Operating System Tutorial

OPERATING SYSTEM TUTORIAL

Simply Easy Learning by tutorialspoint.com

tutorialspoint.com

ABOUT THE TUTORIAL

Operating System Tutorial

An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. The operating system is a vital component of the system software in a computer system.

This tutorial will take you through step by step approach while learning Operating System concepts.

Audience

This reference has been prepared for the computer science graduates to help them understand the basic to advanced concepts related to Operating System.

Prerequisites

Before you start proceeding with this tutorial, I'm making an assumption that you are already aware about basic computer concepts like what is keyboard, mouse, monitor, input, output, primary memory and secondary memory etc. If you are not well aware of these concepts then I will suggest going through our short tutorial on Computer Fundamentals.

Copyright & Disclaimer Notice

Table of Contents

CHAPTER

1

Overview

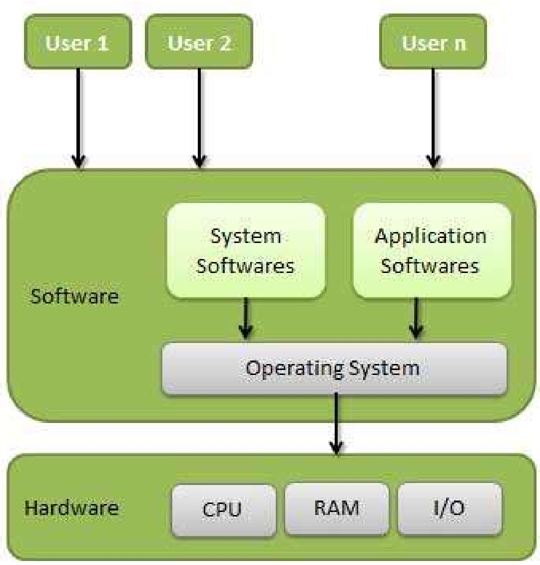
This chapter gives a basic idea about Operating System starting with definition of operating system, and its functions.

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.

In technical terms, it is software which manages hardware. An operating System controls the allocation of resources and services such as memory, processors, devices and information.

Definition

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main memory provides a fast storage that can be access directly by the CPU. So for a program to be executed, it must in the main memory. Operating System does the following activities for memory management.

- Keeps tracks of primary memory i.e. what part of it are in use by whom, what part are not in use.
- In multiprogramming, OS decides which process will get memory when and how much.
- Allocates the memory when the process requests it to do so.
- De-allocates the memory when the process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, OS decides which process gets the processor when and how much time. This function is called process scheduling. Operating System does the following activities for processor management.

- Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De-allocates processor when processor is no longer required.

Device Management

OS manages device communication via their respective drivers. Operating System does the following activities for device management.

- Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Operating System does the following activities for file management.

- Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities

Following are some of the important activities that Operating System does.

- **Security** -- By means of password and similar other techniques, preventing unauthorized access to programs and data.
- **Control over system performance** -- Recording delays between request for a service and response from the system.
- **Job accounting** -- Keeping track of time and resources used by various jobs and users.
- **Error detecting aids** -- Production of dumps, traces, error messages and other debugging and error detecting aids.
- **Coordination between other software and users** -- Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

CHAPTER

2

Types of Operating Systems

This section describes various types of Operating Systems.

Operating systems are there from the very first computer generation. Operating systems keep evolving over the period of time. Following are few of the important types of operating system which are most commonly used.

Batch operating system

The users of batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. Thus, the programmers left their programs with the operator. The operator then sorts programs into batches with similar requirements.

The problems with Batch Systems are following.

- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU.
- Difficult to provide the desired priority.

Time-sharing operating systems

Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing. The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of multiprogrammed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.

Multiple jobs are executed by the CPU by switching between them, but the switches occur so frequently. Thus, the user can receive an immediate response. For example, in a transaction processing, processor execute each user program in a short burst or quantum of computation. That is if n users are present, each user can get time quantum. When the user submits the command, the response time is in few seconds at most.

Operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Advantages of Timesharing operating systems are following

- Provide advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Timesharing operating systems are following.

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Distributed operating System

Distributed systems use multiple central processors to serve multiple real time application and multiple users. Data processing jobs are distributed among the processors accordingly to which one can perform each job most efficiently.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, and computers and so on.

The advantages of distributed systems are following.

- With resource sharing facility user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Network operating System

Network Operating System runs on a server and and provides server the capability to manage data, users, groups, security, applications, and other networking functions. The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks. Examples of network operating systems are Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are following.

- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are following.

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Real Time operating System

Real time system is defines as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. Real time processing is always on line whereas on line system need not be real time. The time taken by the system to respond to an input and display of required updated information is termed as response time. So in this method response time is very less as compared to the online processing.

Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application. Real-time operating system has well-defined, fixed time constraints otherwise system will fail. For example Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-appliance controllers, Air traffic control system etc.

There are two types of real-time operating systems.

Hard real-time systems

Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems secondary storage is limited or missing with data stored in ROM. In these systems virtual memory is almost never found.

Soft real-time systems

Soft real time systems are less restrictive. Critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, Multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers etc.

CHAPTER

3

Operating System Services

This section discusses various services provided by an Operating System.

An Operating System provides services to both the users and to the programs.

- It provides programs, an environment to execute.
- It provides users, services to execute the programs in a convenient manner.

Following are few common services provided by operating systems.

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

Program execution

Operating system handles many kinds of activities from user programs to system programs like printer spooler, name servers, file server etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management.

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

I/O Operation

I/O subsystem comprised of I/O devices and their corresponding driver software. Drivers hides the peculiarities of specific hardware devices from the user as the device driver knows the peculiarities of the specific device.

Operating System manages the communication between user and device drivers. Following are the major activities of an operating system with respect to I/O Operation.

- I/O operation means read or write operation with any file or any specific I/O device.
- Program may require any I/O device while running.
- Operating system provides the access to the required I/O device when required.

File system manipulation

A file represents a collection of related information. Computer can store files on the disk (secondary storage), for long term storage purpose. Few examples of storage media are magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management.

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Communication

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, operating system manages communications between processes. Multiple processes with one another through communication lines in the network.

OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication.

- Two processes often require data to be transferred between them.
- The both processes can be on the one computer or on different computer but are connected through computer network.
- Communication may be implemented by two methods either by Shared Memory or by Message Passing.

Error handling

Error can occur anytime and anywhere. Error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling.

- OS constantly remains aware of possible errors.
- OS takes the appropriate action to ensure correct and consistent computing.

Resource Management

In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management.

- OS manages all kind of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Protection

Considering computer systems having multiple users the concurrent execution of multiple processes, then the various processes must be protected from each another's activities.

Protection refers to mechanism or a way to control the access of programs, processes, or users to the resources defined by computer systems. Following are the major activities of an operating system with respect to protection.

- OS ensures that all access to system resources is controlled.
- OS ensures that external I/O devices are protected from invalid access attempts.
- OS provides authentication feature for each user by means of a password.

CHAPTER

4

Operating System Properties

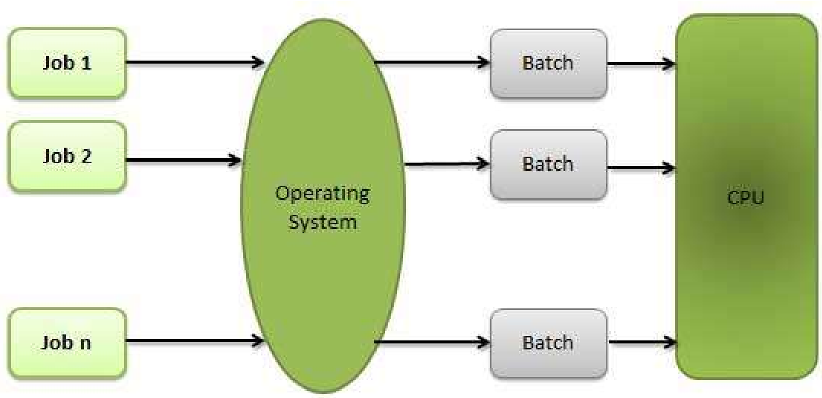
This section discusses various properties of an Operating System.

Following are few of very important tasks that Operating System handles.

Batch processing

Batch processing is a technique in which Operating System collects one programs and data together in a batch before processing starts. Operating system does the following activities related to batch processing.

- OS defines a job which has predefined sequence of commands, programs and data as a single unit.
- OS keeps a number a jobs in memory and executes them without any manual information.
- Jobs are processed in the order of submission i.e. first come first served fashion.
- When job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.



Advantages

- Batch processing takes much of the work of the operator to the computer.

- Increased performance as a new job gets started as soon as the previous job finished without any manual intervention.

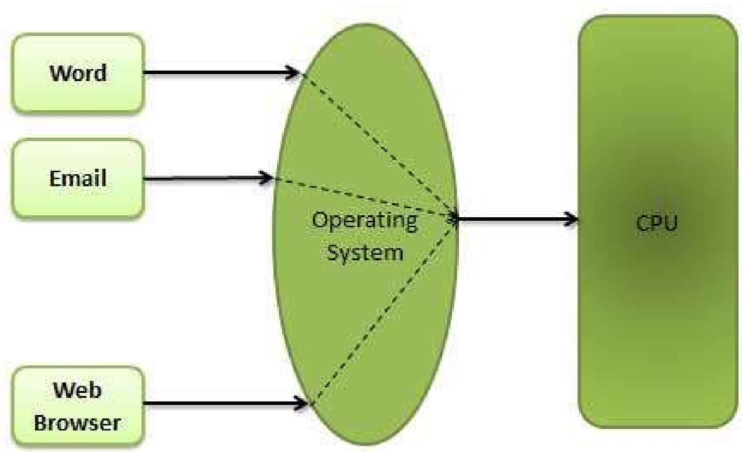
Disadvantages

- Difficult to debug program.
- A job could enter an infinite loop.
- Due to lack of protection scheme, one batch job can affect pending jobs.

Multitasking

Multitasking refers to term where multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running. Operating system does the following activities related to multitasking.

- The user gives instructions to the operating system or to a program directly, and receives an immediate response.
- Operating System handles multitasking in the way that it can handle multiple operations / executes multiple programs at a time.
- Multitasking Operating Systems are also known as Time-sharing systems.
- These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost.
- A time-shared operating system uses concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU.
- Each user has at least one separate program in memory.



