held by another process, the operating system may preempt the required resource from second process.*

Advantage of No Pre-emption

- *It is convenient when applied to resources whose state can be easily saved and restored.*

Disadvantages of No Pre-emption

- *Preemption is possible only for certain types of resources such as CPU and main memory.*
- *For some resources which cannot be safely preempted, this approach alone cannot provide complete deadlock prevention.*
- *Sometimes this method preempts more often than necessary.*

Circular Wait

The circular wait condition can be prevented in either of the following ways:

(i) ***By defining a linear ordering of different types of system resources***

A process allocated resource of type R, may subsequently request only those resources of the type following R in the ordering.

(ii) ***By providing a global numbering of all the resources.***

All the requests for resources must be in increasing numerical order.

Thus a process holding a resource of highest number will never ask for a resource already assigned.

e.g. Let the ordering be as follows:

1. Phototypesetter
2. Printer
3. Plotter
4. Tape drive
5. CD ROM Drive

A process may request first a printer and then the tape drive but it may not request first a plotter and then a printer.

Advantages of Circular Wait

- *With these rules the circular wait condition will not occur.*

Disadvantage of Circular Wait

- *It is not possible to find an ordering that satisfies everyone.*
- *This approach may slow down processes denying resource access unnecessarily.*

2. Deadlock Avoidance

The basic idea of Deadlock avoidance is to grant only those requests for available resources, which cannot possibly result in deadlock.

It requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime. With this

additional information, it can be decided for each request can be satisfied or must be delayed.

A decision is made dynamically whether the current resource if allocated / granted lead to deadlock.

If possibly cannot the resource is granted to the requesting process.

Otherwise, the requesting process is suspended till the time when its pending request can be safely granted.

The two approaches followed for deadlock avoidance are:

- *Do not start the process if its demands might lead to deadlock*
- *Do not grant an incremental resource request to a process if this allocation might result in deadlock*

Disadvantage

- *It is not possible to know future resource requirement of process.*
- *Process can be blocked for long periods.*

3. Deadlock detection and Recovery

This is a very liberal approach. In this approach, the available resources are granted freely and deadlocks are checked occasionally.

Detection means discovering a deadlock. If deadlock exists the system must break or recover the deadlock.

This approach involves two steps:

Q10.(e) Describe Banker's algorithm. MDU BCA 2016

OR

Explain Banker's algorithm to deal with the problem of deadlocks. MDU BCA 2014

Ans. Banker's algorithm

The banker's algorithm is well known algorithm for the deadlock avoidance strategies. The strategy is modeled after the leading policies employed in the banking system.

The name was chosen because this algorithm could be used in banking system to ensure that the bank never allocates its available cash such that it can no longer satisfy the needs of all of its customers.

Several data structures must be maintained to implement the banker's algorithm. These data structures encode the state of resource allocation system. Let n be the number of processes in the system and m be the number of resource types.

The following data structures are needed:

1. **Available**

   A vector of length m indicates the number of available resources of each type. If available [j] = k, there are k instances of resource type R_j available.

2. **Max**

   An n×m matrix defines the maximum demand of each process. If Max[i, j] = k, then process P_i may request at most k instances of resource type R_j.

3. **Allocation**

   An n×m matrix defines the number of resources of each type currently allocated to each process. If

Allocation[i, j] = k, then process P_i is currently allocated k instances of resource type R_j.

4. **Need**

   An n×m matrix indicates the remaining resource need of each process. If Need[i, j] = k, then process P_i may need k more instances of resource type R_j to complete its task. Need [i, j] = Max [i, j] – Allocation [i, j]

We can treat each row in the matrices **Allocation** and **Need** as vectors and refer to them as **Allocation**$_i$ and **Need**$_i$ respectively. The vector Allocation$_i$ specifies the resources currently allocated to process P_i and Need$_i$ specifies the additional resources that process P_i may still request to complete its task.

Safety Algorithm

Safety Algorithm is used to find the state of the system i.e. the safe or unsafe state. The method for this is as follows:

1. Let Work and Finish be vectors of length m and n respectively. Initialise Work = Available and Finish[i] = False for i = 1, 2, 3, … n.

2. Find an i such that both

   (a) Finish[i] = False

   (b) Need$_i$ ≤ Work

   If no such i exists, go to step 4.

3. Work = Work + Allocation$_i$

   Finish[i] = True

go to step 2

4. If Finish[i] = True for all i,
   then the system is in safe state.

Resource-Request Algorithm

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

1. If $Request_i \leq Need$, then go to step 2. Otherwise, raise an error condition, since the process has exceeded its maximum claim.
2. If $Request_i \leq Available$, then go to step 3. Otherwise, P_i must wait since the resources are not available.
3. $Available = Available - Request_i$
   $Allocation_i = Allocation_i + Request_i$
   $Need_i = Need_i - Request_i$

If the resulting resource-allocation state is safe, the transaction is completed and process P_i is allocated its resources. If the new state is unsafe, then P_i must wait for the $Request_i$ and the old resource-allocation state is restored.

Q.12(d) Explain how logical address is translated into physical address with the neat diagram.

MDU BCA 2018

OR

Discuss Logical and Physical address space.

MDU BCA 2016

OR

Describe logical and physical address space. How to convert logical address to physical address? Explain with suitable example.

Ans. ***A logical address is the address of an instruction/data as used by a program or application. Logical address is also referred to as virtual address.***

Logical addresses may be obtained using index, base or segment registers. It is generated by CPU. Logical addresses are used to refer to information/data within a program's address space.

The set of all logical addresses generated by a program is referred as logical address space.

The user program deals with logical addresses. It never sees the real physical addresses. The logical addresses must be mapped to physical addresses before they are used.

Physical address is the effective memory address of an instruction/data. It is obtained by the address translation equation.

It is generated by MMU (Memory Management Unit) Physical addresses represent the actual main memory address where the program/data are stored.

The set of all physical addresses corresponding to all logical addresses of a program is referred as physical address space of that program.

It is the address seen by the memory unit i.e. the one loaded into the memory address register of the memory.

Conversion of Logical to Physical Address

The run-time mapping of logical addresses to physical addresses is done by Memory Management Unit (MMU). MMU makes use of a register called as relocation or base register. The value in the base register is added to every address generated by a process. The base register is also called relocation register. The value in the relocation register is added to every address generated by a process at the time it is sent to memory.

For example, if the base is at 16000, then an attempt by the user to address location 0 is dynamically relocated to location 16000, an access to location is mapped to location 16400.

The program can create a pointer to location 400, store it in memory, manipulate it and compare it with other addresses all as the number 400.

Only when it is used as memory address, it is relocated relative to the base register. The program code deals with the logical addresses.

Q15.(b) What is swapping? How does it help in memory management? Explain.

