

**Scope:** The scope of this course is to teach students the capabilities and limitations of computer operating systems, process management, processor scheduling, deadlocks, memory management, secondary memory management, file management and I/O systems.

**Objectives:** To make student familiar with the memory allocation methods, page replacement algorithms, file allocation methods, multi-threading, process synchronization, and CPU scheduling

### UNIT I

Introduction -Mainframe systems Desktop Systems – Multiprocessor systems – distributed systems – real time systems. Process: - Process concepts – Operation on process – cooperation process - Inter process Communication - Mutual Exclusion - Critical sections- primitives – Semaphores – Deadlock: System Model, Deadlock characterization, Deadlock prevention, avoidance, detection, recovery from deadlock.

### UNIT II

Storage management: Memory Management - swapping- Contiguous memory allocation – paging, segmentation – segmentation with paging – Virtual memory :Virtual storage organization – Demand Paging, Process Creation – Page replacement – Thrashing.

### UNIT III

Processor Scheduling : preemptive scheduling : - Scheduling Criteria – Scheduling Algorithms – FCFS- SJF- Priority – RoundRobin –Multilevel Queue – Multilevel Feedback Queue .  
Multiprocess schedule: Real time schedule, Algorithm evaluation: Deterministic Modeling, Queue Model, Simulation

### UNIT IV

File systems: Introduction – File System Concepts – Access Methods – Directory structure – File Sharing – Allocation Methods – Free space management –Efficiency and performance – Recovery  
Disk Performance Optimization: Introduction – Disk structure – Disk scheduling – Disk management.

### UNIT V

Linux-The Operating System: Linux History, Linux features, Linux distributions, Linux's relationship to Unix, Overview of Linux Architecture, Installation, Start up scripts, system process (an overview), Linux Security, The Ext2 and Ext3 File Systems: General characteristics

of the Ext3 File System, File permissions, User Management: Types of users, the powers of Root, Managing users (adding and deleting) : using the command line and GUI Tools. Resource Management in Linux: File and Directory management, system calls for files process management, Signals, IPC:Pipes, FIFOs, System V IPC, Message Queues, System calls for processes, Memory Management, Library and System calls for Memory.

## **SUGGESTED READINGS**

### **TEXT BOOK**

1. Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

### **REFERENCES**

1. Deitel H.M. (2005). Operating systems, Third Edition, Addison Wesley Publication, New Delhi.
2. Pramod Chandra P. Bhatt. (2007). An Introduction to Operating Systems, Second Edition, Prentice Hall India, New Delhi.
3. Tanenbaum Woodhull. (2005) . Operating Systems., Second Edition, Pearson Education (LPE) , New Delhi.
4. William Stallings. (2010). Operating Systems internals and Design Principles, Sixth Edition, Prentice Hall India, New Delhi.
5. Arnold Robbins., (2008) ., Linux Programming by Examples The Fundamentals, Second Edition., Pearson Education.,.
6. Cox K, (2009).Red Hat Linux Administrator's Guide,PHI.
7. Stevens R., (2009). UNIX Network Programming, Third Edition.,PHI.
8. Sumitabha Das, (2009).Unix Concepts and Applications, Fourth Edition., TMH.
9. Ellen Siever, Stephen Figgins, Robert Love, Arnold Robbins, (2009) . Linux in a Nutshell, Sixth Edition,O'Reilly Media.
10. Neil Matthew, Richard Stones, Alan Cox,(2004) Beginning Linux Programming,Third Edition.

### **WEBSITES**

[www.cs.columbia.edu/~nieh/teaching/e6118\\_s00/](http://www.cs.columbia.edu/~nieh/teaching/e6118_s00/)  
[www.clarkson.edu/~jnm/cs644](http://www.clarkson.edu/~jnm/cs644)  
[pages.cs.wisc.edu/~remzi/Classes/736/Fall2002/](http://pages.cs.wisc.edu/~remzi/Classes/736/Fall2002/)

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
 (Deemed to be University Established under Section 3 of UGC Act 1956)  
 Pollachi Main Road, Eacharani Post, Coimbatore-641 021

**DEPARTMENT OF MATHEMATICS**  
**Semester – IV**  
**OPERATING SYSTEM: LINUX(17MMU404B)**  
**LESSON PLAN**  
**UNIT-1**

| S.NO                 | DURATION | TOPICS   | SUPPORTED MATERIALS       |
|----------------------|----------|--|---------------------------|
| 1                    | 1        | Introduction – MainFrame Systems , Desktop Systems , Multi Processor Systems | T1:3-13<br>W1<br>R1:10-25 |
| 2                    | 1        | Distributed Systems – Real Time Systems                                      | T1:14-19<br>W1            |
| 3                    | 1        | TUTORIAL-I   |                           |
| 4                    | 1        | Process: Concept – Operation On Process , Co-operation On Process            | T1:95-109<br>W1           |
| 5                    | 1        | Inter Process Communication , Mutual Exclusion , Critical Sections           | T1:119-125,191            |
| 6                    | 1        | TUTORIAL-II  |                           |
| 7                    | 1        | Primitives , Semaphores  | T1:201-205<br>W1          |
| 8                    | 1        | Deadlock: system Model, Deadlock Characterization                            | T1:243-247                |
| 9                    | 1        | TUTORIAL-III   |                           |
| 10                   | 1        | Deadlock Prevention , avoidance ,Detection ,Recovery                         | T1:250-264<br>W1          |
| 11                   | 1        | TUTORIAL-IV  |                           |
| 12                   | 1        | RECAPITULATION OF IMPORTANT QUESTIONS  |                           |
| TOTAL HOURS:12 HOURS |          |  |                           |

**TEXTBOOK :**

**T1:** Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

**REFERENCE BOOKS:**

**R1:** Deitel H.M. (2005). Operating systems, Third Edition, Addison Wesley Publication, New Delhi.

**WEBSITE:**

**W1:** [www.cs.columbia.edu/~nieh/teaching/e6118\\_s00/](http://www.cs.columbia.edu/~nieh/teaching/e6118_s00/)

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed to be University Established under Section 3 of UGC Act 1956)  
Pollachi Main Road, Eacharani Post, Coimbatore-641 021

**DEPARTMENT OF MATHEMATICS**  
**Semester – IV**  
**OPERATING SYSTEM: LINUX(17MMU404B)**  
**LESSON PLAN**  
**UNIT-II**

| S.NO | DURATION | TOPICS TO BE COVERED                             | SUPPORTED MATERIALS |
|------|----------|--|---------------------|
| 1    | 1        | Storage Management: Memory Management , Swapping | T1:274-283          |
| 2    | 1        | Contiguous Memory Allocation , Paging            | T1:283-286          |
| 3    | 1        | TUTORIAL-I                                       |                     |
| 4    | 1        | Segmentation: Segmentation with Paging           | T1:303-311<br>W1    |
| 5    | 1        | Virtual Memory : Virtual Storage Organization    | T1:317,318          |
| 6    | 1        | TUTORIAL-II                                      |                     |
| 7    | 1        | Demand Paging                                    | T1:320-327          |
| 8    | 1        | Process Creation                                 | T1:328-330          |
| 9    | 1        | TUTORIAL-III                                     |                     |
| 10   | 1        | Page Replacement , Thrashing                     | T1:330-350<br>W1    |
| 11   | 1        | TUTORIAL-IV                                      |                     |
| 12   | 1        | RECAPITULATION OF IMPORTANT QUESTIONS            |                     |
|      |          | TOTAL HOURS:12HOURS                              |                     |

**TEXTBOOK :**

**T1:** Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

**REFERENCE BOOKS:**

**R1:** Deitel H.M. (2005). Operating systems, Third Edition, Addison Wesley Publication, New Delhi.

**WEBSITE:**

**W1:** [www.cs.columbia.edu/~nieh/teaching/e6118\\_s00/](http://www.cs.columbia.edu/~nieh/teaching/e6118_s00/)

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
 (Deemed to be University Established under Section 3 of UGC Act 1956)  
 Pollachi Main Road, Eacharani Post, Coimbatore-641 021

**DEPARTMENT OF MATHEMATICS**  
**Semester – IV**  
**OPERATING SYSTEM: LINUX(17MMU404B)**  
**LESSON PLAN**  
**UNIT-III**

| S.NO | DURATION | TOPICS  | SUPPORTED MATERIALS     |
|------|----------|---|-------------------------|
| 1    | 1        | Processor Scheduling : Preemptive Scheduling , Scheduling Criteria , Scheduling Algorithm | T1:99-100,155-160<br>W1 |
| 2    | 1        | FCFS , SJF , Priority , Round Robin   | T1:160-163              |
| 3    | 1        | TUTORIAL-I  |                         |
| 4    | 1        | Multi Level Queue, Multi Level FeedBack Queue   | W1                      |
| 5    | 1        | Multi Processor Schedule: Real Time Schedule  | W1                      |
| 6    | 1        | TUTORIAL-II   |                         |
| 7    | 1        | Algorithm Evaluation : Deterministics Modelling   | T1:172-177<br>W1        |
| 8    | 1        | TUTORIAL-III  |                         |
| 9    | 1        | Queue Model ,Simulation   | W1                      |
| 10   | 1        | TUTORIAL-IV   |                         |
| 11   | 1        | RECAPITULATION OF IMPORTANT QUESTIONS   |                         |
|      |          | TOTAL HOURS:11 HOURS  |                         |