- A program that is loaded into memory and is executing is commonly referred to as a process.

- When a process executes, it typically executes for only a very short time before it either finishes or needs to perform I/O.

- Since interactive I/O typically runs at people speeds, it may take a long time to complete. During this time a CPU can be utilized by another process.

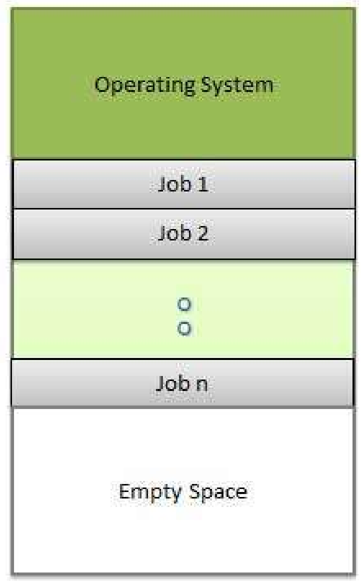
- Operating system allows the users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.

- As the system switches CPU rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users.

Multiprogramming

When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Following figure shows the memory layout for a multiprogramming system.



Operating system does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the job in the memory.
- Multiprogramming operating system monitors the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle unless there are no jobs

Advantages

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.

Disadvantages

- CPU scheduling is required.
- To accommodate many jobs in memory, memory management is required.

Interactivity

Interactivity refers that a User is capable to interact with computer system. Operating system does the following activities related to interactivity.

- OS provides user an interface to interact with system.
- OS managers input devices to take inputs from the user. For example, keyboard.
- OS manages output devices to show outputs to the user. For example, Monitor.
- OS Response time needs to be short since the user submits and waits for the result.

Real Time System

Real time systems represents are usually dedicated embedded systems. Operating system does the following activities related to real time system activity.

- In such systems, Operating Systems typically read from and react to sensor data.
- The Operating system must guarantee response to events within fixed periods of time to ensure correct performance.

Distributed Environment

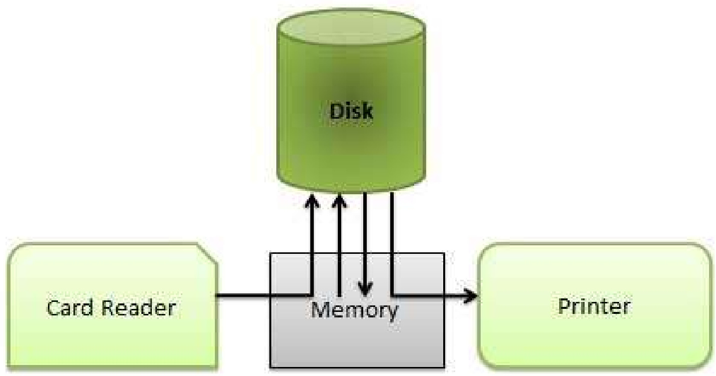
Distributed environment refers to multiple independent CPUs or processors in a computer system. Operating system does the following activities related to distributed environment.

- OS Distributes computation logics among several physical processors.
- The processors do not share memory or a clock.
- Instead, each processor has its own local memory.
- OS manages the communications between the processors. They communicate with each other through various communication lines.

Spooling

Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices. Operating system does the following activities related to distributed environment.

- OS handles I/O device data spooling as devices have different data access rates.
- OS maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.
- OS maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashion. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task.



Advantages

- The spooling operation uses a disk as a very large buffer.
- Spooling is capable of overlapping I/O operation for one job with processor operations for another job.

CHAPTER

5

Operating System Processes

This section describes process, process states and process control block (PCB).

Process

A process is a program in execution. The execution of a process must progress in a sequential fashion. Definition of process is following.

- A process is defined as an entity which represents the basic unit of work to be implemented in the system.

Components of a process are following.

S.N.	Component & Description
1	**Object Program** Code to be executed.
2	**Data** Data to be used for executing the program.
3	**Resources** While executing the program, it may require some resources.
4	**Status** Verifies the status of the process execution. A process can run to completion only when all requested resources have been allocated to the process. Two or more processes could be executing the same program, each using their own data and resources.

Program

A program by itself is not a process. It is a static entity made up of program statement while process is a dynamic entity. Program contains the instructions to be executed by processor.

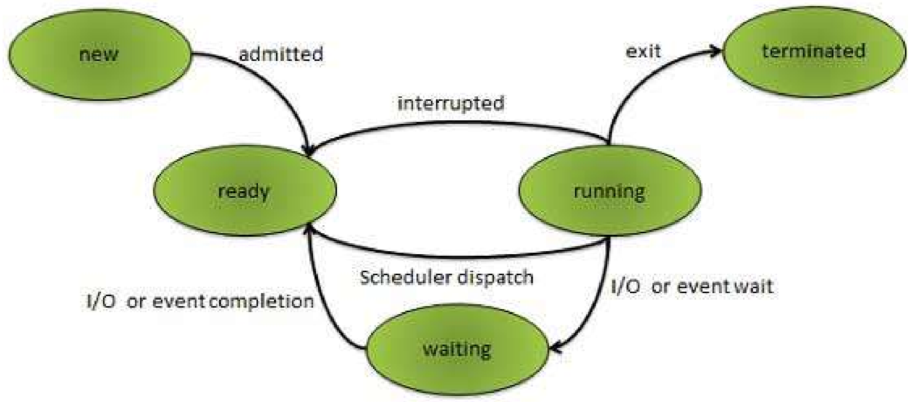
A program takes a space at single place in main memory and continues to stay there. A program does not perform any action by itself.

Process States

As a process executes, it changes state. The state of a process is defined as the current activity of the process.

Process can have one of the following five states at a time.

S.N.	State & Description
1	**New** The process is being created.
2	**Ready** The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3	**Running** Process instructions are being executed (i.e. The process that is currently being executed).
4	**Waiting** The process is waiting for some event to occur (such as the completion of an I/O operation).
5	**Terminated** The process has finished execution.

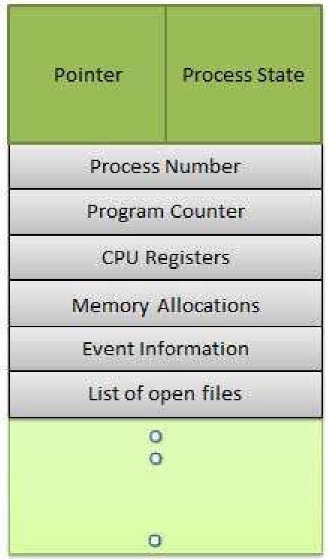

new
admitted
exit
terminated
interrupted
ready
running
Scheduler dispatch
I/O or event wait
I/O or event completion
waiting

Process Control Block, PCB

Each process is represented in the operating system by a process control block (PCB) also called a task control block. PCB is the data structure used by the operating system. Operating system groups all information that needs about particular process.

PCB contains many pieces of information associated with a specific process which is described below.

S.N.	Information & Description
1	**Pointer** Pointer points to another process control block. Pointer is used for maintaining the scheduling list.
2	**Process State** Process state may be new, ready, running, waiting and so on.
3	**Program Counter** Program Counter indicates the address of the next instruction to be executed for this process.
4	**CPU registers** CPU registers include general purpose register, stack pointers, index registers and accumulators etc. number of register and type of register totally depends upon the computer architecture.
5	**Memory management information** This information may include the value of base and limit registers, the page tables, or the segment tables depending on the memory system used by the operating system. This information is useful for deallocating the memory when the process terminates.
6	**Accounting information** This information includes the amount of CPU and real time used, time limits, job or process numbers, account numbers etc.



Process control block includes CPU scheduling, I/O resource management, file management information etc. The PCB serves as the repository for any information which can vary from process to process. Loader/linker sets flags and registers when a process is created. If that process gets suspended, the contents of the registers are saved on a stack and the pointer to the particular stack frame is stored in the PCB. By this technique, the hardware state can be restored so that the process can be scheduled to run again.

CHAPTER

6

Operating System Process Scheduling

This section describes process scheduling, scheduling queues and various types of process schedulers.

Definition

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

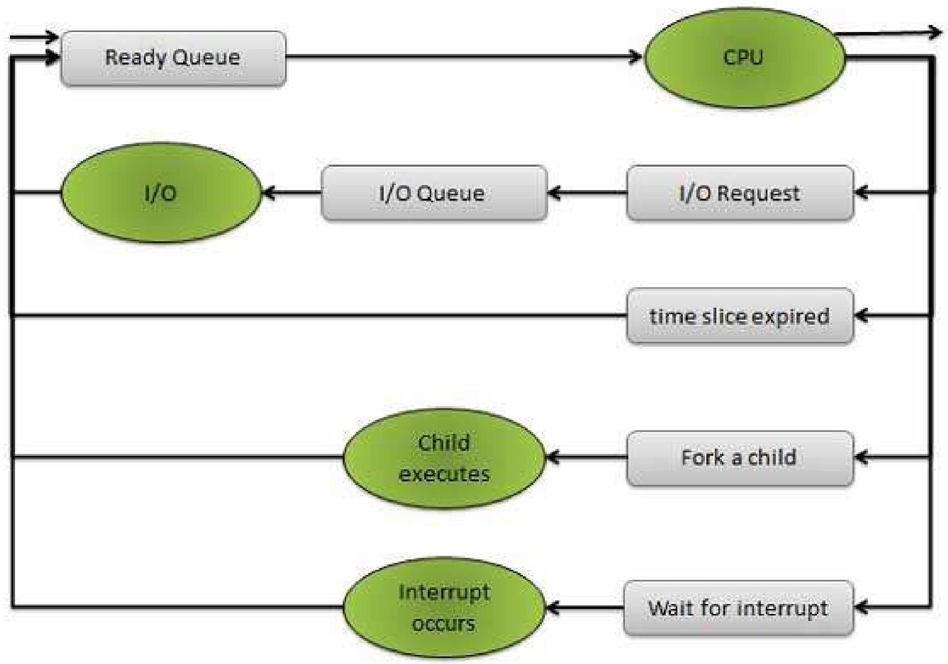
Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU using time multiplexing.