MDU BCA 2015

OR

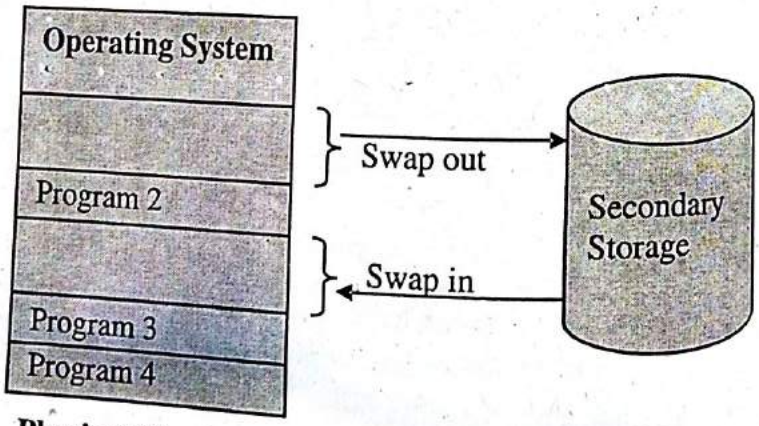
What do you mean by Swapping? Explain the role of swapping in virtual memory management? Also write its advantages and disadvantages.

Ans. Swapping

Swapping is a technique of temporarily shifting a program from the main memory onto the secondary storage, typically a disk.

The moving of program instructions from main memory to the backing store is termed 'swapping out' and back to main memory is 'swapping in'.

Thus, swapping is a technique of temporarily removing inactive program from the memory of a system. It removes the process from the main memory when it is blocked and deallocates the memory. The free memory is allocated to other process.



Physical Memory

A process can be swapped out in the following cases/situations:

- ***On expiry of a time-slice in round robin scheduling.***
- ***Program is neither executing nor performing I/O operation i.e. idle.***
- ***On the arrival of a high priority process.***

Once a program is swapped out, desired program is swapped into the free memory area. Swapping increases the operating system's overhead due to performing swap in and swap out operations. All operating system do not perform swapping like DOS does not perform swapping but UNIX, WINDOWS and OS2 perform swapping.

The memory manager that does the swapping is fast enough to always provide a process in memory for the CPU to execute.

A process that is swapped out of memory can be swapped in either into the same memory location or into a different memory location.

Swapping is used to provide a virtual effect of more memory space i.e. it appears to the user that small main memory is accommodating operating system as well as application program.

Swapping allows the small main memory to hold only a part of the program currently being executed instead of holding the entire program.

Swapping is well suited to timesharing systems. The context switch time in such a swapping system is fairly high. Major part of the swap time is transfer time. The total transfer time is directly proportional to the amount of memory swapped.

Advantages of Swapping

It increases:

- *Memory utilization*
- *CPU utilization*
- *Throughput*

Role of Swapping in Memory management

The virtual memory concept is to accommodate as many application processes in main memory to increase CPU utilization and throughput. This is possible by keeping those portions of these programs which are currently needed for their execution. The remaining portion can be picked up from secondary storage when needed. This means that portions of these programs have to be swapped into the memory for execution and this may also require putting the currently unused portion back onto the disk i.e. swapping out inactive program to disk.

Thus swapping plays an important role in virtual memory management.

Q14.(b) What is meant by contiguous allocation? How are memory allocation achieved under this scheme.

Ans. Contiguous Memory Allocations

This means that each program is loaded or placed in memory locations with consecutive addresses. Such memory partitions may be defined statically or dynamically. The following are contiguous memory allocation methods:

1. Fixed/Static partitioned memory management
2. Dynamic Partitioned Memory Management
3. Relocation Memory Management

1. Fixed / Static Partitioned Memory Management

In this partitioned allocation, the main memory is divided into separate memory regions or partitions. Each partition accommodates a separate job.

Fixed partitioned memory management is also known as static partitioning. In this partitioning, memory is divided into partitions prior to loading of any job.

The partitions may be

- Equal sized or
- Unequal sized.

The size of each partition once fixed can't be changed during the program execution.

Thus, the number of the programs i.e. degree of multiprogramming residing in memory will be bound by the number of partitions.

Whenever a process is ready to execute, a free partition of sufficient amount of memory needed is found and assigned to the requesting process.

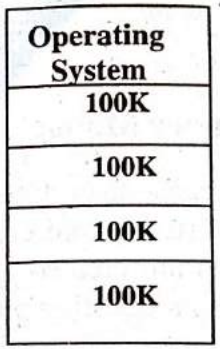


Fig. 1
Equal Sized
Static partitioning

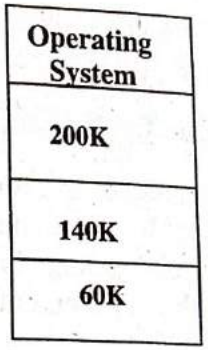


Fig. 2
Unequal Sized
Static partitioning

Advantage

- ***Suitable for batch systems.***

Drawbacks

(i) Inefficient use of memory space

If a program of size say 50K is loaded into memory then it can be accommodated in any of the partition each of 100K in fig.1, thus wasting 50K of memory space or partition of 60K in fig. 2, thus wasting 10K of memory space.

(ii) Internal Fragmentation

The free memory space in different partitions that can't be utilized for other jobs is called internal fragmentation. The memory space left unused in each partition is called a hole.

(iii) Unfitness

Jobs requiring memory space more than the partition size can't be accommodated.

e.g. Consider a job of 210K, it can't be accommodated in any of the given partitions.

2. Dynamic Partitioned Memory Management

This technique is more efficient than fixed partition technique. In this technique partitions are created at the time of job loading. Partitions are created dynamically to meet the requirement of each requesting process/job.

A job is allocated the exact memory space required by it and no more. In this technique neither the size nor the number of partitions is limited.

Partitions are continuously created and allocated to the requesting processes until all physical memory is filled or maximum allowable degree of multiprogramming is reached.

Advantages of Dynamic Partitioned Memory Management

Advantages of Dynamic Partitioned Memory Management are:

(i) Better memory utilization than fixed partitioning.

(ii) Supports greater degree of multiprogramming.

(iii) It requires no special costly hardware.

(iv) Resolves internal fragmentation.

Disadvantages of Dynamic Partitioned Memory Management

Disadvantages of Dynamic Partitioned Memory Management are:

(i) Inefficient use of memory

This memory management technique starts well but after some time a large number of non-continuous holes are created in memory resulting in wastage of memory space.

In fig.(1) job1, job2, job3, job4 are residing in memory. Suppose job1 is completed and job5 requiring 114K is loaded in this partitions. This leaves a blank space called a 'hole' of size 98k.