**TEXTBOOK :**

**T1:** Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

**REFERENCE BOOKS:**

**R1:** Deitel H.M. (2005). Operating systems, Third Edition, Addison Wesley Publication, New Delhi.

**WEBSITE:**

**W1:** [www.cs.columbia.edu/~nieh/teaching/e6118\\_s00/](http://www.cs.columbia.edu/~nieh/teaching/e6118_s00/)

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
 (Deemed to be University Established under Section 3 of UGC Act 1956)  
 Pollachi Main Road, Eacharani Post, Coimbatore-641 021

**DEPARTMENT OF MATHEMATICS**  
**Semester – IV**  
**OPERATING SYSTEM: LINUX(17MMU404B)**  
**LESSON PLAN**  
**UNIT-IV**

| S.NO | DURATION | TOPICS TO BE COVERED   | SUPPORTED MATERIALS      |
|------|----------|--|--------------------------|
| 1    | 1        | File System : Introduction , Concept , Access Methods                  | T1:371-378               |
| 2    | 1        | Directory Structure, File Sharing                                      | T1:379-396               |
| 3    | 1        | TUTORIAL-I   |                          |
| 4    | 1        | Allocation Methods , Frees Space Management                            | T1:396-400,430-432<br>W1 |
| 5    | 1        | Efficiency & Performance   | T1:433-437               |
| 6    | 1        | TUTORIAL-II  |                          |
| 7    | 1        | Recovery Disk Performance Optimization : Introduction , Disk Structure | T1:437-492<br>R1:510-515 |
| 8    | 1        | TUTORIAL-III   |                          |
| 9    | 1        | Disk Scheduling , Disk Management                                      | T1:492-502               |
| 10   | 1        | TUTORIAL-IV  |                          |
| 11   | 1        | RECAPITULATION OF IMPORTANT QUESTIONS                                  |                          |
|      |          | TOTAL HOURS:11HOURS  |                          |

**TEXTBOOK :**

**T1:** Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

**REFERENCE BOOKS:**

**R1:** Deitel H.M. (2005). Operating systems, Third Edition, Addison Wesley Publication, New Delhi.

**WEBSITE:**

**W1:** [www.cs.columbia.edu/~nieh/teaching/e6118\\_s00/](http://www.cs.columbia.edu/~nieh/teaching/e6118_s00/)

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed to be University Established under Section 3 of UGC Act 1956)  
Pollachi Main Road, Eacharani Post, Coimbatore-641 021

**DEPARTMENT OF MATHEMATICS**  
**Semester – IV**  
**OPERATING SYSTEM: LINUX(17MMU404B)**  
**LESSON PLAN**  
**UNIT-V**

| S.NO | DURATION | TOPICS TO BE COVERED  | SUPPORTED MATERIALS |
|------|----------|---|---------------------|
| 1    | 1        | LINUX : THE OPERATING SYSTEM , History , Features , Distribution, Relation ship To UNIX   | T1:695-702          |
| 2    | 1        | Overview Of LINUX, Architecture,Installation, Start up Scripts, Syatem Process , Security                                       | T1:702-712,712-735  |
| 3    | 1        | TUTORIAL-I  |                     |
| 4    | 1        | The EXT 2, EXT3 File System : General Characheritics , File Permission  | T1:735-740          |
| 5    | 1        | TUTORIAL-II   |                     |
| 6    | 1        | User Management: Types of Users Power Of Root, Managing User Using Command Line & GUI Tools                                     | T1:740-755<br>W1    |
| 7    | 1        | TUTORIAL-III  |                     |
| 8    | 1        | Resource Management : File & Directory Management , System Calls For Files Process Manangement , Signals                        | R1:630-650<br>W1    |
| 9    | 1        | IPC: Pipes, FIFO System V IPC, Message Queues , System Calls For Process, Memory Management , Library & System Calls For Memory | R1:605-630,650-661  |
| 10   |          | TUTORIAL-IV   |                     |
| 11   | 1        | RECAPITULATION OF IMPORTANT QUESTIONS   |                     |
| 12   | 1        | DISCUSSION OF ESE QUESTION PAPER  |                     |
| 13   | 1        | DISCUSSION OF ESE QUESTION PAPER  |                     |
| 14   | 1        | DISCUSSION OF ESE QUESTION PAPER  |                     |
|      |          | TOTAL HOURS:14 HOURS  |                     |

**TEXTBOOK :**

**T1:** Silberschatz Galvin Gagne. (2012). Operating system concepts, Ninth Edition, Wiley India (pvt), Ltd, New Delhi.

**REFERENCE BOOKS:**

**R1:** Deitel H.M. (2005). Operating systems, Third Edition, Addison Wesley Publication, New Delhi.

**WEBSITE:**

**W1:** [www.cs.columbia.edu/~nieh/teaching/e6118\\_s00/](http://www.cs.columbia.edu/~nieh/teaching/e6118_s00/)

Introduction -Mainframe systems Desktop Systems – Multiprocessor systems – distributed systems – real time systems. Process: - Process concepts – Operation on process – cooperation process - Inter process Communication - Mutual Exclusion - Critical sections- primitives – Semaphores – Deadlock: System Model, Deadlock characterization, Deadlock prevention, avoidance, detection, recovery from deadlock.

## **OPERATING SYSTEMS**

### **INTRODUCTION**

An Operating System (OS) is an interface between computer user and computer hardware. An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

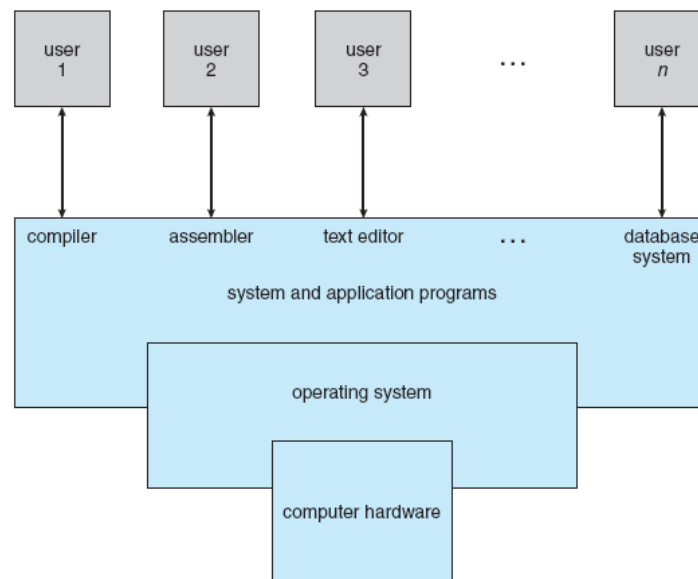
#### **Definition**

**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.**

#### **What is an OS?**

A computer system can be divided roughly into four components: the hardware, the operating system, the application programs, and the users. The hardware provides the basic computing resources for the system. The application programs define the ways in which these resources are used to solve users' computing problems.





### **System Components**

The operating system controls the hardware and coordinates its use among the various application programs for the various users. OS cannot be defined exactly because, it differs in perspective.

#### **➤ User View**

The user's view of the computer varies according to the interface being used. In a personal Computing environment the goal of OS is ease to use with some attention paid for resource-sharing. In Computing environment like mainframes and minicomputer Resource utilization is maximized for computer availability and prevent user from sharing other's fair time. In environment like client server Individual Usability and Resource sharing are compromised in designing. In latest technologies like mobile and touch-pads, lap-tops the work of OS is to improve battery-life for better efficiency. In some systems like embedded system user's interaction is needed in initial phases only. The design principles of user view differ, so defining the work of OS cannot be made on their perspective.

#### **➤ System View**

In system (Computer) point of view, the work of OS is involved with the efficiency of handling hardware or software resources. In context, an OS can be viewed as a Resource allocator. A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the manager of these resources. Facing numerous and possibly conflicting requests for resources, the operating system must decide how to allocate them to specific programs and users so that it can operate the computer system efficiently and fairly.

An operating system can be viewed as a -Control Program that manages the execution of user programs to prevent errors and improper use of the computer.

### **BASIC OS FUNCTION**

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 accessed directly by the CPU. For a program to be executed, it must be in the main memory. An 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, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

#### **Processor Management**

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

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

### **Device Management**

An Operating System manages device communication via their respective drivers. It 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.

An 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 an Operating System performs –

- Security – By means of password and similar other techniques, it prevents 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 softwares and users – Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

The operating system is the core software component of your computer. It performs many functions and is, in very basic terms, an interface between your computer and the outside world. In the section about hardware, a computer is described as consisting of several component parts including your monitor,

keyboard, mouse, and other parts. The operating system provides an interface to these parts using what is referred to as "drivers". This is why sometimes when you install a new printer or other piece of hardware, your system will ask you to install more software called a driver.