Scheduling Queues

Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. The operating system also maintains other queues such as device queue. Device queue is a queue for which multiple processes are waiting for a particular I/O device. Each device has its own device queue.

This figure shows the queuing diagram of process scheduling.

- Queue is represented by rectangular box.
- The circles represent the resources that serve the queues.
- The arrows indicate the process flow in the system.



Queues are of two types

- Ready queue
- Device queue

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.

- The process could issue an I/O request and then it would be placed in an I/O queue.
- The process could create new sub process and will wait for its termination.
- The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

Two State Process Model

Two state process model refers to running and non-running states which are described below.

S.N.	State & Description
1	**Running** When new process is created by Operating System that process enters into the system as in the running state.
2	**Non-Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types

- Long Term Scheduler
- Short Term Scheduler
- Medium Term Scheduler

Long Term Scheduler

It is also called job scheduler. Long term scheduler determines which programs are admitted to the system for processing. Job scheduler selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling. The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When process changes the state from new to ready, then there is use of long term scheduler.

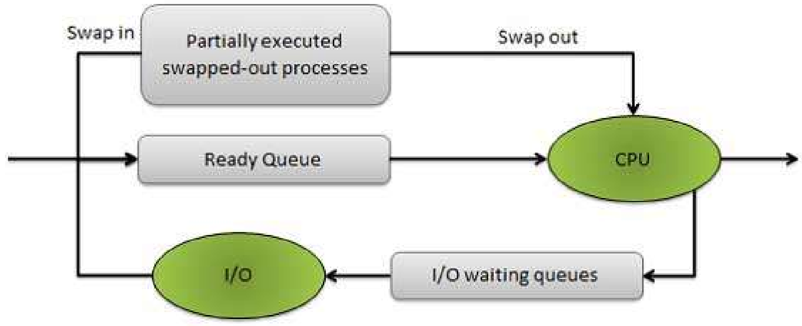
Short Term Scheduler

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.

Short term scheduler also known as dispatcher, execute most frequently and makes the fine grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Medium Term Scheduler

Medium term scheduling is part of the swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium term scheduler is in-charge of handling the swapped out-processes.



Running process may become suspended if it makes an I/O request. Suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other process, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

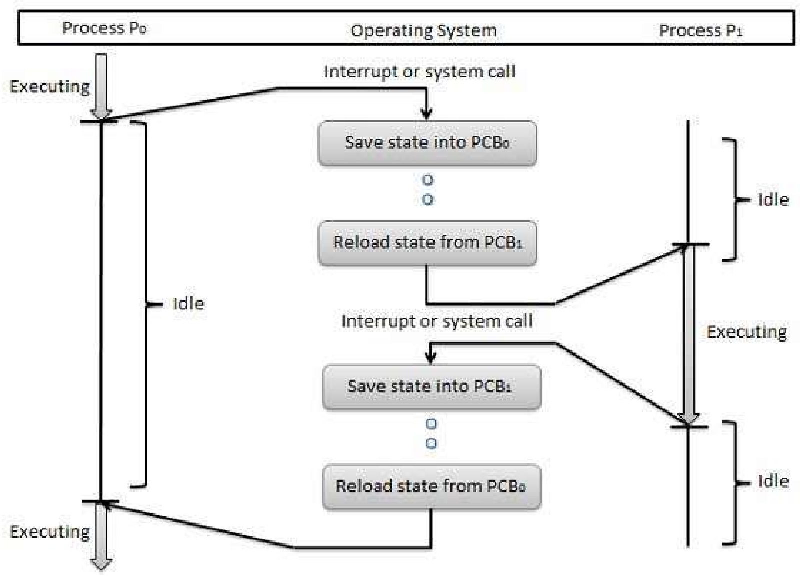
Comparison between Scheduler

S.N.	Long Term Scheduler	Short Term Scheduler	Medium Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch

A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CPU from executing one process to execute another, the context switcher saves the content of all processor registers for the process being removed from the CPU, in its process descriptor. The context of a process is represented in the process control block of a process. Context switch time is pure overhead. Context switching can significantly affect performance as modern computers have a lot of general and status registers to be saved. Content switching times are highly dependent on hardware support. Context switch requires (n + m) bxK time units to save the state of the processor with n general registers, assuming b are the store operations are required to save n and m registers of two process control blocks and each store instruction requires K time units.



Some hardware systems employ two or more sets of processor registers to reduce the amount of context switching time. When the process is switched, the following information is stored.

- Program Counter
- Scheduling Information
- Base and limit register value
- Currently used register
- Changed State
- I/O State
- Accounting

CHAPTER

7

Process Scheduling Algorithms

This section describes various scheduling algorithms like FCFS, SJF, RR and Multilevel Queue Scheduling.

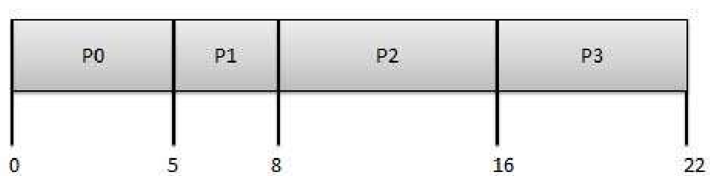
We'll discuss four major scheduling algorithms here which are following

- First Come First Serve (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round Robin(RR) Scheduling
- Multilevel Queue Scheduling

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- Easy to understand and implement.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is following

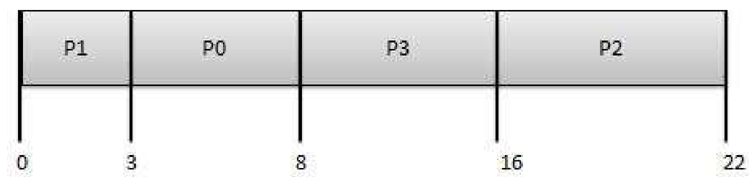
Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.55

Shortest Job First (SJF)

- Best approach to minimize waiting time.
- Impossible to implement
- Processer should know in advance how much time process will take.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	3
P2	2	8	8
P3	3	6	16



Wait time of each process is following

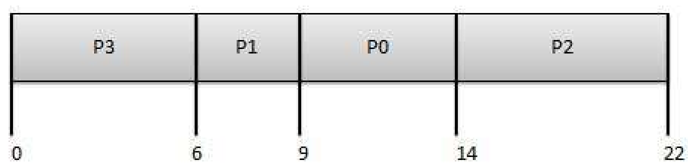
Process	Wait Time : Service Time - Arrival Time
P0	3 - 0 = 3
P1	0 - 0 = 0
P2	16 - 2 = 14
P3	8 - 3 = 5

Average Wait Time: (3+0+14+5) / 4 = 5.50

Priority Based Scheduling

- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first serve basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	3
P2	2	8	1	8
P3	3	6	3	16



Wait time of each process is following

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	3 - 1 = 2
P2	8 - 2 = 6
P3	16 - 3 = 13

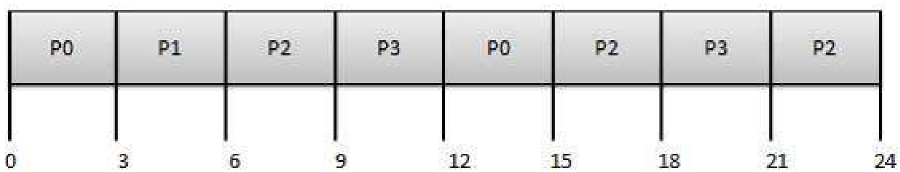
Average Wait Time: (0+2+6+13) / 4 = 5.25

Round Robin Scheduling

- Each process is provided a fix time to execute called quantum.
- Once a process is executed for given time period. Process is preempted and other process executes for given time period.
- Context switching is used to save states of preempted processes.

Process	Arrival Time	Execute Time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

Quantum = 3



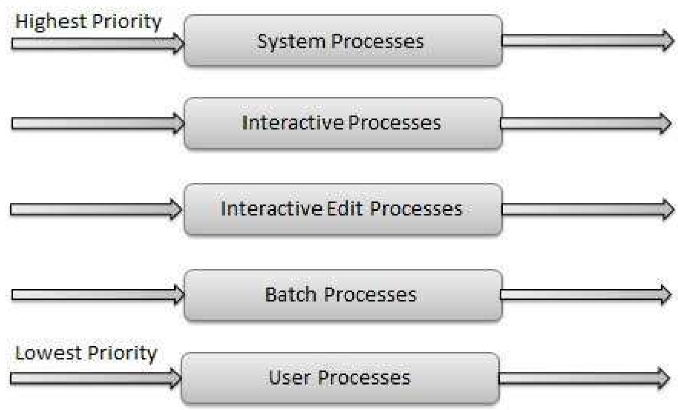
Wait time of each process is following

Process	Wait Time : Service Time - Arrival Time
P0	(0-0) + (12-3) = 9
P1	(3-1) = 2
P2	6-2) + (15-9) = 10
P3	(9-3) + (18-12) = 12

Average Wait Time: (9+2+10+12) / 4 = 8.25

Multi Queue Scheduling

- Multiple queues are maintained for processes.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.



CHAPTER

8

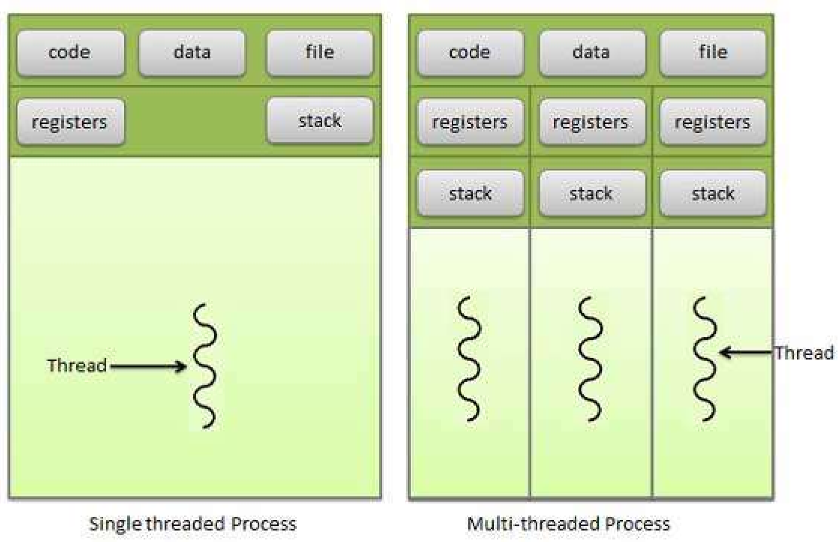
Operating System Multi-Threading

This section describes thread, types of threads and various thread models.