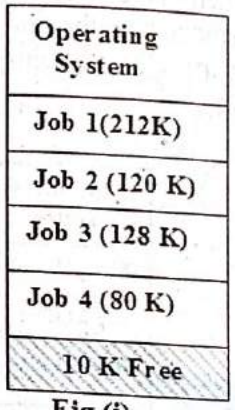


Fig (i)

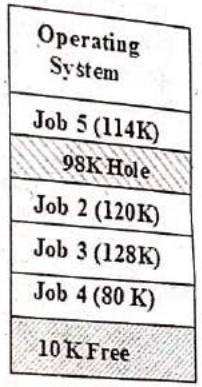


Fig (ii)

(ii) External fragmentation

External fragmentation refers to the free memory space between the partitions, that is enough for a requesting process but it cannot satisfy a request because it is not continuous. For example, in fig.(ii) free memory space is 108K, but it can't satisfy job 6 requiring 100K.

(iii) Complex

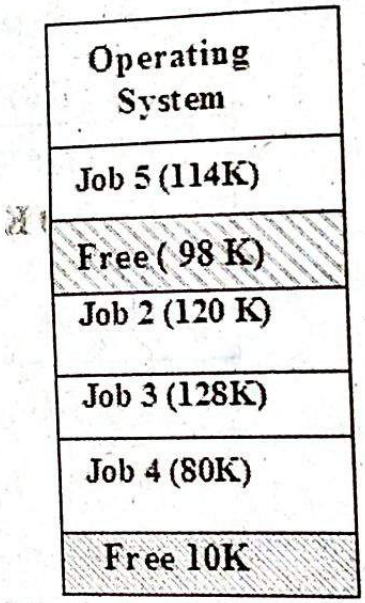
This technique requires complex memory management algorithm and book keeping operation.

3. Relocation Memory Management

This management technique solves the problem of wasted free space, which is generated in the form of non-continuous holes in dynamic partitioned allocations.

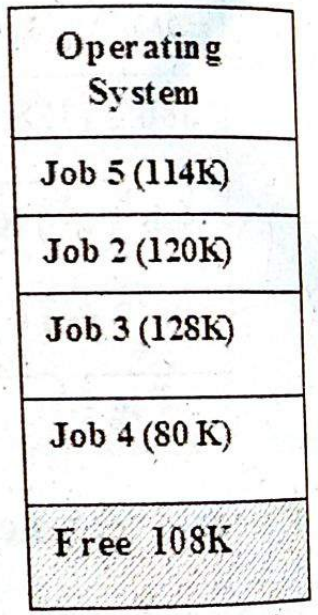
This method deals with the problem of external fragmentation. In this technique the programs in the memory are reallocated in such a way that all the holes or free spaces are moved to one end of the memory space.

The aim is to shuffle the memory contents in order to place all scattered free memory together in one large free area. This free memory space being continuous can be used to store the other jobs.



Before Relocation

Fig (1)



After Relocation

Fig (2)

Figure (1) shows free memory spaces. Figure (2) shows compaction performed by bringing the free memory spaces together. To accommodate job6 requiring 100K, the uncontinuous space in dynamic partition is moved /relocated to one end. Memory compaction may be performed when needed.

Q14.(a) Explain paging. **MDU BCA 2017**

OR

What is paging and how it works?

MDU BCA 2015

OR

What do you mean by paging? How address mapping is performed in paging technique? Also enumerate the advantages and disadvantages of paging.

Ans. Paging

Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous. It avoids external fragmentation.

Thus, to overcome the problems of external fragmentation, one solution is 'paging'. Paging permits the use of non-contiguous memory i.e. allows a program to be allocated to physical memory wherever available.

The program to be executed is divided into blocks of fixed size called pages. The physical memory/main memory is also divided into blocks of the same fixed size as pages called the page frames. When a process is to be executed, its pages are assigned to the available page frames in the memory.

The page size is defined by the hardware. Usually a page size is powers of 2 varying between 512 bytes to 8192 bytes per page.

Implementation of Paging

To implement paging, the operating system maintains a table to keep the record of free and occupied page frames.

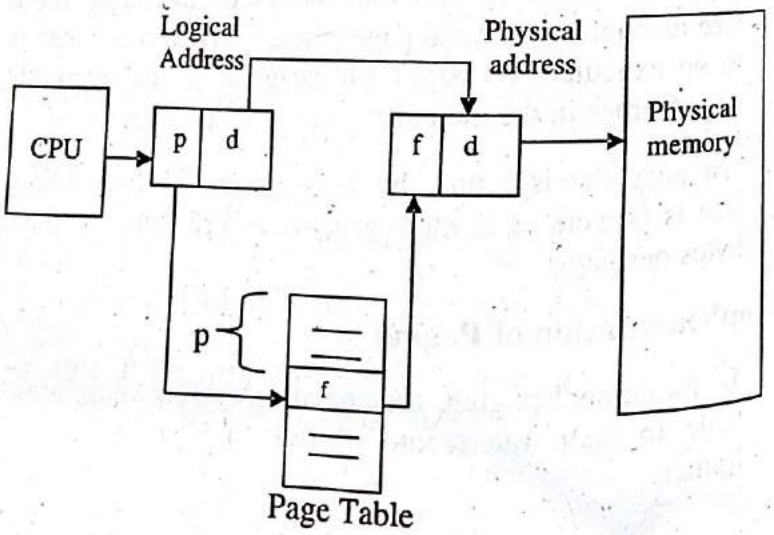
When a process is to be loaded, its pages are moved to free frames in the physical memory. When the program is completed, the page frames occupied by this program are deallocated which can be used for next program. A mapping table, called the page map table (PMT) is constructed at process loading time to establish correspondence between the virtual and physical addresses. This page table has one entry for each page of the process. It also indicates the status of each page i.e. whether space in the physical memory allocated to it or not? If yes, which frame it occupies currently?

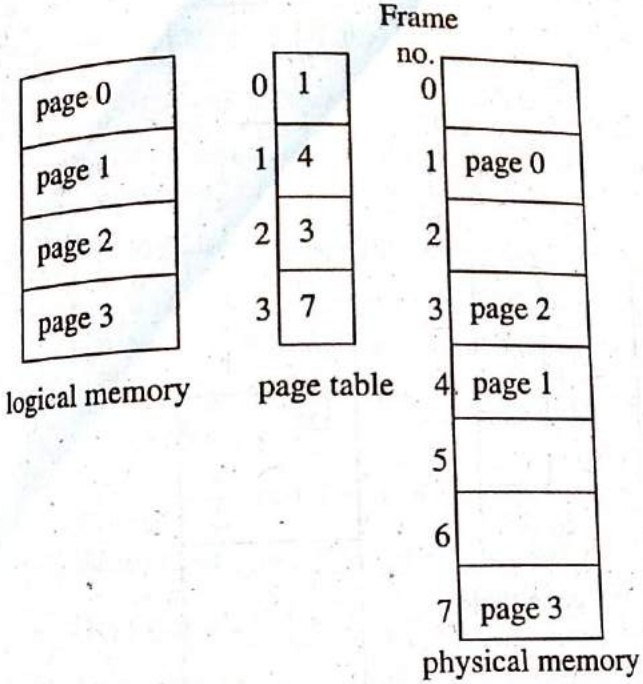
Address mapping/address translation mechanism

(a) During the process execution, CPU generates a logical address, that comprises of page number (p) and offset within the page (d).