An operating system has three main functions: (1) manage the computer's resources, such as the central processing unit, memory, disk drives, and printers, (2) establish a user interface, and (3) execute and provide services for applications software.

### Other Operating System Functions

The operating system provides for several other functions including:

- System tools (programs) used to monitor computer performance, debug problems, or maintain parts of the system.
- A set of libraries or functions which programs may use to perform specific tasks especially relating to interfacing with computer system components.
- The operating system makes these interfacing functions along with its other functions operate smoothly and these functions are mostly transparent to the user.
- The operating system underpins the entire operation of the modern computer.

### **RESOURCE ABSTRACTION**

- Resource abstraction is the process of "hiding the details of how the hardware operates, thereby making computer hardware relatively easy for an application programmer to use"

### **OPERATING SYSTEM TYPES**

- There are many types of operating systems. The most common is the Microsoft suite of operating systems. They include from most recent to the oldest:
- Windows XP Professional Edition - A version used by many businesses on workstations. It has the ability to become a member of a corporate domain.
- Windows XP Home Edition - A lower cost version of Windows XP which is for home use only and should not be used at a business.

- Windows 2000 - A better version of the Windows NT operating system which works well both at home and as a workstation at a business. It includes technologies which allow hardware to be automatically detected and other enhancements over Windows NT.
- Windows ME - A upgraded version from windows 98 but it has been historically plagued with programming errors which may be frustrating for home users.
- Windows 98 - This was produced in two main versions. The first Windows 98 version was plagued with programming errors but the Windows 98 Second Edition which came out later was much better with many errors resolved.
- Windows NT - A version of Windows made specifically for businesses offering better control over workstation capabilities to help network administrators.
- Windows 95 - The first version of Windows after the older Windows 3.x versions offering a better interface and better library functions for programs.

There are other worthwhile types of operating systems not made by Microsoft. The greatest problem with these operating systems lies in the fact that not as many application programs are written for them. However if you can get the type of application programs you are looking for, one of the systems listed below may be a good choice.

- Unix - A system that has been around for many years and it is very stable. It is primary used to be a server rather than a workstation and should not be used by anyone who does not understand the system. It can be difficult to learn. Unix must normally run on a computer made by the same company that produces the software.
- Linux - Linux is similar to Unix in operation but it is free. It also should not be used by anyone who does not understand the system and can be difficult to learn.
- Apple Macintosh - Most recent versions are based on Unix but it has a good graphical interface so it is both stable (does not crash often or have as many software problems as other systems may have) and easy to learn. One drawback to this system is that it can only be run on Apple produced hardware.

### **TYPES OF OPERATING SYSTEM**

- Types of operating system which are commonly used

#### **MULTI-PROGRAMMING SYSTEM**

- The work of the server is to execute the job in sequence assigned by the users at their fair intervals. This is the first time the OS are programmed (Control Program or Handler) to handle the

users with the required resources. The switching between the users and the allocation of same resources to multiple processes was the difficult task. There was plenty of algorithm design for this by various research sectors in this time which paved a new way for multi-processing.

- *Multiprogramming* is a rudimentary form of parallel processing in which several programs are run at the same time on a uniprocessor. Since there is only one processor, there can be no true simultaneous execution of different programs.

### **BATCH OPERATING SYSTEM**

- The tasks are grouped as batch based on the priority specified by the user. Once the tasks are grouped they are executed as a batch by the machine. The duration of execution may be a week or even months. The tasks are grouped manually by a person and after proper execution the results are given to them by that person. The processing of OS is to just execute the task and not on scheduling.
- The users of a 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. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.

The problems with Batch Systems are as follows –

- Lack of interaction between the user and the job.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the 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, the objective is to maximize

processor use, whereas in Time-Sharing Systems, the 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, the processor executes each user program in a short burst or quantum of computation.
- That is, if  $n$  users are present, then each user can get a time quantum. When the user submits the command, the response time is in few seconds at most. The 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 as follows –

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

Disadvantages of Time-sharing operating systems are as follows –

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

### **REALTIMEOPERATINGSYSTEM**

- A real-time system is defined 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. The time taken by the system to respond to an input and display of required updated information is termed as the response time. So in this method, the response time is very less as compared to 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. A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail. For example, Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, 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 and the data is stored in ROM. In these systems, virtual memory is almost never found.

### **Soft real-time systems**

- Soft real-time systems are less restrictive. A 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.

### **DISTRIBUTED OPERATING SYSTEM**

- Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.
- The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred to as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred to as sites, nodes, computers, and so on.

The advantages of distributed systems are as follows –

- With resource sharing facility, a 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.

### **Distributed Systems**

- A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide users with access to the various resources that the system maintains. Access to a



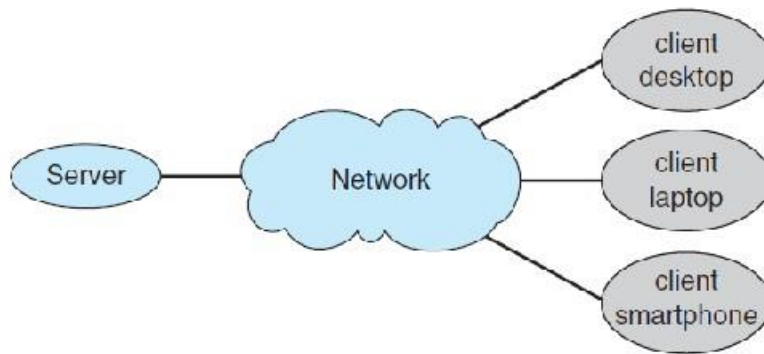
shared resource increases computation speed, functionality, data availability, and reliability. Some operating systems generalize network access as a form of file access, with the details of networking contained in the network interface's device driver. Distributed systems depend on networking for their functionality.

- Networks vary by the protocols used, the distances between nodes, and the transport media. TCP/IP is the most common network protocol, and it provides the fundamental architecture of the Internet. Most operating systems support TCP/IP, including all general-purpose ones. The media to carry networks are equally varied. They include copper wires, fiber strands, and wireless transmissions between satellites, microwave dishes, and radios.
- A network operating system is an operating system that provides features such as file sharing across the network, along with a communication scheme that allows different processes on different computers to exchange messages. A computer running a network operating system acts autonomously from all other computers on the network, although it is aware of the network and is able to communicate with other networked computers.

**Client-Server Computing:** As PCs have become faster, more powerful, and cheaper, designers have shifted away from centralized system architecture. Terminals connected to centralized systems are now being supplanted by PCs and mobile devices. Correspondingly, user-interface functionality once handled directly by centralized systems is increasingly being handled by PCs, quite often through a web interface. As a result, many of today's systems act as server systems to satisfy requests generated by client systems. This form of specialized distributed system, called a client-server system

Server systems can be broadly categorized as compute servers and file servers:

- The compute-server system provides an interface to which a client can send a request to perform an action (for example, read data). In response, the server executes the action and sends the results to the client. A server running a database that responds to client requests for data is an example of such a system.
- The file-server system provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a web server that delivers files to clients running web browsers.

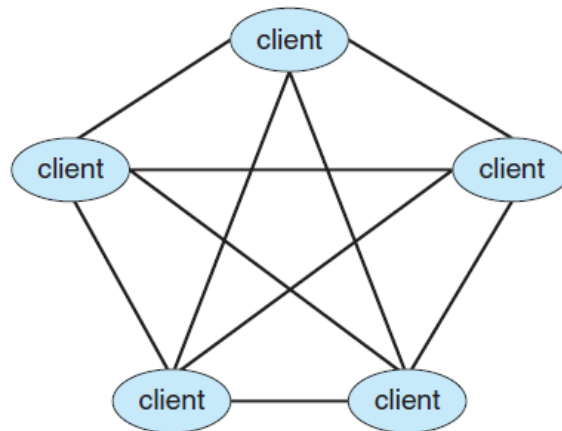


**Client-Server Model**

### **Peer to peer Systems**

Another structure for a distributed system is the peer-to-peer (P2P) system model. In this model, clients and servers are not distinguished from one another. Instead, all nodes within the system are considered peers, and each may act as either a client or a server, depending on whether it is requesting or providing a service. Peer-to-peer systems offer an advantage over traditional client-server systems. In a client-server system, the server is a bottleneck; but in a peer-to-peer system, services can be provided by several nodes distributed throughout the network. Determining what services are available is accomplished in one of two general ways:

- When a node joins a network, it registers its service with a centralized lookup service on the network. Any node desiring a specific service first contacts this centralized lookup service to determine which node provides the service. The remainder of the communication takes place between the client and the service provider.
- An alternative scheme uses no centralized lookup service. Instead, a peer acting as a client must discover what node provides a desired service by broadcasting a request for the service to all other nodes in the network. The node (or nodes) providing that service responds to the peer making the request. To support this approach, a discovery protocol must be provided that allows peers to discover services provided by other peers in the network.