What is Thread?

A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows the working of the single and multithreaded processes.



Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight taking lesser resources than a process.
1	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
1	In multiple processing environments each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
1	If one process is blocked then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, second thread in the same task can run.
1	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
1	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

- Thread minimizes context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- Economy- It is more economical to create and context switch threads.
- Utilization of multiprocessor architectures to a greater scale and efficiency.

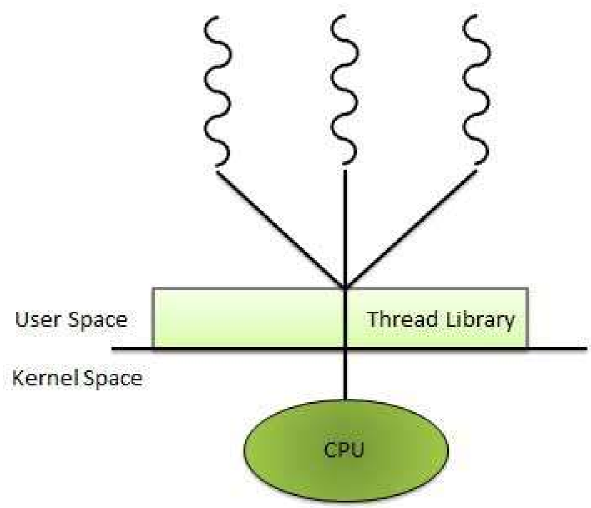
Types of Thread

Threads are implemented in following two ways

- **User Level Threads** -- User managed threads
- **Kernel Level Threads** -- Operating System managed threads acting on kernel, an operating system core.

User Level Threads

In this case, application manages thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application begins with a single thread and begins running in that thread.



Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads

In this case, thread management done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals' threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within same process requires a mode switch to the Kernel.

Multithreading Models

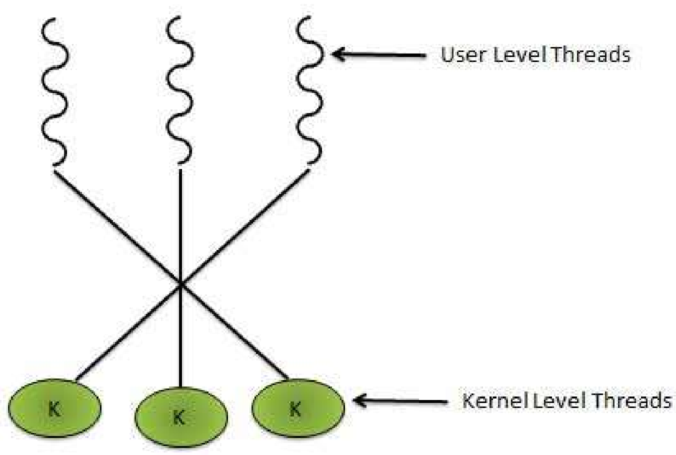
Some operating system provides a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

Many to Many Model

In this model, many user level threads multiplexes to the Kernel thread of smaller or equal numbers. The number of Kernel threads may be specific to either a particular application or a particular machine.

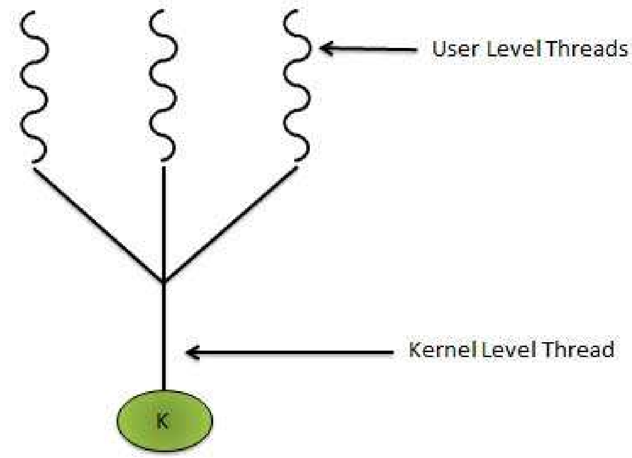
Following diagram shows the many to many model. In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallels on a multiprocessor.



Many to One Model

Many to one model maps many user level threads to one Kernel level thread. Thread management is done in user space. When thread makes a blocking system call, the entire process will be blocks. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

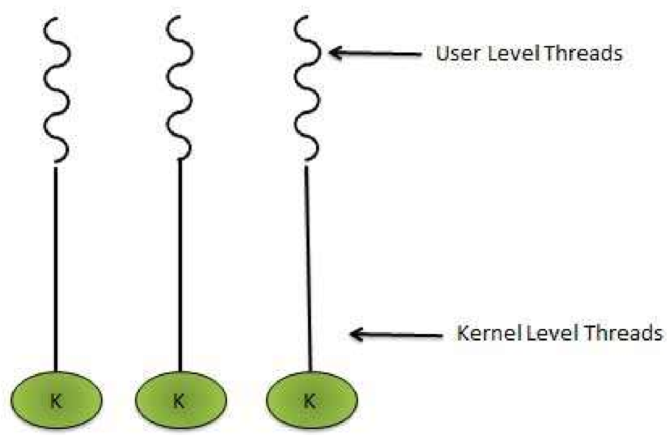
If the user level thread libraries are implemented in the operating system in such a way that system does not support them then Kernel threads use the many to one relationship modes.



One to One Model

There is one to one relationship of user level thread to the kernel level thread. This model provides more concurrency than the many to one model. It also another thread to run when a thread makes a blocking system call. It support multiple thread to execute in parallel on microprocessors.

Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, Windows NT and windows 2000 use one to one relationship model.



Difference between User Level & Kernel Level Thread

S.N.	User Level Threads	Kernel Level Thread
1	User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User level thread is generic and can run on any operating system.	Kernel level thread is specific to the operating system.
4	Multi-threaded application cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.

CHAPTER

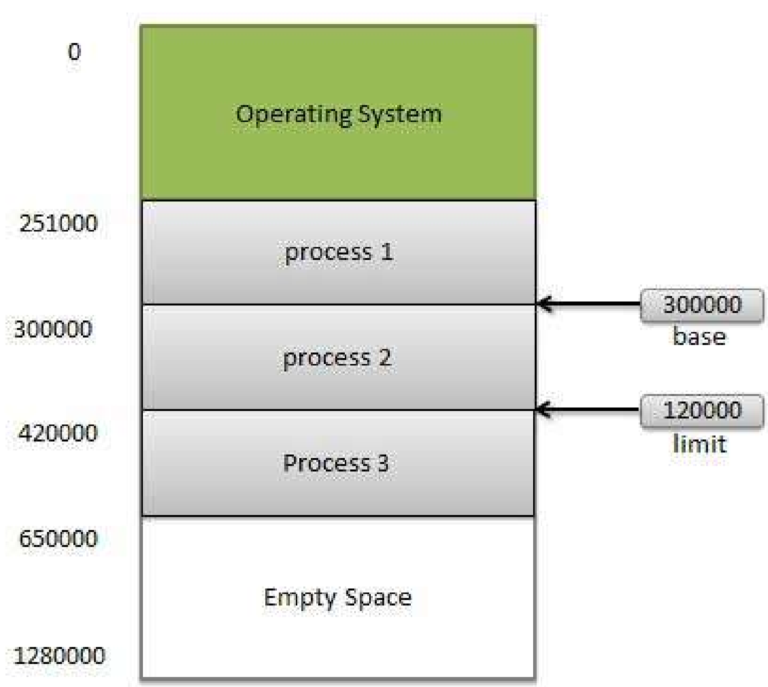
9

Memory Management

This section describes memory management techniques, logical v/s actual address space and various paging techniques.

Memory management is the functionality of an operating system which handles or manages primary memory. Memory management keeps track of each and every memory location either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the limit register is 1209000, then the program can legally access all addresses from 300000 through 411999.



Instructions and data to memory addresses can be done in following ways

- **Compile time** -- When it is known at compile time where the process will reside, compile time binding is used to generate the absolute code.

- **Load time** -- When it is not known at compile time where the process will reside in memory, then the compiler generates re-locatable code.

- **Execution time** -- If the process can be moved during its execution from one memory segment to another, then binding must be delayed to be done at run time

Dynamic Loading

In dynamic loading, a routine of a program is not loaded until it is called by the program. All routines are kept on disk in a re-locatable load format. The main program is loaded into memory and is executed. Other routines methods or modules are loaded on request. Dynamic loading makes better memory space utilization and unused routines are never loaded.

Dynamic Linking

Linking is the process of collecting and combining various modules of code and data into a executable file that can be loaded into memory and executed. Operating system can link system level libraries to a program. When it combines the libraries at load time, the linking is called static linking and when this linking is done at the time of execution, it is called as dynamic linking.

In static linking, libraries linked at compile time, so program code size becomes bigger whereas in dynamic linking libraries linked at execution time so program code size remains smaller.

Logical versus Physical Address Space

An address generated by the CPU is a logical address whereas address actually available on memory unit is a physical address. Logical address is also known a Virtual address.

Virtual and physical addresses are the same in compile-time and load-time address-binding schemes. Virtual and physical addresses differ in execution-time address-binding scheme.

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

The run-time mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address.

- The value in the base register is added to every address generated by a user process which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.

- The user program deals with virtual addresses; it never sees the real physical addresses.

Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory to a backing store, and then brought back into memory for continued execution.