(b) The page number (p) is used to index into the page table and fetch corresponding frame number (f), the physical address is obtained by combining the frame number (f) with the offset (d).

Each job has its own page table. The size of a page is usually a power of 2.





Corresponding physical address = $f * 2^n + d$.

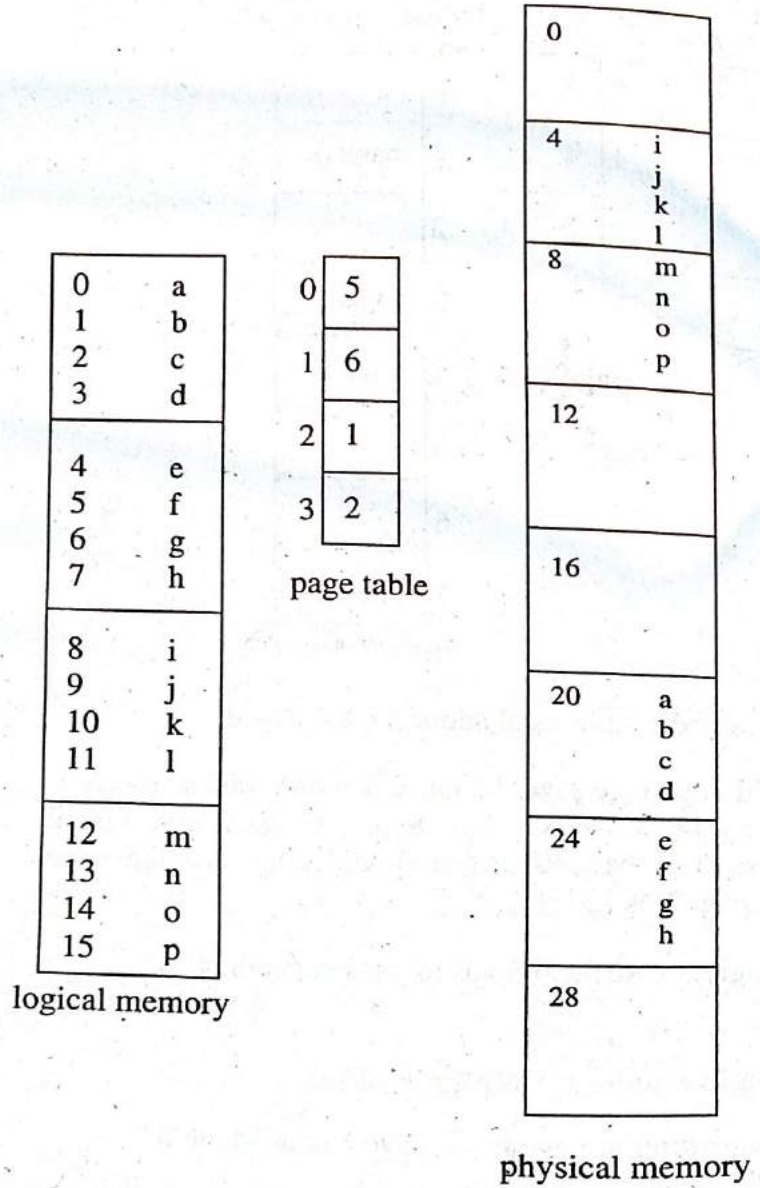
Consider a page size 4 (i.e. 2^2) words and a physical memory of 32 words (i.e. 8 pages). Now the logical address 0 is page 0, offset 0. Indexing into the page table, page 0 is in frame 5.

(i) Logical address 0 maps to physical address =
$5 \times 4 + 0 = 20$

(ii) Logical address 5 is page 1, offset 1.

According to page table, page 1 is in frame 6.

∴Logical address 5 maps to physical address =
$6 \times 4 + 1 = 25$



Thus the logical addresses are translated into physical memory. This mapping is controlled by the operating system and is hidden from the user.

Advantages of Paging

Advantages of Paging are:

(a) Allocation and deallocation of memory in paged system is quite simple.

(b) This technique eliminates external fragmentation.

(c) It allows for a higher degree of multiprogramming by accommodating more jobs simultaneously.

(d) This technique increases memory and processor utilization.

(e) The relocation partition scheme is also eliminated.

Disadvantages of Paging

Disadvantages of Paging are:

(i) A process can be loaded in memory, if and only if, there is sufficient space in memory.

For example, suppose two frame size totaling 8K bytes are available, they can not be used if the minimum job size required is over 8K bytes.

(ii) There may occur some internal fragmentation also.

For example, if a job requires 5K bytes and the block/pages frame size is 4K, two blocks must be allocated thereby wasting 3K memory.

(iii) A page table may become large. This may occupy large space in memory.

Q12.(c) Explain Segmentation. **MDU BCA 2017**

OR

Describe the concept and working of memory segmentation. **MDU BCA 2013**

OR

What is segmentation? What are its advantages and disadvantages? Explain.

Ans. Segmentation

The logical address space of a program is interpreted according to a programmer's own view point instead of as an array of bytes.

A segment can be defined as a logical grouping of information such as subroutine, array, function or data area. Each job is taken as a collection of segments.

Segmentation is the technique for managing these segments.

Segments are formed at program translation time by grouping together, logically related entities. Formation of these segments varies from one compiler to another.

A Pascal compiler may create different segments for:

- ***Global data or variable.***
- ***Local data or variable.***
- ***Code of each procedure.***
- ***Code of each function.***

For simplicity of implementation, each segment in a program is numbered and is referred to by a segment number rather than a segment name.

Q14.(c) Explain demand paging. **MDU BCA 2017**

OR

What do you mean by demand paged memory management? Also write its advantages and disadvantages.

Ans. Demand Paging is a refinement of paging technique. The basic principle of demand paging technique is that a page of a program is loaded in main memory only when it is required. The remaining pages of the program remain on the disk/secondary storage.

While executing a process if the CPU needs a page of the process, which is not in main memory, then a page fault occurs. This page fault is a signal to operating system to bring the desired page from disk to memory.