#### **Peer-Peer with no-centralized Machine**

Skype is another example of peer-to-peer computing. It allows clients to make voice calls and video calls and to send text messages over the Internet using a technology known as voice over IP (VoIP). Skype uses a hybrid peer-to-peer approach. It includes a centralized login server, but it also incorporates decentralized peers and allows two peers to communicate.

#### **Network operating System**

- A Network Operating System runs on a server and provides the 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 include Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are as follows –

- 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 as follows –

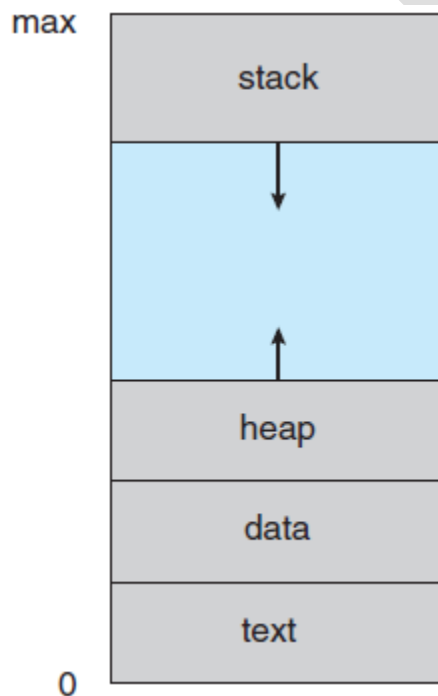
- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

**Process Concept**

Process is a program that is in execution. It is defined as unit of work in modern systems. A batch system executes jobs, whereas a time-shared system has user programs, or tasks. Even on a single-user system, a user may be able to run several programs at one time: a word processor, a Web browser, and an e-mail package. And even if a user can execute only one program at a time, such as on an embedded device that does not support multitasking, the operating system may need to support its own internal programmed activities, such as memory management. In many respects, all these activities are similar, so we call all of them processes.

**Process in memory**

A process is more than the program code, which is sometimes known as the text section. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. A process generally also includes the process stack, which contains temporary data (such as function parameters, return addresses, and local variables), and a data section, which contains global variables. A process may also include a heap, which is memory that is dynamically allocated during process run time.

**Process in Memory**

A program is a passive entity, such as a file containing a list of instructions stored on disk (often called an executable file). In contrast, a process is an active entity, with a program counter specifying the next instruction to

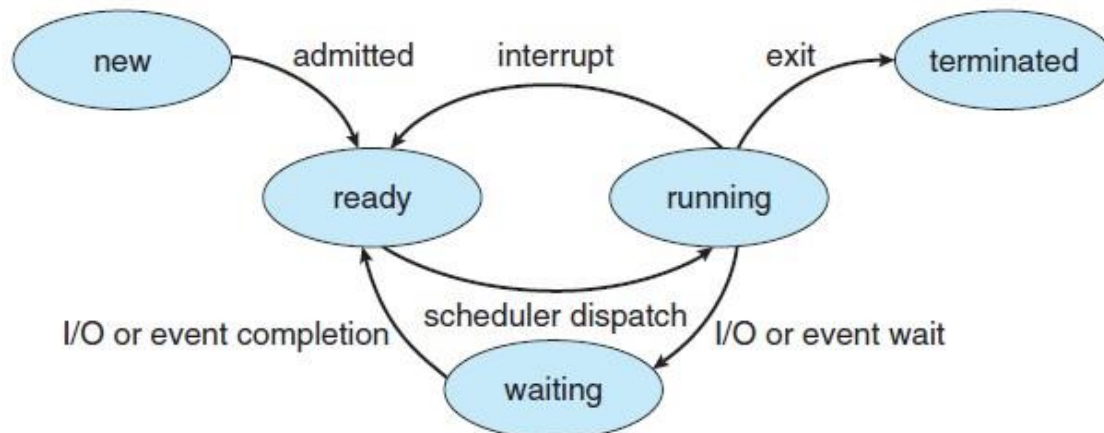
execute and a set of associated resources. A program becomes a process when an executable file is loaded into memory.

### Process State

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. A process may be in one of the following states:

- New. The process is being created.
- Running. Instructions are being executed.
- Waiting. The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- Ready. The process is waiting to be assigned to a processor.
- Terminated. The process has finished execution.

These names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems, however. Certain operating systems also more finely delineate process states. It is important to realize that only one process can be running on any processor at any instant. Many processes may be ready and waiting, however. The state diagram corresponding to these states is presented in the following Figure.



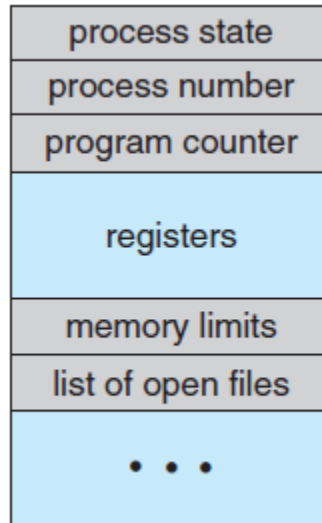
**Process State Diagram**

### Process Control Block (PCB)

Each process is represented in the operating system by a process control block (PCB)—also called a task control block. It contains many pieces of information associated with a specific process, including these: Process state. The state may be new, ready, running, and waiting, halted, and so on.

- **Program counter.** The counter indicates the address of the next instruction to be executed for this process.
- **CPU registers.** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.

- **CPU-scheduling information.** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

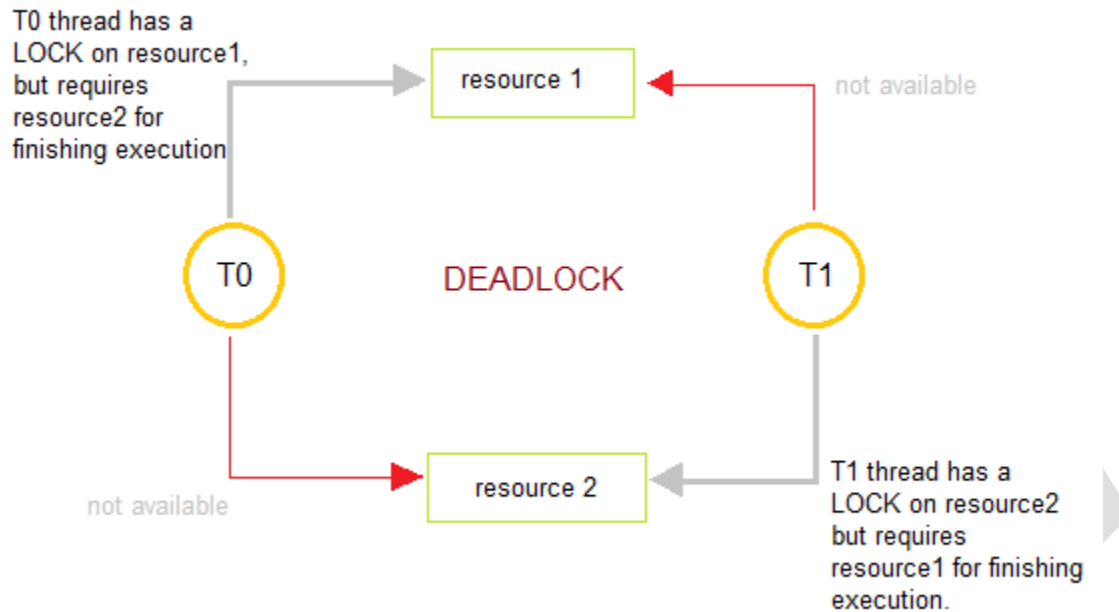


**Process Control Block (PCB)**

- **Memory-management information.** This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system
- **Accounting information.** This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- **I/O status information.** This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

### What is a Deadlock?

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.



### How to avoid Deadlocks

Deadlocks can be avoided by avoiding at least one of the four conditions, because all these four conditions are required simultaneously to cause a deadlock.

#### 1. Mutual Exclusion

Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, require exclusive access by a single process.

#### 2. Hold and Wait

In this condition, processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

#### 3. No Preemption

Preemption of process resource allocations can avoid the condition of deadlocks, wherever possible.

#### 4. Circular Wait

Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing (or decreasing) order.

### *Handling Deadlock*

The above points focus on preventing deadlocks. But what to do once a deadlock has occurred. Following three strategies can be used to remove deadlock after its occurrence.

#### **1. Preemption**

We can take a resource from one process and give it to other. This will resolve the deadlock situation, but sometimes it does causes problems.

#### **2. Rollback**

In situations where deadlock is a real possibility, the system can periodically make a record of the state of each process and when deadlock occurs, roll everything back to the last checkpoint, and restart, but allocating resources differently so that deadlock does not occur.

#### **3. Kill one or more processes**

This is the simplest way, but it works.

### *Deadlock Prevention*

Deadlock prevention algorithms ensure that at least one of the necessary conditions (Mutual exclusion, hold and wait, no preemption and circular wait) does not hold true. However most prevention algorithms have poor resource utilization, and hence result in reduced throughputs.