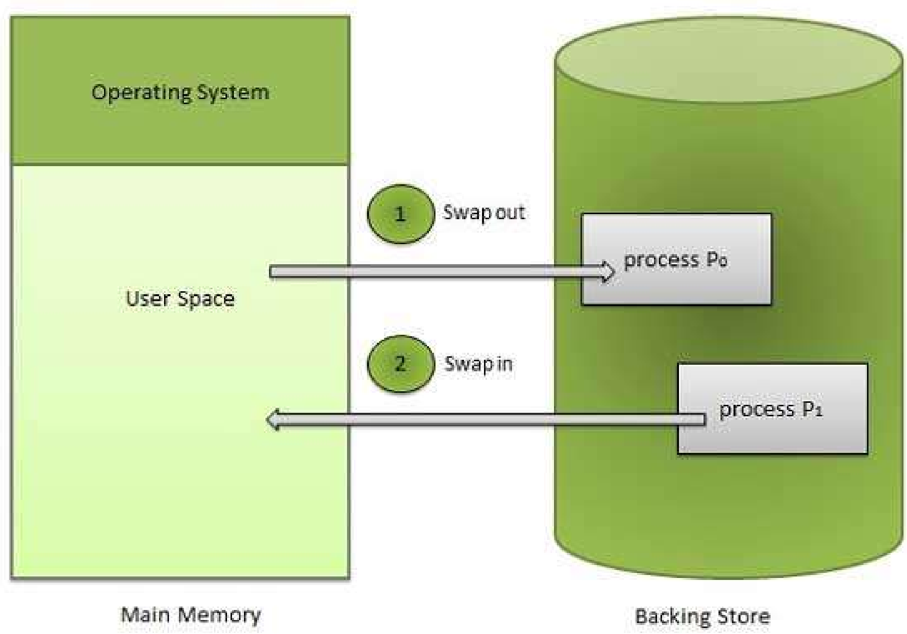
Backing store is a usually a hard disk drive or any other secondary storage which fast in access and large enough to accommodate copies of all memory images for all users. It must be capable of providing direct access to these memory images.

Major time consuming part of swapping is transfer time. Total transfer time is directly proportional to the amount of memory swapped. Let us assume that the user process is of size 100KB and the backing store is a standard hard disk with transfer rate of 1 MB per second. The actual transfer of the 100K process to or from memory will take

100KB / 1000KB per second

= 1/10 second

= 100 milliseconds



Memory Allocation

Main memory usually has two partitions

- **Low Memory** -- Operating system resides in this memory.
- **High Memory** -- User processes then held in high memory.

Operating system uses the following memory allocation mechanism.

S.N.	Memory Allocation	Description
1	**Single-partition allocation**	In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. Each logical address must be less than the limit register.
2	**Multiple-partition allocation**	In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. When a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

Fragmentation

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Fragmentation is of two types

S.N.	Fragmentation	Description
1	**External fragmentation**	Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it cannot be used.
2	**Internal fragmentation**	Memory block assigned to process is bigger. Some portion of memory is left unused as it cannot be used by another process.

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

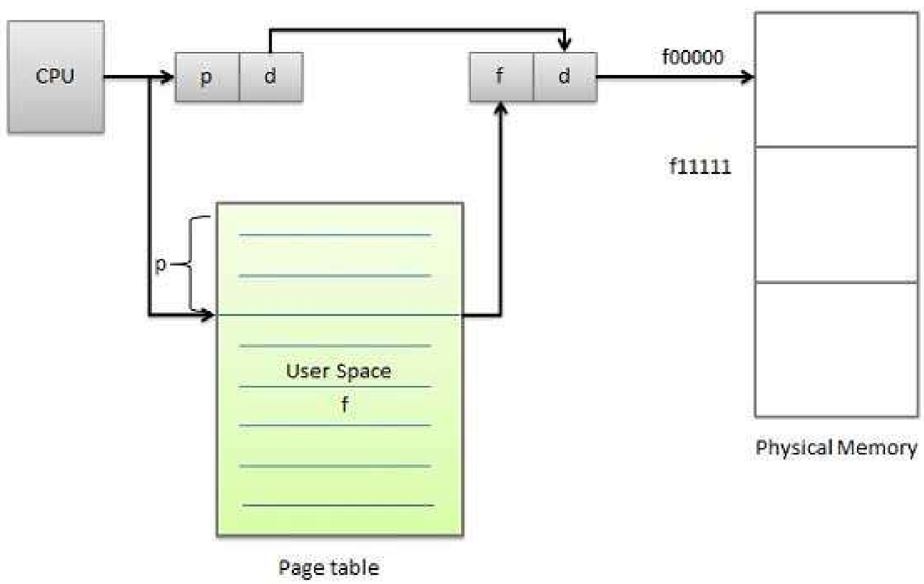
Paging

External fragmentation is avoided by using paging technique. Paging is a technique in which physical memory is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). When a process is to be executed, it's corresponding pages are loaded into any available memory frames.

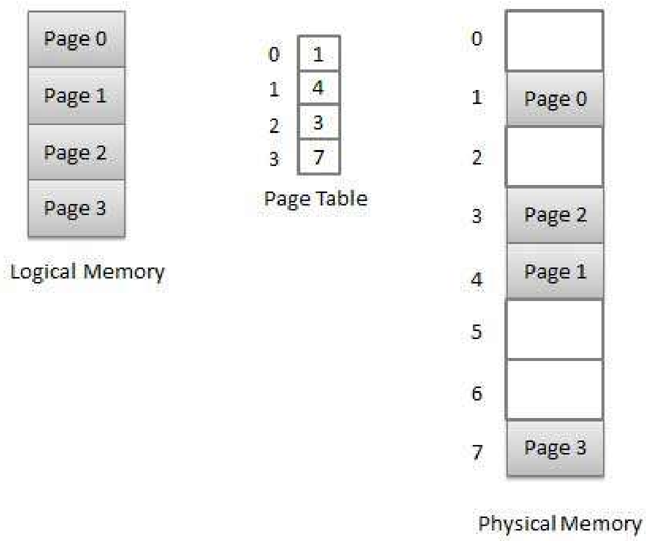
Logical address space of a process can be non-contiguous and a process is allocated physical memory whenever the free memory frame is available. Operating system keeps track of all free frames. Operating system needs n free frames to run a program of size n pages.

Address generated by CPU is divided into

- **Page number (p)** -- page number is used as an index into a page table which contains base address of each page in physical memory.
- **Page offset (d)** -- page offset is combined with base address to define the physical memory address.



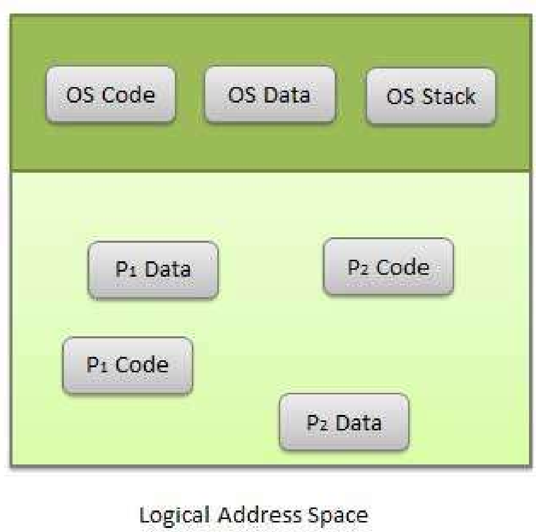
Following figure show the paging table architecture.



Segmentation

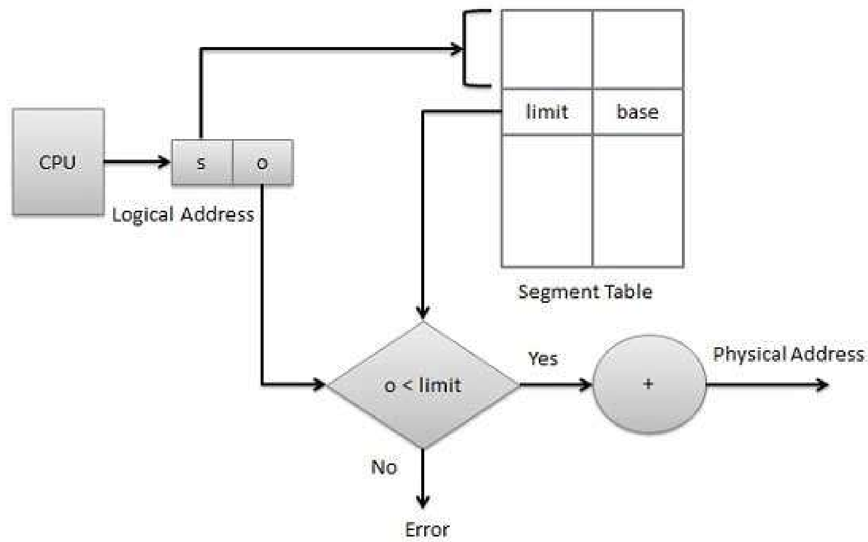
Segmentation is a technique to break memory into logical pieces where each piece represents a group of related information. For example, data segments or code segment for each process, data segment for operating system and so on. Segmentation can be implemented using or without using paging.

Unlike paging, segment is having varying sizes and thus eliminates internal fragmentation. External fragmentation still exists but to lesser extent.



Address generated by CPU is divided into

- **Segment number (s)** -- segment number is used as an index into a segment table which contains base address of each segment in physical memory and a limit of segment.
- **Segment offset (o)** -- segment offset is first checked against limit and then is combined with base address to define the physical memory address.



CHAPTER

10

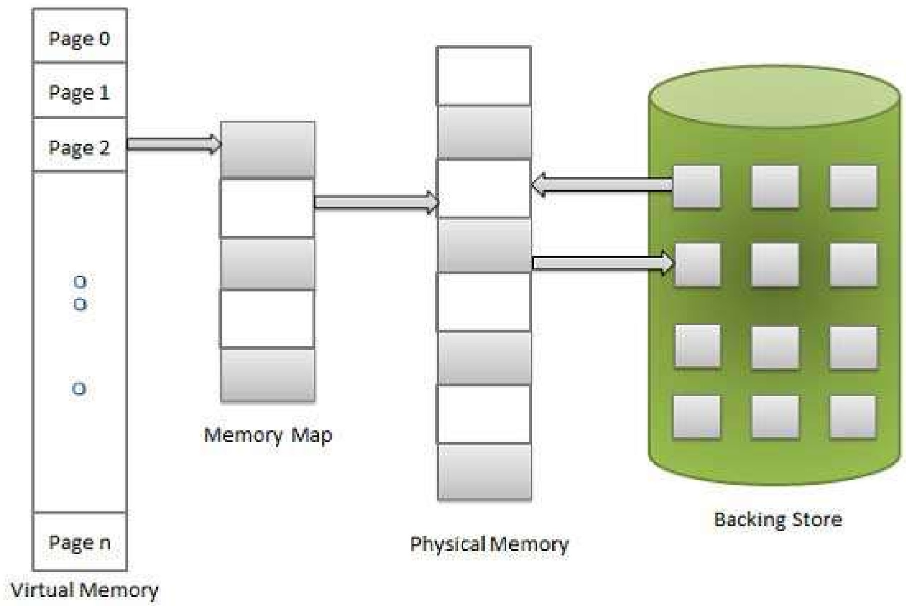
Virtual Memory

This section describes concepts of virtual memory, demand paging and various page replacement algorithms.