To bring a page, first a free frame is searched for. Then the desired page is swapped in the main memory from the disk. If no frame is free then page removal takes place i.e. a page is swapped out from memory to disk to make room for the desired/needed page. A page to be removed is selected using anyone of the following replacement algorithms:

- ***The optimal page replacement algorithms***
- ***NRU policy***
- ***FIFO page replacement algorithms***
- ***LRU***
- ***NFU***

The page map table is updated to reflect the change.

Thus, with demand paging, a page is brought into main memory only when a reference is made to a

location on that page. Lazy swapper concept is used in demand paging. A Lazy swapper never swaps a page into a memory unless that page will be needed.

Advantages of demand paged memory management

Since in demand paging, the entire process is not loaded in memory but only a part of it is loaded at a time, therefore this means:

(i) A process requiring memory more than the available memory of the system can be executed.

(ii) More processes can be loaded in memory at a time and the degree of multiprogramming is enhanced.

Disadvantage of demand paged memory management

In case a page removed from memory is needed immediately, it is brought back to memory again immediately. If this process of swapping in and out is too frequent, a condition, known as thrashing occurs. In this condition, CPU is more busy in swapping in and swapping out of pages than in execution.

Q13.(a) What do you mean by page replacement?

MDU BCA 2016

Ans. When a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in. This is known as page replacement.

Page replacement refers to the process in which a resident page in main memory is replaced by a new page transferred from the disk.

Since the number of available page frames is much smaller than the number of pages, the frames will eventually be fully occupied. In order to accommodate a new page, one of the resident pages must be replaced. Different policies have been suggested for page replacement.

The goal of a page replacement policy is to minimize the number of possible page faults so that the effective memory access time can be reduced. The effectiveness of a replacement algorithm depends on the program behavior and memory traffic patterns encountered.

Page replacement takes the following approach:

1. ***It finds the location of the desired page in the disk.***
2. ***It finds a free frame:***
   - ***(a) If there is a free frame, it uses it, otherwise, it uses a page replacement algorithm to select a victim frame.***

Q13.(b) What are the page replacement algorithms? Explain. MDU BCA 2016

OR

Explain the page replacement algorithms in detail. MDU BCA 2015

OR

What are page replacement algorithms? Explain any two page replacement algorithms with example. MDU BCA 2014

Ans. Page Replacement Algorithms

When a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in. The algorithm used for this purpose is known as page replacement algorithm.

Now to select the unrequired page or the page to be removed, many algorithms are used.

Various Page Replacement Algorithms

1. The Optimal Page Replacement Algorithm.
2. NRU (Not Recently Used)
3. FIFO Page Replacement Algorithm.
4. LRU (Least Recently Used)

1. The Optimal Page Replacement Algorithm

This algorithm replaces the page that will not be used for the longest period of time.

The moment the page fault occurs, some set of pages is in memory. One of these will be referenced on the very

next instruction. Other pages may not be referenced until 10,100 or perhaps 1000 instructions. This information can be associated with each page in the PMT(Page Map Table).

P#	Base	Offset	MISC
1			10
2			NIL
3			1000
⋮	⋮	⋮	⋮
10			100

PMT (including several entries)

The optimal page algorithm simply removes the page with the highest number of such instructions implying that it will be needed in the most distant future.

It was introduced long back. It is hardly practicable. It is difficult to implement because it requires future knowledge of the program behaviour.

However it is possible to implement optimal page replacement on the second run by using the page reference information collected on the first run.

2. NRU (Not Recently Used) Page Replacement Algorithm

This algorithm requires that each page is associated with two additional status bits 'R' and 'M' called reference bit and change bit respectively.

The reference bit (R) is automatically set to 1 whenever the page is referenced. The change bit (M) is set to 1

whenever the page is modified. These bits are stored in the PMT and are updated on every memory reference.

When a page fault occurs, the memory manager inspects all the pages and divides them into 4 classes based on R and M bits:

Class 1: (0,0) neither recently used nor modified - the best page to replace.

Class 2: (0,1) not recently used but modified - the page will need to be written out before replacement.

Class 3: (1,0) recently used but clean - probably will be used again soon.

Class 4: (1,1) recently used and modified – probably will be used again, and write out will be needed before replacing it.

This algorithm removes a page at random from the lowest numbered non-empty class.

Advantages

- ***It is easy to understand.***
- ***It is efficient to implement.***

3. FIFO (First in, First out) Page Replacement Algorithm

It is the simplest page replacement algorithm.

The oldest page, which has spent the longest time in memory is chosen and replaced. This algorithm is implemented with the help of FIFO queue to hold the pages in memory. A page is inserted at the rear of the queue and is replaced at the front of the queue.

Initially, 3 page frames are empty. The first 3 references (7,0,1) cause page faults and are brought into these empty frames. The next reference (2) replaces page 7. Since next page reference (0) is already in memory, there is no page fault. Now, for the next reference (3), LRU replacement sees that, of the three frames in memory, page 1 was used least recently, and thus is replaced. And thus the process continues.

Advantages

- ***LRU page replacement algorithm is quite good.***
- ***It does not suffer from Belady's Anomaly.***

Disadvantage

- ***Its implementation is not very easy.***
- ***Its implementation may require substantial hardware assistance.***

Q15.(a) What is thrashing and explain the methods to avoid thrashing. MDU BCA 2018

OR

Explain Thrashing. MDU BCA 2017

OR

What is thrashing? What causes thrashing? How do we overcome it? Explain.

MDU BCA 2016

Ans. Thrashing

Removing a page from memory and then immediately needing it, would require excessively moving the page back and forth between memory and secondary storage. This is called thrashing.

Thus, thrashing occurs when the system spends more of its time in shifting pages between main memory and secondary memory. The activity of repeated swapping is called thrashing.

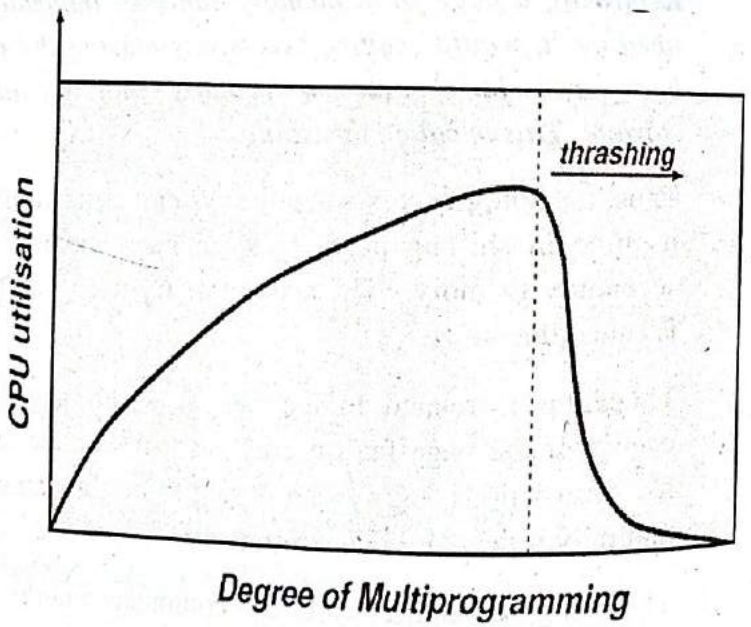
Thrashing is related to the rate at which page fault occurs. If the page fault occurs every time after only a few instructions, the system is said to be thrashing and naturally it decrease the performance.