#### **Mutual Exclusion**

Not always possible to prevent deadlock by preventing mutual exclusion (making all resources shareable) as certain resources are cannot be shared safely.

#### **Hold and Wait**

We will see two approaches, but both have their disadvantages.

A resource can get all required resources before it start execution. This will avoid deadlock, but will result in reduced throughputs as resources are held by processes even when they are not needed. They could have been used by other processes during this time.

Second approach is to request for a resource only when it is not holding any other resource. This may result in a starvation as all required resources might not be available freely always.

#### **No preemption**



We will see two approaches here. If a process request for a resource which is held by another waiting resource, then the resource may be preempted from the other waiting resource. In the second approach, if a process request for a resource which are not readily available, all other resources that it holds are preempted.

The challenge here is that the resources can be preempted only if we can save the current state can be saved and processes could be restarted later from the saved state.

### **Circular wait**

To avoid circular wait, resources may be ordered and we can ensure that each process can request resources only in an increasing order of these numbers. The algorithm may itself increase complexity and may also lead to poor resource utilization.

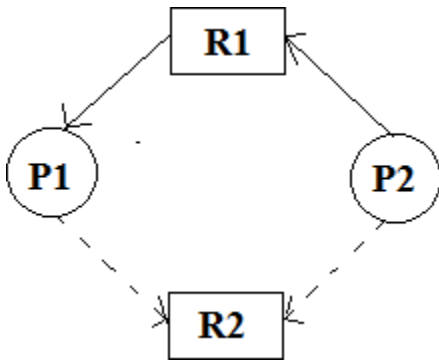
### **Deadlock avoidance**

As you saw already, most prevention algorithms have poor resource utilization, and hence result in reduced throughputs. Instead, we can try to avoid deadlocks by making use prior knowledge about the usage of resources by processes including resources available, resources allocated, future requests and future releases by processes. Most deadlock avoidance algorithms need every process to tell in advance the maximum number of resources of each type that it may need. Based on all these info we may decide if a process should wait for a resource or not, and thus avoid chances for circular wait.

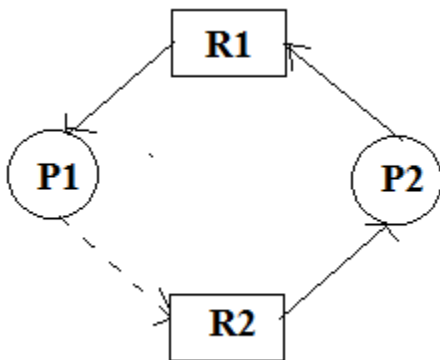
If a system is already in a safe state, we can try to stay away from an unsafe state and avoid deadlock. Deadlocks cannot be avoided in an unsafe state. A system can be considered to be in safe state if it is not in a state of deadlock and can allocate resources upto the maximum available. A safe sequence of processes and allocation of resources ensures a safe state. Deadlock avoidance algorithms try not to allocate resources to a process if it will make the system in an unsafe state. Since resource allocation is not done right away in some cases, deadlock avoidance algorithms also suffer from low resource utilization problem.

A resource allocation graph is generally used to avoid deadlocks. If there are no cycles in the resource allocation graph, then there are no deadlocks. If there are cycles, there may be a deadlock. If there is only one instance of every resource, then a cycle implies a deadlock. Vertices of the resource allocation graph are resources and processes. The resource allocation graph has request edges and assignment edges. An edge from a process to resource is a request edge and an edge from a resource to process is an allocation edge. A calm edge denotes that a request may be made in future and is represented as a dashed line. Based on calm edges we can see if there is a chance for a cycle and then grant requests if the system will again be in a safe state.

Consider the image with calm edges as below:



If R2 is allocated to p2 and if P1 request for R2, there will be a deadlock.



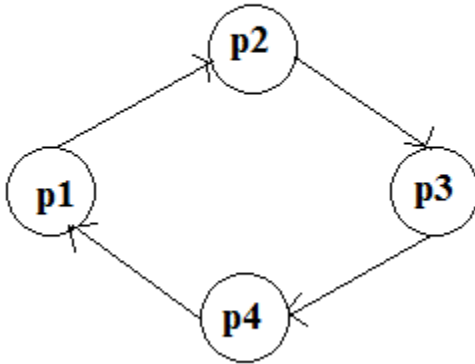
The resource allocation graph is not much useful if there are multiple instances for a resource. In such a case, we can use Banker's algorithm. In this algorithm, every process must tell upfront the maximum resource of each type it need, subject to the maximum available instances for each type. Allocation of resources is made only, if the allocation ensures a safe state; else the processes need to wait. The Banker's algorithm can be divided into two parts: Safety algorithm if a system is in a safe state or not. The resource request algorithm make an assumption of allocation and see if the system will be in a safe state. If the new state is unsafe, the resources are not allocated and the data structures are restored to their previous state; in this case the processes must wait for the resource. *You can refer to any operating system text books for details of these algorithms.*

### Deadlock Detection

If deadlock prevention and avoidance are not done properly, as deadlock may occur and only things left to do is to detect the recover from the deadlock.

If all resource types has only single instance, then we can use a graph called wait-for-graph, which is a variant of resource allocation graph. Here, vertices represent processes and a directed edge from P1 to P2 indicate that P1 is waiting for a resource held by P2. Like in the case of resource allocation graph, a

cycle in a wait-for-graph indicate a deadlock. So the system can maintain a wait-for-graph and check for cycles periodically to detect any deadlocks.



The wait-for-graph is not much useful if there are multiple instances for a resource, as a cycle may not imply a deadlock. In such a case, we can use an algorithm similar to Banker's algorithm to detect deadlock. We can see if further allocations can be made or not based on current allocations. *You can refer to any operating system text books for details of these algorithms.*

### Deadlock Recovery

Once a deadlock is detected, you will have to break the deadlock. It can be done through different ways, including, aborting one or more processes to break the circular wait condition causing the deadlock and preempting resources from one or more processes which are deadlocked.

**POSSIBLE QUESTIONS**

**PART –B**

**(Each Question carries 2 Marks)**

1. What is Process?
2. What is Process Control?
3. List the Basic OS Functions?
4. What is an Operating System?
5. What is critical section?
6. What is semaphore?
7. What are all basic conditions for deadlock?
8. Define: Software
9. Define: Hardware
10. What is IPC?

**PART –C**

**(Each Question carries 6 Marks)**

1. Write banker's algorithm and explain.
2. Write a short notes on inter process communication.
3. Explain the Types of Operating System.
4. Explain Basic OS Functions
5. Explain in detail about Multiprogramming Systems
6. Discuss about Process Control Block in detail
7. Explain in detail about Distributed System
8. Discuss about the overview of Operating System in detail
9. Discuss about (i) Batch System (ii) Real time System
10. Explain in detail about the Resource Abstraction.

## Semester : III

## Subject : Operating Systems: Linux

| S.NO | Question   | Opt1                               |
|------|--|------------------------------------|
| 1    | The queue has maximum length 0; thus, the link ca  | Zero capacity                      |
| 2    | A page fault occurs  | when the page is not in the memory |
| 3    | Let S and Q be two semaphores initialized to 1, where P <sub>0</sub> and P <sub>1</sub> processes the following statements wait(S);wait(Q);--;signal(S) and signal(Q) and wait(Q);wait(S);_---;signal(Q);signal(S);respectively. The above situation depicts a | semaphore                          |
| 4    | computer hardware and acts as an intermediary between the computer user and the computer hardware.   | hardware acceleration              |
| 5    | _____manages the execution of user programs t  | resource allocator                 |
| 6    | system is the one program running at all times on the computer usually called_____   | bootstrap                          |
| 7    | computer system can be divided into  | 2                                  |
| 8    | _____were the first computers used to tackle ma  | Mainframe computer system          |