Virtual memory is a technique that allows the execution of processes which are not completely available in memory. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory is the separation of user logical memory from physical memory.

This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occured in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.



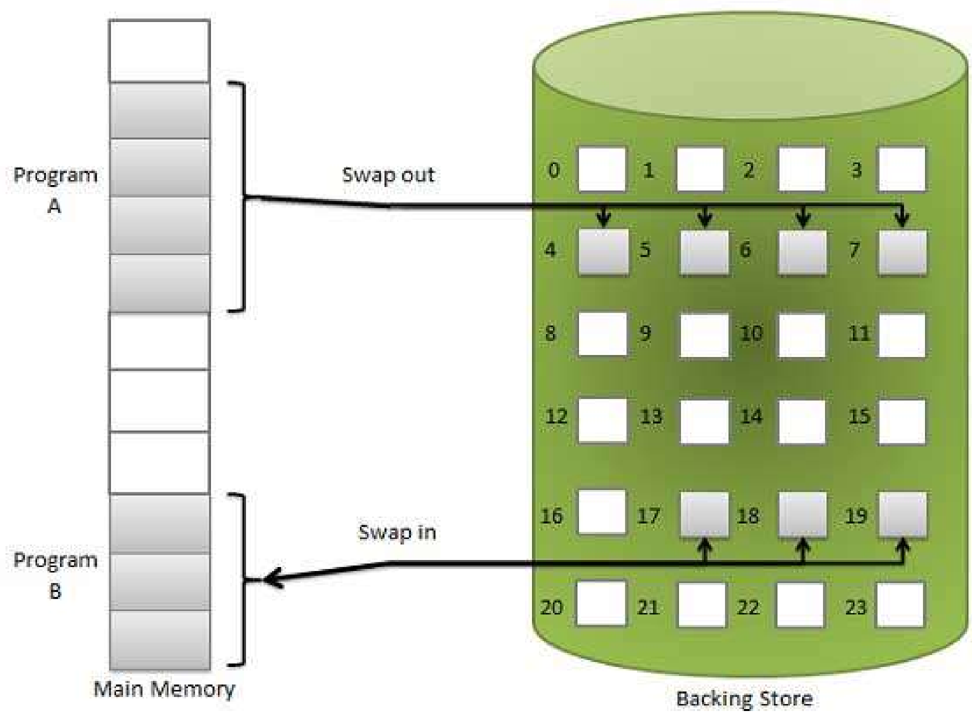
Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Demand Paging

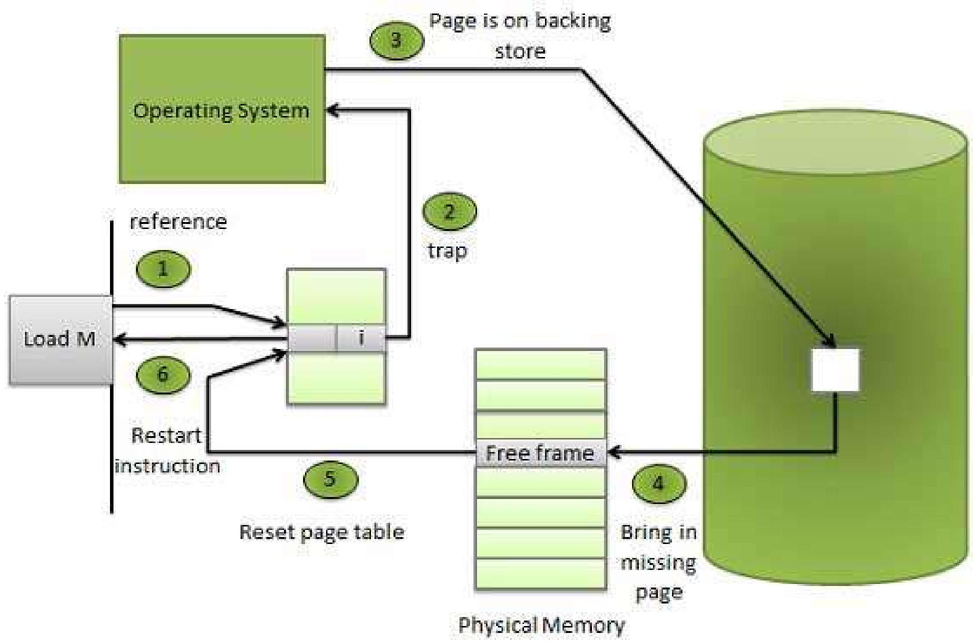
A demand paging system is quite similar to a paging system with swapping. When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, however, we use a lazy swapper called pager.

When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again. Instead of swapping in a whole process, the pager brings only those necessary pages into memory. Thus, it avoids reading into memory pages that will not be used in anyway, decreasing the swap time and the amount of physical memory needed.

Hardware support is required to distinguish between those pages that are in memory and those pages that are on the disk using the valid-invalid bit scheme. Where valid and invalid pages can be checked by checking the bit. Marking a page will have no effect if the process never attempts to access the page. While the process executes and accesses pages that are memory resident, execution proceeds normally.



Access to a page marked invalid causes a **page-fault trap**. This trap is the result of the operating system's failure to bring the desired page into memory. But page fault can be handled as following



Step	Description
Step 1	Check an internal table for this process, to determine whether the reference was a valid or it was an invalid memory access.
Step 2	If the reference was invalid, terminate the process. If it was valid, but page have not yet brought in, page in the latter.
Step 3	Find a free frame.
Step 4	Schedule a disk operation to read the desired page into the newly allocated frame.
Step 5	When the disk read is complete, modify the internal table kept with the process and the page table to indicate that the page is now in memory.
Step 6	Restart the instruction that was interrupted by the illegal address trap. The process can now access the page as though it had always been in memory. Therefore, the operating system reads the desired page into memory and restarts the process as though the page had always been in memory.

Advantages

Following are the advantages of Demand Paging

- Large virtual memory.
- More efficient use of memory.
- Unconstrained multiprogramming. There is no limit on degree of multiprogramming.

Disadvantages

Following are the disadvantages of Demand Paging

- Number of tables and amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.
- Due to the lack of explicit constraints on jobs address space size.

Page Replacement Algorithm

Page replacement algorithms are the techniques using which Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again then it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm. A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults.

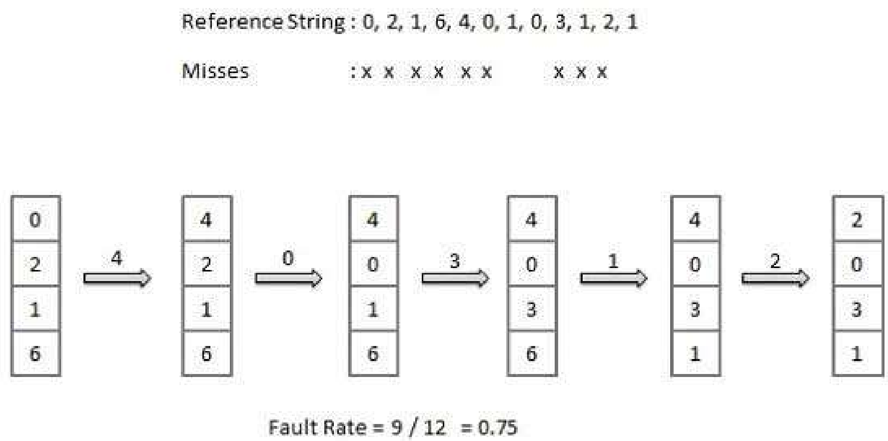
Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

- For a given page size we need to consider only the page number, not the entire address.
- If we have a reference to a page p, then any immediately following references to page p will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses - 123,215,600,1234,76,96
- If page size is 100 then the reference string is 1,2,6,12,0,0

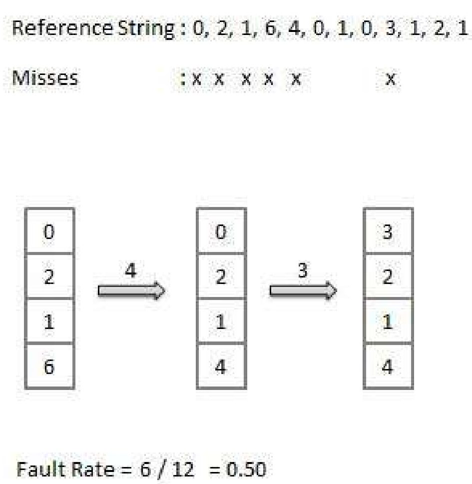
First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.



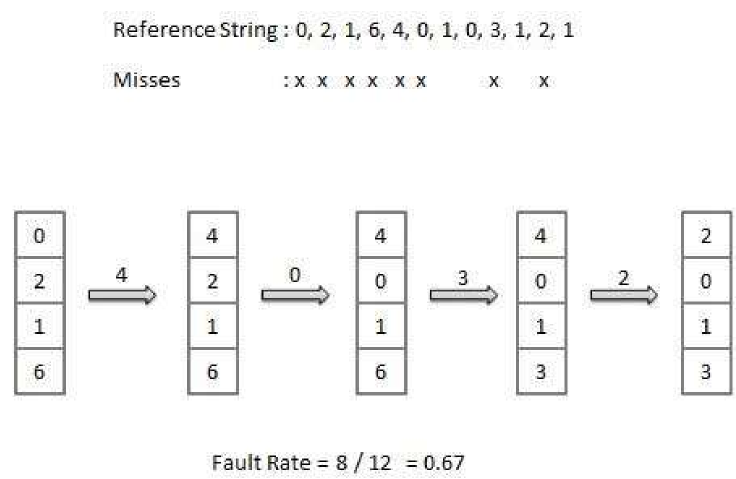
Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time . Use the time when a page is to be used.



Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.



Page Buffering algorithm

- To get process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used (LFU) algorithm

- Page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

Most frequently Used (LFU) algorithm

- This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

CHAPTER

11

I/O Hardware

This section describes I/O devices, Direct Memory Access (DMA), device controllers and ports.

Overview

Computers operate on many kinds of devices. General types include *storage devices* (disks, tapes), *transmission devices* (network cards, modems), and *human-interface devices* (screen, keyboard, mouse). Other devices are more specialized. A device communicates with a computer system by sending signals over a cable or even through the air.

The device communicates with the machine via a connection point termed a *port* (for example, a serial port). If one or more devices use a common set of wires, the connection is called a *bus*. In other terms, a bus is a set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.

Daisy chain

When device A has a cable that plugs into device B, and device B has a cable that plugs into device C, and device C plugs into a port on the computer, this arrangement is called a **daisy chain**. It usually operates as a bus.

Controller

A controller is a collection of electronics that can operate a port, a bus, or a device. A serial-port controller is an example of a simple device controller. This is a single chip in the computer that controls the signals on the wires of a serial port.

The SCSI bus controller is often implemented as a separate circuit board (a host adapter) that plugs into the computer. It contains a processor, microcode, and some private memory to enable it to process the SCSI protocol messages. Some devices have their own built-in controllers.

I/O port

An I/O port typically consists of four registers, called the **status, control**, **data-in**, and **data-out** registers.

S.N.	Register & Description
1	**Status Register** The status register contains bits that can be read by the host. These bits indicate states such as whether the current command has completed, whether a byte is available to be read from the data-in register, and whether there has been a device error.
2	**Control register** The control register can be written by the host to start a command or to change the mode of a device. For instance, a certain bit in the control register of a serial port chooses between full-duplex and half-duplex communication, another enables parity checking, a third bit sets the word length to 7 or 8 bits, and other bits select one of the speeds supported by the serial port.
3	**Data-in register** The data-in register is read by the host to get input.
4	**Data-out register** The data out register is written by the host to send output.

Polling

Polling is a process by which a host waits for controller response. It is a looping process, reading the status register over and over until the busy bit of status register becomes clear. The controller uses/sets the busy bit when it is busy working on a command, and clears the busy bit when it is ready to accept the next command. The host signals its wish via the command-ready bit in the command register. The host sets the command-ready bit when a command is available for the controller to execute.

In the following example, the host writes output through a port, coordinating with the controller by handshaking

- The host repeatedly reads the busy bit until that bit becomes clear.
- The host sets the write bit in the command register and writes a byte into the data-out register.
- The host sets the command-ready bit.
- When the controller notices that the command-ready bit is set, it sets the busy bit.
- The controller reads the command register and sees the write command.
- It reads the data-out register to get the byte, and does the I/O to the device.
- The controller clears the command-ready bit, clears the error bit in the status register to indicate that the device I/O succeeded, and clears the busy bit to indicate that it is finished.

I/O devices

I/O Devices can be categorized into following category.

S.N.	Category & Description
1	**Human readable** Human Readable devices are suitable for communicating with the computer user. Examples are printers, video display terminals, keyboard etc.
2	**Machine readable** Machine Readable devices are suitable for communicating with electronic equipment. Examples are disk and tape drives, sensors, controllers and actuators.
2	**Communication** Communication devices are suitable for communicating with remote devices. Examples are digital line drivers and modems.

Following are the differences between I/O Devices

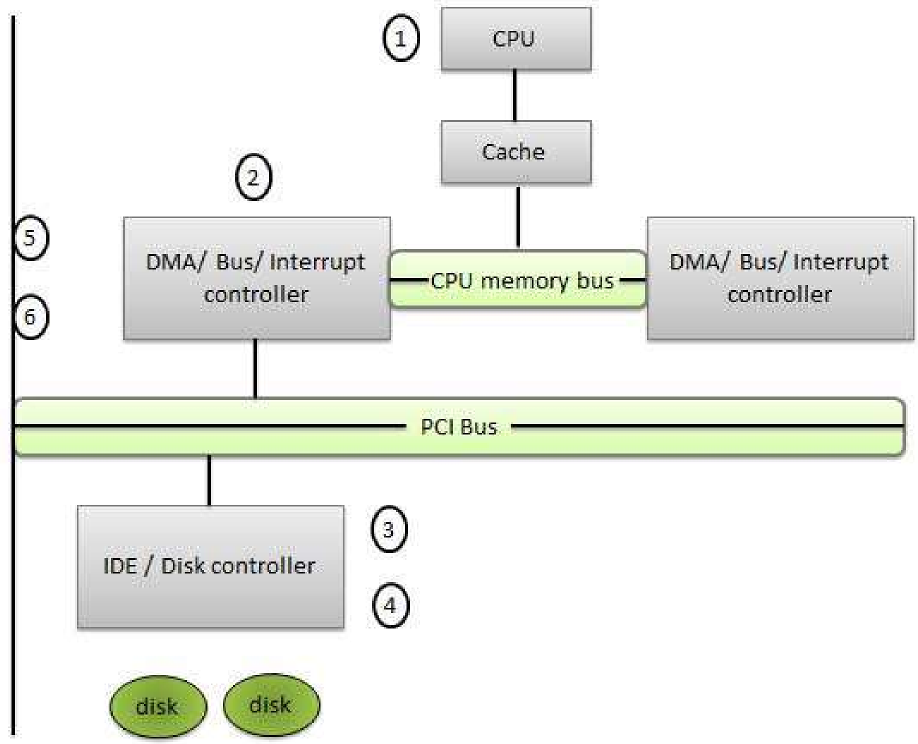
S.N.	Criteria & Description
1	**Data rate** There may be differences of several orders of magnitude between the data transfer rates.
2	**Application** Different devices have different use in the system.
3	**Complexity of Control** A disk is much more complex whereas printer requires simple control interface.
4	**Unit of transfer** Data may be transferred as a stream of bytes or characters or in larger blocks.
5	**Data representation** Different data encoding schemes are used for different devices.
6	**Error Conditions** The nature of errors differs widely from one device to another.

Direct Memory Access (DMA)

Many computers avoid burdening the main CPU with programmed I/O by offloading some of this work to a special purpose processor. This type of processor is called, a Direct Memory Access (DMA) controller. A special control unit is used to transfer block of data directly between an external device and the main memory, without intervention by the processor. This approach is called Direct Memory Access (DMA).

DMA can be used with either polling or interrupt software. DMA is particularly useful on devices like disks, where many bytes of information can be transferred in single I/O operations. When used with an interrupt, the CPU is notified only after the entire block of data has been transferred. For each byte or word transferred, it must provide the memory address and all the bus signals controlling the data transfer. Interaction with a device controller is managed through a device driver.

Handshaking is a process between the DMA controller and the device controller. It is performed via wires using terms DMA request and DMA acknowledge.



Step	Description
1	Device driver is instructed to transfer disk data to a buffer address X.
2	Device driver then instruct disk controller to transfer data to buffer.
3	Disk controller starts DMA transfer.

4	Disk controller sends each byte to DMA controller.
5	DMA controller transfers bytes to buffer, increases the memory address, decreases the counter C until C becomes zero.
6	When C becomes zero, DMA interrupts CPU to signal transfer completion.

Device Controllers

A computer system contains a many types of I/O devices and their respective controllers

- network card
- graphics adapter
- disk controller
- DVD-ROM controller
- serial port
- USB
- sound card

CHAPTER

12

I/O Software

This section describes interrupts, application I/O interface, Kernel Subsystem and device driver.

Interrupts

The CPU hardware uses an interrupt request line wire which helps CPU to sense after executing every instruction. When the CPU checks that a controller has put a signal on the interrupt request line, the CPU saves a state, such as the current value of the instruction pointer, and jumps to the interrupt handler routine at a fixed address. The interrupt handler part determines the cause of the interrupt performs the necessary processing and executes an interrupt instruction to return the CPU to its execution state.

The basic mechanism of interrupt enables the CPU to respond to an asynchronous event, such as when a device controller becomes ready for service. Most CPUs have two interrupt request lines.

- **non-maskable interrupt** - Such kind of interrupts are reserved for events like unrecoverable memory errors.
- **maskable interrupt** - Such interrupts can be switched off by the CPU before the execution of critical instructions that must not be interrupted.

The interrupt mechanism accepts an address - a number that selects a specific interrupt handling routine/function from a small set. In most architecture, this address is an offset stored in a table called the interrupt vector table. This vector contains the memory addresses of specialized interrupt handlers.

Application I/O Interface

Application I/O Interface represents the structuring techniques and interfaces for the operating system to enable I/O devices to be treated in a standard, uniform way. The actual differences lies kernel level modules called device drivers which are custom tailored to corresponding devices but show one of the standard interfaces to applications. The purpose of the device-driver layer is to hide the differences among device controllers from the I/O subsystem of the kernel, such as the I/O system calls. Following are the characteristics of I/O interfaces with respected to devices.

- **Character-stream / block** - A character-stream device transfers bytes in one by one fashion, whereas a block device transfers a complete unit of bytes.

- **Sequential / random-access** - A sequential device transfers data in a fixed order determined by the device, random-access device can be instructed to seek position to any of the available data storage locations.

- **Synchronous / asynchronous** - A synchronous device performs data transfers with known response time where as an asynchronous device shows irregular or unpredictable response time.

- **Sharable / dedicated** - A sharable device can be used concurrently by several processes or threads but a dedicated device cannot be used.

- **Speed of operation** - Device speeds may range from a few bytes per second to a few gigabytes per second.

- **Read-write, read only, or write only** - Some devices perform both input and output, but others support only one data direction that is read only.

Clocks

Clocks are also called timers. The clock software takes the form of a device driver though a clock is neither a blocking device nor a character based device. The clock software is the clock driver. The exact function of the clock driver may vary depending on operating system. Generally, the functions of the clock driver include the following.