Thrashing consumes lot of computer energy but accomplishes very little useful results. A process is thrashing if it is spending more time paging than executing. Thrashing results in several performance problems. It decreases or drops the CPU utilization and the system throughput.

Causes of Thrashing

Repeatedly removing processes from memory almost immediately after loading them into memory is the main cause of thrashing. To increase CPU utilization the degree of multi-programming is increased by adding a new process. Gradually, a maximum level of multiprogramming is reached. If the degree of multiprogramming is increased further, thrashing begins and CPU utilization drops sharply.

The diagram shows thrashing:



Prevention of Thrashing

A process must be provided as many frames as it needs. A process that experiences a large number of page faults should be allocated more memory if possible or suspended.

Thrashing has a high page fault rate. To prevent thrashing, the page fault rate must be controlled. When the maximum level of multiprogramming is reached and thrashing begins, the degree of multiprogramming must be decreased at that very point.

Q17.(a) Explain linked and indexed file allocation methods with examples. IGU BCA 2018

OR

What do you understand by file system structure? Describe the various types of allocation method. MDU BCA 2016

OR

What do you mean by file system structure and allocation methods? MDU BCA 2015

Ans. File System Structure

The file system structure is the most basic level of organization in an operating system.

The way an operating system interacts with its users, applications, and security model nearly always depends on how the operating system organizes files on storage devices. Providing a common file system structure ensures users and programs can access and write files.

The file system structure must be designed in such a way that the capacity of the hard disk is optimally utilized without slowing the process of retrieving the data from files.

The goals of the file system structure are as follows:

- ***The required information should be retrieved with a single access to the disk. If the required information is infeasible to retrieve with one access, then it should include as many few accesses to the disk as possible.***

- ***The related information on the disk should be grouped at a single place so that it is easy to retrieve all the related information.***

File allocation methods

Three major methods of allocating disk space widely in use are:

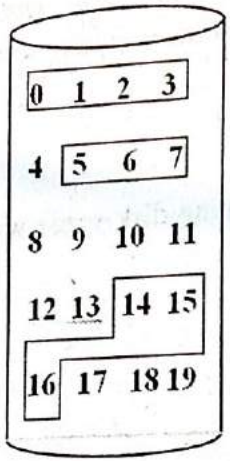
1. Contiguous allocation
2. Linked allocation
3. Indexed allocation

1. Contiguous allocation

In contiguous allocation, files are allocated to a contiguous area on secondary storage. Each file occupies sequential blocks of storage space. For each file it is necessary to indicate the starting address of the file and the file's size.

Thus, a single contiguous set of blocks is allocated to a file at the time of file creation. This is a preallocation strategy that uses portions of variable size. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. Disk addresses define a linear ordering on the disk.

Consider a file of size N blocks long. If starts at location L then it occupies blocks L, L+1, ... L+N-1. The address of the starting block and the size of the length for each file are indicated in the directory entry. This can be shown in the figure:



Directory

File	Start	Length
file A	0	4
file B	5	3
file C	14	3

Contiguous allocation has some problems. External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length.

For time to time, it is necessary to run compaction algorithm for additional space of disk.

Advantages of Contiguous allocation

Advantages of Contiguous allocation are:

1. *It supports both sequential and direct access methods.*
2. *It allows simple and fast sequential access.*
3. *Reading all blocks belonging to each file is very fast.*
4. *Direct access to a file record is also fast.*
5. *It provides good performance because entire file can be read from the disk in a single operation.*

6. *Accessing a file is easy.*
7. *It is also easy to retrieve a single block from a file.*

Disadvantages of Contiguous allocation

Disadvantages of Contiguous allocation are:

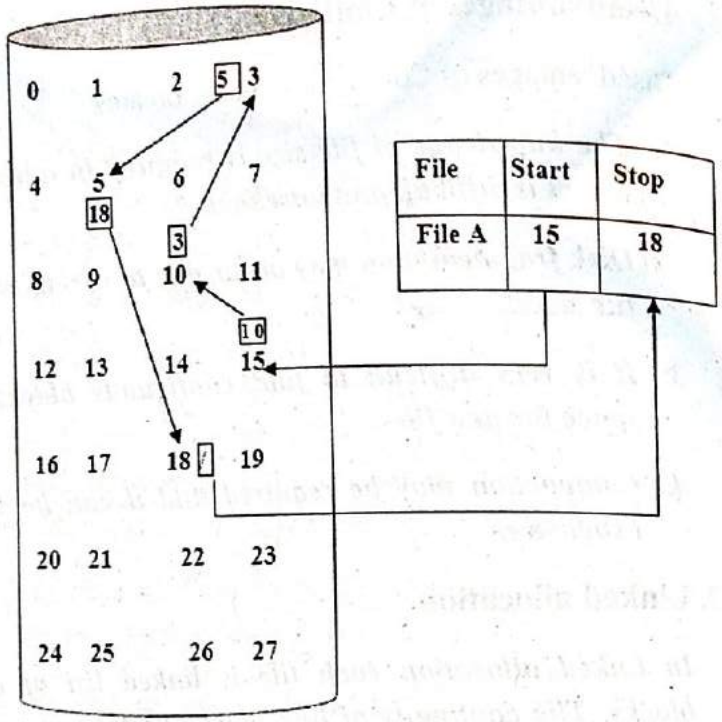
1. *The knowledge of file size is required in advance, which is difficult and unreliable.*
2. *Disk fragmentation may occur due to variability of file sizes.*
3. *It is very difficult to find contiguous blocks of space for new files.*
4. *Compaction may be required and it can be very expensive.*

2. Linked allocation

In linked allocation each file is linked list of disk blocks. The contiguity of link blocks is not necessary. These disk blocks may be scattered through the disk. A few bytes of each disk block are used to indicate the address of the next block.

Thus, linked allocation solves the problem of contiguous allocation. This allocation is on the basis of an individual block. Each block contains a pointer to the next block & the last block contains NIL (/) which shows the end of the file. The disk blocks may be scattered anywhere on the disk. The directory (file allocation table) contains a pointer to the first and last blocks of the file.