Academy of Higher Education

Department of Mathematics

Academic Year : 2017-2020

Class : II B.Sc Maths

Subject Code: 17MMU404B

| UNIT -I                        |  |  |
|--------------------------------|--|--|
| Opt2                           | Opt3                                       | Opt4                                   |
| Bounded capacity               | Unbounded capacity                         | synchronous                            |
| when the page is in the memory | when the process entered the blocked state | when the process is in the ready state |
|                                |  |  |
| deadlock                       | signal                                     | interrupt                              |
|                                |  |  |
| Operating System               | compiler                                   | logical transaction                    |
| work station                   | main frame                                 | control program                        |
|                                |  |  |
| firmware                       | kernel                                     | read-only memory                       |
| 3                              | 4  | 5                                      |
| Mainframe computer service     | multiframe computer system                 | multiframe computer service            |

|                                    |
|------------------------------------|
|                                    |
| Answer                             |
| Zero capacity                      |
| when the page is not in the memory |
|                                    |
| deadlock                           |
|                                    |
| Operating System                   |
| control program                    |
|                                    |
| kernel                             |
| 2                                  |
| Mainframe computer system          |

|  |  |                                  |
|--|--|----------------------------------|
| 9  | _____system is the collection of computer that act,work and appear as one large computer size of a distributed system. | distributed                      |
| 10                                       | _____contains the address of an instruction to be fetched from memory  | Program counter (                |
| <small>Academic Year : 2017-2018</small> |  |                                  |
|  | _____contains the instruction mo   | Program counter (                |
| 12                                       | The kernel is a _____  | memory manager                   |
| 13                                       | Main function of shared memory is _____  | to use primary memory effciently |
| 14                                       | Disk scheduling includes deciding _____  | which should be accessed next    |
| 15                                       | Memory protection is normally done by _____  | the processor and the associated |
| 16                                       | _____controls the nodes hardware   | ubiquitous minimal kernel        |
| 17                                       | In Banker's Algorithm _____co  | mutual-exclusion                 |
| 18                                       | _____is the process of actually determinin   | Deadlock detection               |
| 19                                       | Once a system has become _____the deadlock must be broken by removing one or more of the necessary conditions          | deadlocked                       |
| 20                                       | _____is also known as parallel system  | Multiprocessor systems           |
| 21                                       | _____operating systems are even more   | Time-sharing                     |
| 22                                       | _____ operating system keeps several jobs in memory simultaneously.  | Time-sharing                     |



|   |                                   |                                   |  |
|---|-----------------------------------|-----------------------------------|--|
| symmetric                                   | asymmetric                        | multiple                          | distributed  |
| Instruction register                        | Control registers                 | Status registers                  | Instruction register (IR)                            |
| Instruction register                        | Control registers                 | Status registers                  | Program counter (PC)                                 |
| resource manager                            | file manager                      | directory manager                 | resource manager                                     |
| <b>Subject Code:<br/>17MMU404B</b>          | to do inter process communication | to do other process communication | to do inter process communication                    |
| order in which disk access requests must be | the physical location of the file | the logical location of the file  | order in which disk access requests must be serviced |
| the operating system                        | the compiler                      | the user program                  | the processor and the associated hardware            |
| ubiquitous maximal kernel                   | ubiquitous normal kernel          | high level system management      | ubiquitous minimal kernel                            |
| Time-sharing                                | race condition                    | cooperating processes             | mutual-exclusion                                     |
| Deadlock prevention                         | fault tolerant                    | process synchronization           | Deadlock detection                                   |
| race condition                              | mutual-exclusion                  | cooperating processes             | deadlocked   |
| desktop systems                             | Time sharing systems              | Multiprogrammed systems           | Multiprocessor systems                               |
| desktop systems                             | Multiprogrammed systems           | Multiprocessor systems            | Time-sharing   |
| desktop systems                             | Multiprogrammed systems           | Multiprocessor systems            | Multiprogrammed systems                              |

|                           |  |   |
|---------------------------|--|---|
| 23                        | _____ can save more money than multiple s  | Multiprocessor systems                      |
| 24                        | This ability to continue providing service proportio   | fault tolerant.                             |
| Academic Year : 2017-2020 | Systems designed for graceful degradation are also   | graceful degradation                        |
| 26                        | The most common multiple-processor systems now   | symmetric multiprocessing                   |
| 27                        | Another form of a special-purpose operating system   | real-time system                            |
| 28                        | The assignment of the CPU to the first process on t  | graceful degradation                        |
| 29                        | The manifestation of a process in an operating syst  | Process state transitions                   |
| 30                        | A process may spawn a new process. If it does, the creating process is called the parent process and the created process is called the | child process                               |
| 31                        | The communication is direct or indirect, messages exchanged by communicating processes reside in a temporary queue known as            | Buffering                                   |
| 32                        | The message-passing facility in Windows 2000 is c  | MUTUAL EXCLUSION                            |
| 33                        | _____ can assume only the value 0 or the   | Binary semaphore                            |
| 34                        | Semaphores are used to solve the problem of  | race condition                              |
| 35                        | For multiprogramming operating system  | special support from processor is essential |
| 36                        | Which is single user operating system  | MS-DOS                                      |
| 37                        | Which operating system reacts in the actual time   | Batch system                                |
| 38                        | In real time OS, which is most suitable scheduling scheme  | round robin                                 |
| 39                        | Dispatcher function is to  | put tasks in I/O wait                       |

|    |  |   |
|----|--|---|
| 40 | Multiprogramming systems   | Are easier to develop than single programming |
| 41 | Operating system is  | A collection of hardware components           |
| 42 | Semaphores function is to  | synchronize critical resources to prevent     |
| 43 | Which operating system use write through caches  | UNIX  |
| 44 | Which process is known for initializing a microcomputer with its OS                          | cold booting                                  |
| 45 | Four necessary conditions for deadlock are non pre-emption, circular wait, hold and wait and | mutual exclusion                              |
| 46 | Remote computing services involves the use of timesharing and                                | multiprocessing                               |
| 47 | A series of statements explaining how the data is to be processed is called                  | instruction                                   |
| 48 | Banker's algorithm deals with  | deadlock prevention                           |
| 49 | Which is non pre-emptive   | Round robin                                   |
| 50 | A hardware device which is capable of executing a sequence of instructions, is known as      | CPU   |
| 51 | Distributed systems should   | high security                                 |
| 52 | Which of the following is always there in a computer   | Batch system                                  |
| 53 | Which of following is not an advantage of multiprogramming                                   | increased throughput                          |
| 54 | In which of the following usually a front end processor is used                              | Virtual storage                               |
| 55 | Remote computing services involves the use of  | multiprocessing                               |
| 56 | Banker's algorithm for resource allocation deals with  | deadlock prevention                           |

|    |  |                  |
|----|--|------------------|
| 57 | When did IBM released the first version of its disk operating system DOS version 1.0 | 1981             |
| 58 | The queue has finite length $n$ ; thus, at most $n$ mess                             | Zero capacity    |
| 59 | The queue has potentially infinite length; thus, any                                 | Zero capacity    |
| 60 | all others from doing so simultaneously and this is called                           | mutual exclusion |
|    |  |                  |

|   |                                |                               |   |
|---|--------------------------------|-------------------------------|---|
| desktop systems                                 | Time sharing systems           | Multiprogrammed systems       | Multiprocessor systems                          |
| graceful degradation                            | Economy of scale               | Increased throughput          | graceful degradation                            |
| Economy of scale                                | fault tolerant                 | Increased throughput          | fault tolerant                                  |
| asymmetric multiprocessing                      | multithreading                 | multiprogramming              | symmetric multiprocessing                       |
| distributed operating system                    | <b>Subject Code: 17MMU404B</b> | multiframe computer system    | real-time system                                |
| Time-sharing                                    | dispatching                    | Multiprocessor systems        | dispatching                                     |
| process control block                           | child process                  | cooperating processes         | process control block                           |
| Process state transitions                       | Process state transitions      | process control block         | child process                                   |
| synchronization                                 | asynchronization               | communication link            | Buffering                                       |
| Buffering                                       | local procedure call facility  | CRITICAL SECTIONS             | local procedure call facility                   |
| Counting semaphores                             | semaphore operations           | normal semaphores             | Binary semaphores                               |
| process synchronization                         | mutual exclusion               | belady problem                | mutual exclusion                                |
| special support from processor is not essential | cache memory is essential      | cache memory is not essential | special support from processor is not essential |
| UNIX  | XENIX                          | LINUX                         | MS-DOS  |
| Quick response system                           | Real time system               | Time sharing system           | Real time system                                |
| FCFS  | pre-emptive scheduling         | random scheduling             | pre-emptive scheduling                          |
| schedule tasks in processor                     | change task priorities         | Multitasking                  | put tasks in I/O wait                           |

|  |                                      |   |  |
|--|--------------------------------------|---|--|
| Execute each job faster                          | Execute more jobs in the same time   | Are used only on large main frame computers | Execute more jobs in the same time                 |
| A collection of input output devices             | A collection of software routines    | last entered the queue                      | A collection of software routines                  |
| synchronize processes for better CPU utilization | used for memory management           | may cause a high I/O rate                   | synchronize critical resources to prevent deadlock |
| XENIX  | ULTRIX                               | DOS   | DOS  |
| boot recording                                   | booting                              | warm booting                                | booting  |
| race condition                                   | buffer overflow                      | multiprocessing                             | mutual exclusion                                   |
| interactive processing                           | batch processing                     | real time processing                        | batch processing                                   |
| compiler   | program                              | interpretor                                 | program  |
| deadlock avoidance                               | deadlock recovery                    | mutual exclusion                            | deadlock avoidance                                 |
| FIFO   | MQS                                  | MQSF  | FIFO   |
| ALU  | CU                                   | Processor                                   | Processor  |
| have better resource sharing                     | better system utilization            | low system overhead                         | have better resource sharing                       |
| Operating system                                 | Time sharing system                  | Controlling system                          | Operating system                                   |
| shorter response time                            | ability to assign priorities of jobs | decreased system overload                   | decreased system overload                          |
| Timesharing                                      | Multiprogramming                     | Multithreading                              | Timesharing  |
| multiprogramming                                 | batch processing                     | real time processing                        | batch processing                                   |
| deadlock aviodance                               | deadlock recovery                    | circular wait                               | deadlock aviodance                                 |

|                  |                      |                  |                    |
|------------------|----------------------|------------------|--------------------|
| 1982             | 1983                 | 1984             | 1981               |
| Bounded capacity | Unbounded capacity   | multiprocessing  | Bounded capacity   |
| Bounded capacity | Unbounded capacity   | multiprocessing  | Unbounded capacity |
| multiprocessing  | real time processing | multiprogramming | mutual exclusion   |
|                  |                      |                  |                    |

Storage management: Memory Management - swapping- Contiguous memory allocation – paging, segmentation – segmentation with paging – Virtual memory :Virtual storage organization – Demand Paging, Process Creation – Page replacement – Thrashing.