S.N.	Task	Description
1	Maintaining the time of the day	The clock driver implements the time of day or the real time clock function. It requires incrementing a counter at each clock tick.
2	Preventing processes from running too long	As a process is started, the scheduler initializes the quantum counter in clock ticks for the process. The clock driver decrements the quantum counter by 1, at every clock interrupts. When the counter gets to zero, clock driver calls the scheduler to set up another process. Thus clock driver helps in preventing processes from running longer than time slice allowed.
3	Accounting for CPU usage	Another function performed by clock driver is doing CPU accounting. CPU accounting implies telling how long the process has run.
4	Providing watchdog timers for parts of the system itself	Watchdog timers are the timers set by certain parts of the system. For example, to use a floppy disk, the system must turn on the motor and then wait about 500msec for it to come up to speed.

Kernel I/O Subsystem

Kernel I/O Subsystems responsible to provide many services related to I/O. Following are some of the services provided.

- **Scheduling** - Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.

- **Buffering** - Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a devices with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

- **Caching** - Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.

- **Spooling and Device Reservation** - A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.

- **Error Handling** - An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Device driver

Device driver is a program or routine developed for an I/O device. A device driver implements I/O operations or behaviors on a specific class of devices. For example a system supports one or a number of multiple brands of terminals, all slightly different terminals may have a single terminal driver. In the layered structure of I/O system, device driver lies between interrupt handler and device independent I/O software. The job of a device driver is following.

- To accept request from the device independent software above it.
- To see to it that the request is executed.

How a device driver handles a request is as follows: Suppose a request comes to read a block N. If the driver is idle at the time a request arrives, it starts carrying out the request immediately. Otherwise, if the driver is already busy with some other request, it places the new request in the queue of pending requests.

CHAPTER

13

File System

This section describes file, file types, file access mechanisms and space allocation techniques.

File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

File structure is a structure, which is according to a required format that operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. UNIX, MS-DOS support minimum number of file structure.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX has the following types of files:

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files:

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types

- **Character special files** - data is handled character by character as in case of terminals or printers.
- **Block special files** - data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence i.e. the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Space Allocation

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

Contiguous Allocation

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement.
- External fragmentation is a major issue with this type of allocation technique.

Linked Allocation

- Each file carries a list of links to disk blocks.
- Directory contains link / pointer to first block of a file.
- No external fragmentation
- Effectively used in sequential access file.
- Inefficient in case of direct access file.

Indexed Allocation

- Provides solutions to problems of contiguous and linked allocation.
- A index block is created having all pointers to files.
- 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.

CHAPTER

14

Operating System Security

This section describes various security related aspects like authentication, one time password, threats and security classifications.

Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by unauthorized user then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc. We're going to discuss following topics in this article.

- Authentication
- One Time passwords
- Program Threats
- System Threats
- Computer Security Classifications

Authentication

Authentication refers to identifying the each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways:

- **Username / Password** - User need to enter a registered username and password with Operating system to login into the system.

- **User card/key** - User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.

- **User attribute - fingerprint/ eye retina pattern/ signature** - User need to pass his/her attribute via designated input device used by operating system to login into the system.

One Time passwords

One time passwords provides additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used then it cannot be used again. One time password are implemented in various ways.

- **Random numbers** - Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.

- **Secret key** - User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.

- **Network password** - Some commercial applications send one time password to user on registered mobile/ email which is required to be entered prior to login.

Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks then it is known as Program Threats. One of the common examples of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan horse** - Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** - If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** - Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** - Virus as name suggests can replicate them on computer system .They are highly dangerous and can modify/delete user files, crash systems. A virus is generally a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user.

System Threats

System threats refer to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats create such an environment that operating system resources/ user files are mis-used. Following is the list of some well-known system threats.

- **Worm** -Worm is a process which can choke down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worm processes can even shut down an entire network.

- **Port Scanning** - Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.

- **Denial of Service** - Denial of service attacks normally prevents user to make legitimate use of the system. For example user may not be able to use internet if denial of service attacks browser's content settings.

Computer Security Classifications

As per the U.S. Department of Defense Trusted Computer System's Evaluation Criteria there are four security classifications in computer systems: A, B, C, and D. This is widely used specifications to determine and model the security of systems and of security solutions. Following is the brief description of each classification.

S.N.	Classification Type	Description
1	Type A	Highest Level. Uses formal design specifications and verification techniques. Grants a high degree of assurance of process security.
2	Type B	Provides mandatory protection system. Have all the properties of a class C2 system. Attaches a sensitivity label to each object. It is of three types. • **B1** - Maintains the security label of each object in the system. Label is used for making decisions to access control. • **B2** - Extends the sensitivity labels to each system resource, such as storage objects, supports covert channels and auditing of events. • **B3** - Allows creating lists or user groups for access-control to grant access or revoke access to a given named object.
3	Type C	Provides protection and user accountability using audit capabilities. It is of two types. • **C1** - Incorporates controls so that users can protect their private information and keep other users from accidentally reading / deleting their data. UNIX versions are mostly Cl class. • **C2** - Adds an individual-level access control to the capabilities of a Cl level system
4	Type D	Lowest level. Minimum protection. MS-DOS, Window 3.1 fall in this category.

CHAPTER

15

Linux Operating System

This section describes Linux operating system's component and its functioning.

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

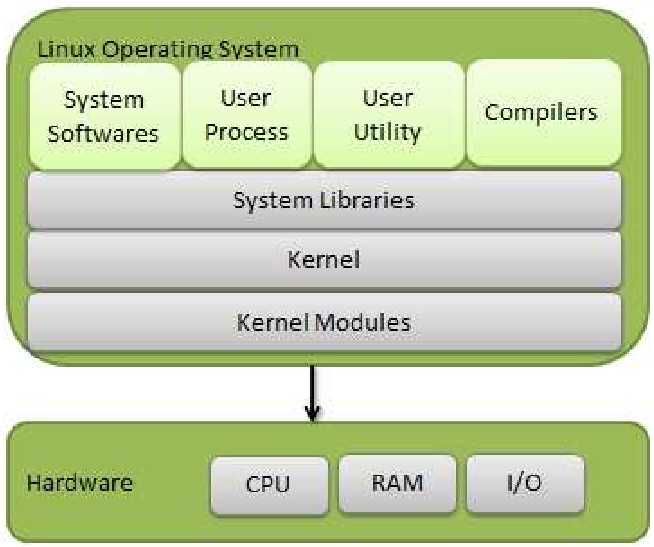
Components of Linux System

Linux Operating System has primarily three components

- **Kernel** - Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It is consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

- **System Library** - System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implements most of the functionalities of the operating system and do not requires kernel module's code access rights.

- **System Utility** - System Utility programs are responsible to do specialized, individual level tasks.



Kernel Mode v/s User Mode

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each process and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Basic Features

Following are some of the important features of Linux Operating System.

- **Portable** - Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.

- **Open Source** - Linux source code is freely available and it is community based development project. Multiple teams works in collaboration to enhance the capability of Linux operating system and it is continuously evolving.

- **Multi-User** - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.

- **Multiprogramming** - Linux is a multiprogramming system means multiple applications can run at same time.

- **Hierarchical File System** - Linux provides a standard file structure in which system files/ user files are arranged.

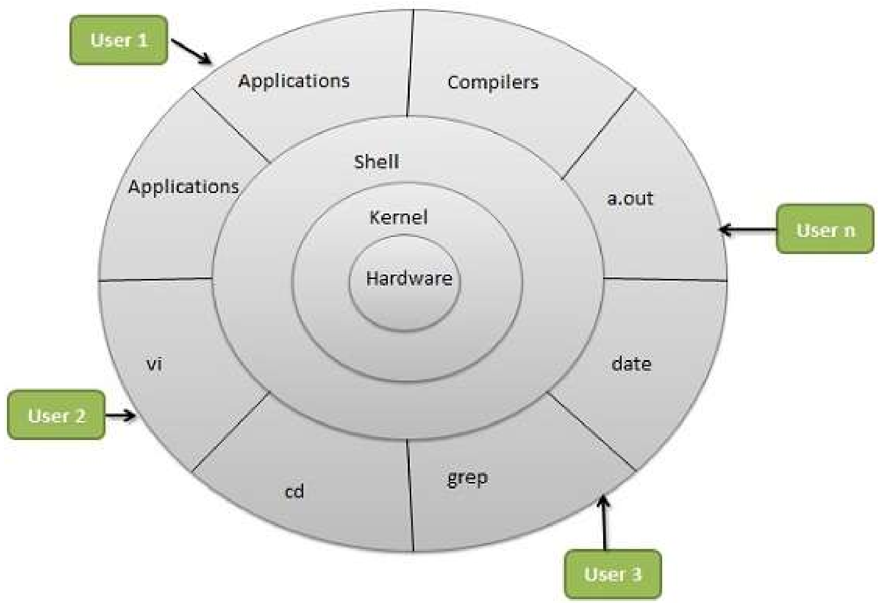
- **Shell** - Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.

- **Security** - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Architecture

Linux System Architecture is consists of following layers

- **Hardware layer** - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
- **Kernel** - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.
- **Utilities** - Utility programs giving user most of the functionalities of an operating systems.



APPENDIX

A

About tutorialspoint.com

Tutorials Point is not a commercial site, this site has been created just for educational purposes and to help the people who are enthusiastic to learn new technologies....

Tutorials Point is aiming to provide the Best Training Materials on highly demanding technical and managerial subjects like:

- Python
- Ruby
- HTML5
- CSS
- JavaScript and related frameworks
- Ruby on Rails
- JAVA and related technologies
- PMP Exams
- Earned Value Management
- Six Sigma
- Parrot
- AJAX
- PHP Programming
- HTML and XHTML
- CGI and Perl
- C Programming
- C++ Programming
- XML-RPC
- SOAP Communication
- HTTP Protocol
- Unix Makefile
- Web Services
- WSDL and UDDI
- Wi-Fi and WiMAX
- Many more...

TutorialsPoint is a FREE site and will remain FREE in future as well.....If you think it is worth to visit this website, kindly share it with your friends and colleagues.

Please send us your feedback at webmaster@tutorialspoint.com

Thanks for visiting us.