In the figure there is a file of 5 disk blocks which starts from block 15 and ends with a disk block 18.



Advantages of Linked allocation

Advantages of Linked allocation are:

1. *The linked allocation eliminates fragmentation.*
2. *Eliminates the contiguity requirement of free space. Any free block on the free-space list can be used to satisfy a request.*
3. *There is no need to declare the size of the file at the moment of file creation.*
4. *A file can be easily extended when required as long as there are free blocks.*

Disadvantages of Linked allocation

Disadvantages of Linked allocation are:

1. *For linked allocation files, sequential access to file is inefficient.*
2. *Linked allocation does not support direct access. To access the Nth block of a file, we have to start at the beginning of that file and follow the pointer until we get to the Nth block.*
3. *Requires extra storage space for pointers, thus decreasing the space utilization.*
4. *Less reliable. As the disk blocks are linked by pointers, so a single damaged pointer can make further disk blocks inaccessible.*

3. Indexed allocation

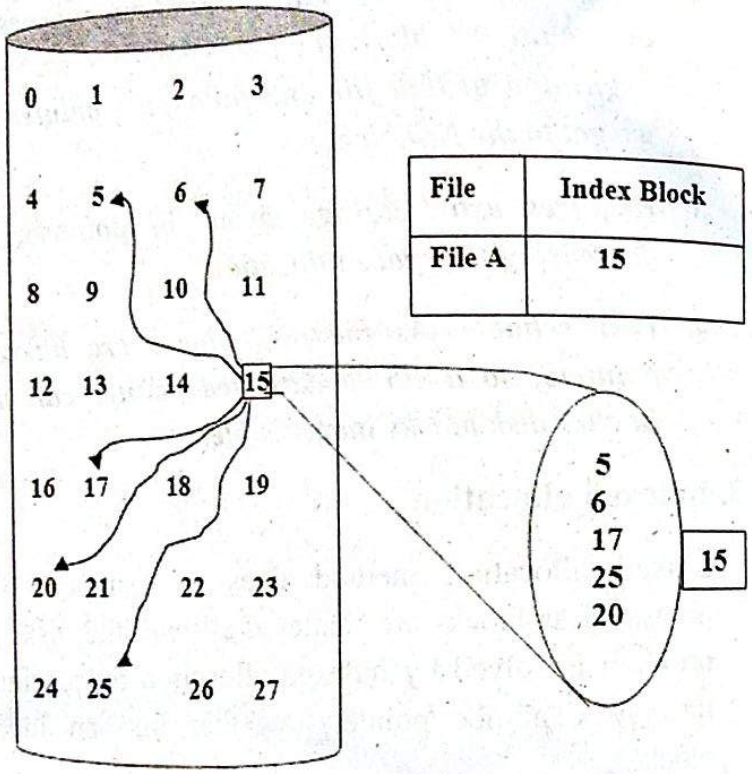
Linked allocation method does not support direct accessing as blocks are scattered all over the disk. This problem is solved by indexed allocation by placing all the block number/pointers together into an indexed block.

Indexed file allocation table contains a separate one level index for each file. The index has one entry for each portion allocated to the file. The file allocation table contains block numbers for the index. If a file requires n blocks, then $n+1$ blocks are used, where first block contains index information to data blocks.

Thus, in indexed allocation method, all the pointers are gathered together into one location known as Index

Block. Each file has its own index block which stores the addresses of disk space occupied by the file. Directory contains the addresses of index blocks of files.

The principle of indexed allocation is explained as below:



In indexed allocation, all the block numbers/pointers of a file are kept into one block, known as index block. The directory entry contains the address of index block.

Advantages of Indexed allocation

Advantages of Indexed allocation are:

1. *This allocation supports direct access.*
2. *This allocation does not suffer from any fragmentation.*
3. *The entire block is available for data.*

Disadvantages of Indexed allocation

Disadvantages of Indexed allocation are:

1. *Indexed allocation requires a lot of space for keeping pointers causing wastage of space.*
2. *Most files are small, so we only need a space for one pointer per block. With this allocation, an entire index block must be allocated even if only one or two pointers exist.*
3. *Looking up for an entry in a large index will also become very time consuming.*

Q19.(a) Explain the concept of Free space management in detail. MDU BCA 2015

OR

Explain Free Space Management. List and explain the following different methods for free space management:

1. Bit vector

2. Linked list

3. Grouping

4. Counting

Ans. Free Space Management

Since there is only a limited amount of disk space, it is necessary to reuse the space from deleted files for new files. To keep track of free disk space, the system maintains a free-space list. The free-space list records all disk blocks that are free (i.e., are not allocated to some file). To create a file, the free-space list has to be searched for the required amount of space, and allocate that space to a new file. This space is then removed from the free-space list. When a file is deleted, its disk space is added to the free-space list.

Four techniques are used for free space management. These are:

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

1. Bit vector

For knowing the list of free blocks information on the disk, the technique used is called bit vector.

The free-space list is implemented as a bit map or bit vector. Each block is represented by one bit. So to track all the free and used blocks on a disk with total n blocks, a bit map having n bits is required. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

Example: Consider a disk where blocks 1, 6, 7, 14 and 15 are free and the rest of the blocks are allocated. the free space bit map would be:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	.....
0	1	0	0	0	0	1	1	0	0	0	0	0	0	1	1	.....

The bit map is usually kept in the main memory to optimize the search for free blocks. However, for systems with large disks, keeping the complete bit map in the main memory becomes difficult.

The main advantage of this approach is its relative simplicity and its efficiency in finding the first free block or *n* consecutive free blocks on the disk.

Advantages

- ***Simple to implement.***
- ***Efficient to find the first free block or n consecutive free block blocks.***
- ***Easy to get contiguous files.***
- ***It is as small as possible and can be kept in main memory.***

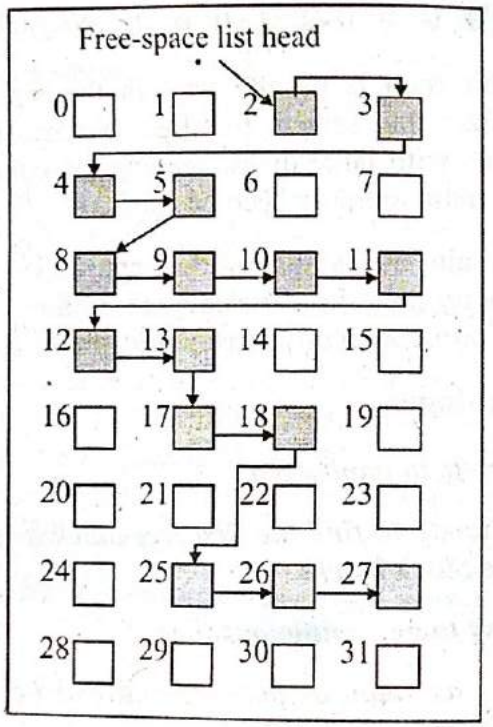
Disadvantage

- ***This method is inefficient if the entire vector is not kept in main memory.***
- ***Bit map requires extra space.***

2. Linked List

Linked list approach links all free disk blocks together. It keeps a pointers to the first free block. This block contains a pointer to the next free disk block, and so on.