## Storage management

### Memory Management

Main Memory refers to a physical memory that is the internal memory to the computer. The word main is used to distinguish it from external mass storage devices such as disk drives. Main memory is also known as RAM. The computer is able to change only data that is in main memory. Therefore, every program we execute and every file we access must be copied from a storage device into main memory.

All the programs are loaded in the main memory for execution. Sometimes complete program is loaded into the memory, but some times a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called **Dynamic Loading**, this enhance the performance.

Also, at times one program is dependent on some other program. In such a case, rather than loading all the dependent programs, CPU links the dependent programs to the main executing program when its required. This mechanism is known as **Dynamic Linking**.

### Swapping

A process needs to be in memory for execution. But sometimes there is not enough main memory to hold all the currently active processes in a timesharing system. So, excess process are kept on disk and brought in to run dynamically. Swapping is the process of bringing in each process in main memory, running it for a while and then putting it back to the disk.

### Contiguous Memory Allocation

In contiguous memory allocation each process is contained in a single contiguous block of memory. Memory is divided into several fixed size partitions. Each partition contains exactly one process. When a partition is free, a process is selected from the input queue and loaded into it. The free blocks of memory are known as *holes*. The set of holes is searched to determine which hole is best to allocate.



### Memory Protection

Memory protection is a phenomenon by which we control memory access rights on a computer. The main aim of it is to prevent a process from accessing memory that has not been allocated to it. Hence prevents a bug within a process from affecting other processes, or the operating system itself, and instead results in a segmentation fault or storage violation exception being sent to the disturbing process, generally killing of process.

### Memory Allocation

Memory allocation is a process by which computer programs are assigned memory or space. It is of three types :

1. **First Fit**

The first hole that is big enough is allocated to program.

2. **Best Fit**

The smallest hole that is big enough is allocated to program.

3. **Worst Fit**

The largest hole that is big enough is allocated to program.

### Fragmentation

Fragmentation occurs in a dynamic memory allocation system when most of the free blocks are too small to satisfy any request. It is generally termed as inability to use the available memory.

In such situation processes are loaded and removed from the memory. As a result of this, free holes exists to satisfy a request but is non contiguous i.e. the memory is fragmented into large no. Of small holes. This phenomenon is known as **External Fragmentation**.

Also, at times the physical memory is broken into fixed size blocks and memory is allocated in unit of block sizes. The memory allocated to a space may be slightly larger than the requested memory. The difference between allocated and required memory is known as **Internal fragmentation** i.e. the memory that is internal to a partition but is of no use.

### Paging

A solution to fragmentation problem is Paging. Paging is a memory management mechanism that allows the physical address space of a process to be non-contiguous. Here physical memory is divided into blocks of equal size called **Pages**. The pages belonging to a certain process are loaded into available memory frames.

---

### Page Table

A Page Table is the data structure used by a virtual memory system in a computer operating system to store the mapping between *virtual address* and *physical addresses*.

Virtual address is also known as Logical address and is generated by the CPU. While Physical address is the address that actually exists on memory.

---

### Segmentation

Segmentation is another memory management scheme that supports the user-view of memory. Segmentation allows breaking of the virtual address space of a single process into segments that may be placed in non-contiguous areas of physical memory.

---

### Segmentation with Paging

Both paging and segmentation have their advantages and disadvantages, it is better to combine these two schemes to improve on each. The combined scheme is known as 'Page the Elements'. Each segment in this scheme is divided into pages and each segment is maintained in a page table. So the logical address is divided into following 3 parts :

- Segment numbers(S)
- Page number (P)
- The displacement or offset number (D)

### Virtual Memory

Virtual Memory is a space where large programs can store themselves in form of pages while their execution and only the required pages or portions of processes are loaded into the main memory. This technique is useful as large virtual memory is provided for user programs when a very small physical memory is there.

In real scenarios, most processes never need all their pages at once, for following reasons :

- Error handling code is not needed unless that specific error occurs, some of which are quite rare.
- Arrays are often over-sized for worst-case scenarios, and only a small fraction of the arrays are actually used in practice.
- Certain features of certain programs are rarely used.

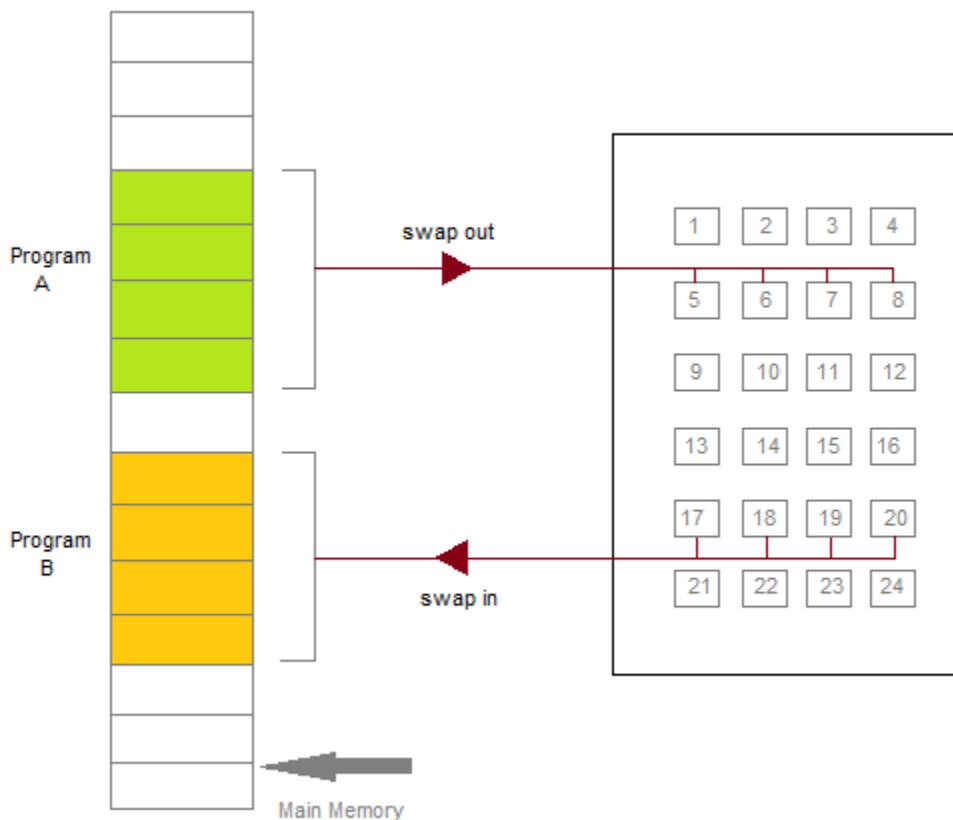
### Benefits of having Virtual Memory :

1. Large programs can be written, as virtual space available is huge compared to physical memory.
2. Less I/O required, leads to faster and easy swapping of processes.

3. More physical memory available, as programs are stored on virtual memory, so they occupy very less space on actual physical memory.

### **Demand Paging**

The basic idea behind demand paging is that when a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them (On demand). This is termed as lazy swapper, although a pager is a more accurate term.



Initially only those pages are loaded which will be required the process immediately.

The pages that are not moved into the memory, are marked as invalid in the page table. For an invalid entry the rest of the table is empty. In case of pages that are loaded in the memory, they are marked as valid along with the information about where to find the swapped out page.

When the process requires any of the page that is not loaded into the memory, a page fault trap is triggered and following steps are followed,

1. The memory address which is requested by the process is first checked, to verify the request made by the process.
2. If its found to be invalid, the process is terminated.

3. In case the request by the process is valid, a free frame is located, possibly from a free-frame list, where the required page will be moved.
4. A new operation is scheduled to move the necessary page from disk to the specified memory location. ( This will usually block the process on an I/O wait, allowing some other process to use the CPU in the meantime. )
5. When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to valid.
6. The instruction that caused the page fault must now be restarted from the beginning.

There are cases when no pages are loaded into the memory initially, pages are only loaded when demanded by the process by generating page faults. This is called **Pure Demand Paging**.

The only major issue with Demand Paging is, after a new page is loaded, the process starts execution from the beginning. Its is not a big issue for small programs, but for larger programs it affects performance drastically.