In the figure on the next page, a pointer is kept to block 2, as the first free block. Block 2 contains a pointer to block 3, which points to block 4, which points to block 5, which points to block 8, and so on.



Advantages

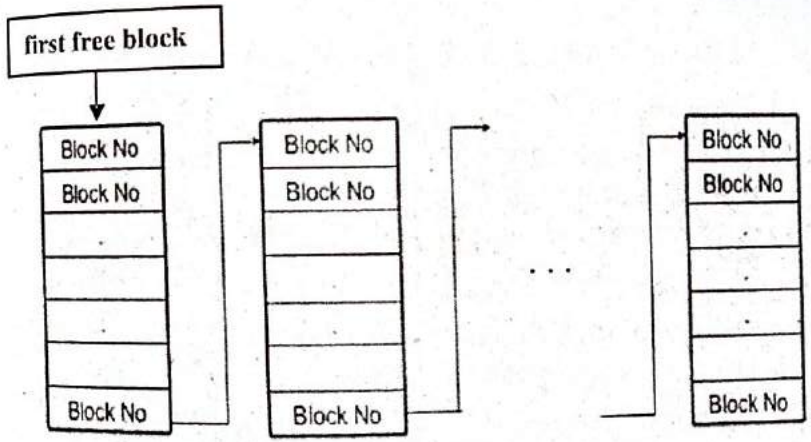
- *This method has negligible space overhead because there is no need for a disk allocation table, merely for a pointer to the beginning of the chain.*

Disadvantages

- *It is not efficient for faster access.*
- *This scheme is not efficient because to traverse the list, each block must be read, which requires substantial I/O time.*
- *It cannot get contiguous space easily.*

3. Grouping

A modification of the free-list approach is to store the addresses of n free blocks in the first free block. The first n-1 of these are actually free. The last one is the disk address of another block containing addresses of another n free blocks.



Advantage

The main advantage of this implementation is that addresses of a large number of free blocks can be found quickly.

4. Counting

Another approach is to take advantage of the fact that, generally, several contiguous blocks may be allocated or

freed simultaneously, particularly when contiguous allocation is used. Thus, rather than keeping a list of free disk addresses, the address of the first free block is kept and the number n of free contiguous blocks that follow the first block.

Thus, this approach keeps the address of the first free block and the number n of free contiguous blocks that follow the first block. Each entry in the free space list then consists of a disk address and a count.

Q20.(a) Discuss disk scheduling. **IGU BCA 2018**

OR

Why disk scheduling is necessary? Explain the various disk scheduling methods with example. **MDU BCA 2017**

OR

Explain the concept of disk scheduling in detail. **MDU BCA 2015**

Ans. Disk Scheduling

To store data and instructions in the system for a long time, non-volatile disks are very important. Disks not only store huge data but also allow random access to the data stored in units called disk blocks or pages.

To retrieve data from a disk block depends upon the location of the block on the disk. And the disk is organized into zones of cylinders with a number of tracks and sectors. In multiprogramming environment many processes running simultaneously may request for the disk access. To service these requests put into the queue, the faster accessing is needed. Hence the process of ordering/scheduling of these requests in some manner is called Disk scheduling.

Why disk scheduling is necessary?

In the multiprogramming environment there may be frequent requests for the disk access. These requests are put into a queue. Faster accessing is needed to service the requests in the queue. So disk scheduling or ordering of the requests in the queue is needed.

Thus disk scheduling is necessary to:

- ***Minimize time spent seeking records.***
- ***Minimize the average time that a request must wait.***
- ***Maximize the throughput.***

Disk Scheduling algorithms

Various disk scheduling algorithms are:

1. First Come First Service (FCFS) Scheduling

This is the simplest form of disk scheduling. In this scheduling, the first request to arrive is the first one to be serviced. Once a request has arrived, its place in the schedule is fixed irrespective of arrival of any higher priority request. This FCFS implements a fair policy. FCFS does not always provide the best service.

2. Shortest Seek Time First (SSTF) Scheduling

In this scheduling approach, priority is given to those requests which need the shortest seek time, without considering their order of arrival. It intends to service the requests nearer to the current head position first before moving head to distant tracks.

3. SCAN Scheduling

In scan scheduling, steps taken are:

- ***Firstly, a direction of sweep, from the current head position, is chosen (inward or outward).***
- ***Then all requests are responded in SSTF manner.***

The read/write head starts from one end and moves towards the other end, services requests as it reaches each track until it reaches to the other end of the disk. After reaching the other end of disk, disk head reverses its path direction while continuing with services whichever comes on the way. The disk head, thus, continuously oscillates from end to end, from the most inner cylinder to the outer cylinder, then it changes its direction back towards the center. This scheduling is fairer/better than SSTF as it reduces starvation.

4. Circular-scan (C-SCAN) scheduling

This algorithm is obtained by a small modification in the SCAN scheduling. C-SCAN scheduling like SCAN scheduling moves the head from one end of the disk to the other, servicing requests along the way. The only difference is that when the head reaches the other end, it immediately returns to the beginning of the disk, without servicing any request on its way back i.e. return trip.

5. LOOK Scheduling

Both Scan and C-scan scheduling move the disk arm across the full width of the disk. Look scheduling is a modified algorithm of the two. In Look scheduling, the arm goes only as far as the final request in each direction, then it reverses direction. It implies looking for a request before moving in that direction. Such versions of Scan and C-scan are called as Look and C-Look algorithms.

Q20.(b) State the desirable characteristics of disk scheduling policies and explain briefly the various seek optimization scheduling policies.

Ans. Disk scheduling involves ordering of the requests in the most efficient way to service the requests in the queue. The desirable main characteristics of disk scheduling policy are:

- **Throughput**

  A scheduling policy should allow the maximum number of requests to be serviced per unit of time.

  To improve throughput the scheduling policy should minimize the time wasted in performing lengthy seeks.

- **Mean response time**

  The scheduling policy should minimize the average waiting time or mean response time.

Seek optimization scheduling policies

Some of the most popular seek optimization algorithms/policies are:-

1. First Come First Serve (FCFS) Scheduling.
2. Shortest Seek Time First (SSTF) Scheduling.
3. Scan Scheduling.
4. Circular-scan (C-SCAN) Scheduling.

These algorithms will be discussed in detail in Q21.