---

### Page Replacement

As studied in Demand Paging, only certain pages of a process are loaded initially into the memory. This allows us to get more number of processes into the memory at the same time. but what happens when a process requests for more pages and no free memory is available to bring them in. Following steps can be taken to deal with this problem :

1. Put the process in the wait queue, until any other process finishes its execution thereby freeing frames.
2. Or, remove some other process completely from the memory to free frames.
3. Or, find some pages that are not being used right now, move them to the disk to get free frames. This technique is called **Page replacement** and is most commonly used. We have some great algorithms to carry on page replacement efficiently.

### Basic Page Replacement

- Find the location of the page requested by ongoing process on the disk.
- Find a free frame. If there is a free frame, use it. If there is no free frame, use a page-replacement algorithm to select any existing frame to be replaced, such frame is known as **victim frame**.
- Write the victim frame to disk. Change all related page tables to indicate that this page is no longer in memory.
- Move the required page and store it in the frame. Adjust all related page and frame tables to indicate the change.
- Restart the process that was waiting for this page.

### FIFO Page Replacement

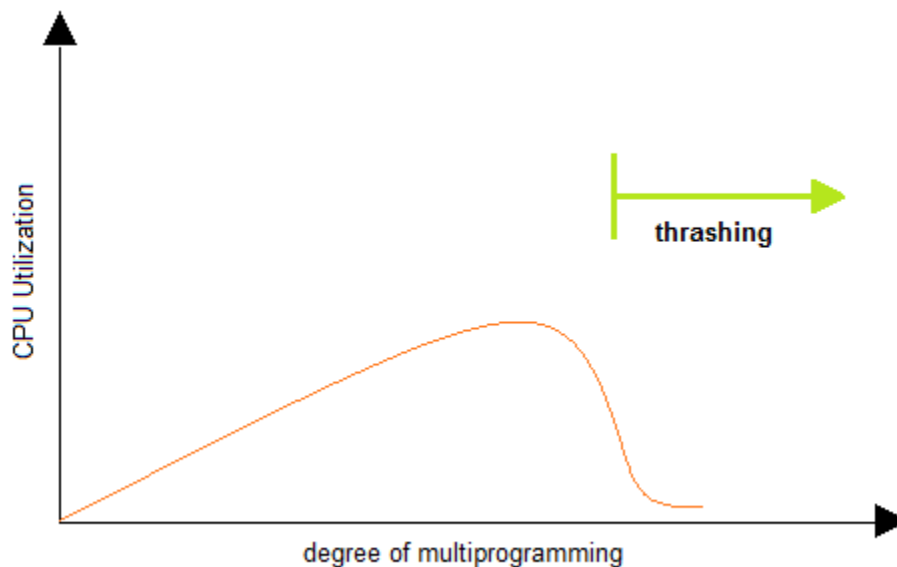
- A very simple way of Page replacement is FIFO (First in First Out)

- As new pages are requested and are swapped in, they are added to tail of a queue and the page which is at the head becomes the victim.
- Its not an effective way of page replacement but can be used for small systems.

### Thrashing

A process that is spending more time paging than executing is said to be thrashing. In other words it means, that the process doesn't have enough frames to hold all the pages for its execution, so it is swapping pages in and out very frequently to keep executing. Sometimes, the pages which will be required in the near future have to be swapped out.

Initially when the CPU utilization is low, the process scheduling mechanism, to increase the level of multiprogramming loads multiple processes into the memory at the same time, allocating a limited amount of frames to each process. As the memory fills up, process starts to spend a lot of time for the required pages to be swapped in, again leading to low CPU utilization because most of the processes are waiting for pages. Hence the scheduler loads more processes to increase CPU utilization, as this continues at a point of time the complete system comes to a stop.



To prevent thrashing we must provide processes with as many frames as they really need "right now".

**POSSIBLE QUESTIONS**

**PART –B**

**(Each Question carries 2 Marks)**

- 1.What is memory?
- 2.Define: Swapping
- 3.What is paging?
- 4.Define:Segmentation
- 5.Define:word in memory
- 6.What is thrashing?
- 7.Define: Semaphore
- 8.How can you achieve page replacement?
- 9.What is virtual storage organization?
10. Define: Virtual memory

**PART –C**

**(Each Question carries 6 Marks)**

1. Explain about Memory Allocation Strategies.
2. Explain about Fixed and Variable partition.
3. Explain the Comparison between paging and Fragmentation
4. Difference between Physical address space and Virtual Address space
5. Discuss about swapping of two processes
6. Difference between Paging and Segmentation
7. Explain the concept of Physical address space in detail.
8. Explain in detail about Segmentation.
9. Explain about Virtual address space.
10. Discuss about Paging in detail

Semester : III

Subject : Operating Systems: Linux

| S.NO | Question   | Opt1                |
|------|--|---------------------|
| UNI  |  |                     |
| 1    | Memory is array of _____                           | bytes               |
| 2    | CPU fetches instructions from _____                | memory              |
| 3    | Program must be in _____                           | memory              |
| 4    | Collection of process in disk forms _____          | input queue         |
| 5    | Address space of computer starts at _____          | 0000                |
| 6    | If process location is found during compile time   | absolute            |
| 7    | Address generated by CPU is _____                  | logical             |
| 8    | Logical address can be also called as _____        | physical            |
| 9    | Run time mapping is done using                     | MMU                 |
| 10   | In address binding base register is also called as | relocation register |
| 11   | Better memory space is utilized using              | dynamic loading     |
| 12   | _____ routine is never loaded in                   | unused              |
| 13   | Some operating systems support only                | static              |
| 14   | _____ is a code that locates library r             | stub                |
| 15   | _____ can be used to manage large me               | overlays            |
| 16   | _____ error is raised in memory                    | addressing          |
| 17   | Set of _____ are scattered throughout th           | holes               |
| 18   | _____ can be internal and external                 | fragmentation       |
| 19   | _____ is used to divide a process into f           | paging              |
| 20   | In paging physical memory is divided into _____    | frames              |
| 21   | In paging virtual memory is divided into _____     | frames              |
| 22   | _____ is first of virtual address in pag           | page number         |
| 23   | _____ is second part of virtual addres             | page number         |
| 24   | Page mapping entries are found in _____            | page table          |
| 25   | Page size is defined by _____                      | hardware            |
| 26   | _____ is first in mapping of virtual to            | direct              |
| 27   | _____ is second in mapping of virtual              | direct              |
| 28   | _____ is third in mapping of virtual to            | direct              |
| 29   | _____ is used to divide a process into v           | paging              |
| 30   | In segmentation virtual memory is divided into     | frames              |

|    |  |               |
|----|--|---------------|
| 31 | _____ view is supported in segmentatio         | user          |
| 32 | _____ is format for segmentation virtu         | (s,d)         |
| 33 | _____ is the first element in segment ta       | limit         |
| 34 | _____ is the second element in segmen          | limit         |
| 35 | Addressing in segmentation is similar as _____ | direct        |
| 36 | How many elements are there in segmentation a  | 1             |
| 37 | _____ is organization in physical mem          | frames        |
| 38 | _____ memory is used to manage i               | virtual       |
| 39 | virtual memory abstracts _____ memory          | virtual       |
| 40 | _____ reasons are there for existence          | 1             |
| 41 | _____ benefits are there from virtual          | 1             |
| 42 | Virtual memory is commonly implemented by _    | demand        |
| 43 | _____ fault occurs when desired pa             | page          |
| 44 | _____ table is used in demand pagin            | page          |
| 45 | _____ methods are there for process            | 1             |
| 46 | _____ method implements partial sh             | copy on write |
| 47 | _____ is done for page fault                   | replacement   |
| 48 | _____ is unrealizable page replacem            | optimal       |
| 49 | _____ is first page replacement algo           | optimal       |
| 50 | _____ is second page replacement al            | optimal       |
| 51 | _____ is third page replacement algo           | optimal       |
| 52 | _____ is associated with each page i           | label         |
| 53 | _____ labelled page replaced in opti           | highest       |
| 54 | _____ end page is removed in fifo al           | rear          |
| 55 | Modified version of fifo algorithm gives _____ | 1             |
| 56 | _____ is called as high paging activity        | thrashing     |
| 57 | _____ occurs frequently during thrash          | page fault    |
| 58 | _____ strategy is used to solve thras          | working set   |
| 59 | _____ algorithm is used to solve thra          | working set   |
| 60 | _____ is a basic solution for thrashin         | working set   |
|    |  |               |



**7 of Higher Education**  
**nent of Mathematics**

Class : II B.Sc Maths

Subject Code:

17MMU404B

| Opt2            | Opt3                | Opt4           | Answer               |
|-----------------|---------------------|----------------|----------------------|
| <b>T -II</b>    |                     |                |                      |
| circuits        | ics                 | ram            | bytes                |
| pendrive        | dvd                 | cmos           | memory               |
| pendrive        | dvd                 | cmos           | memory               |
| output queue    | stack               | circle         | input queue          |
| 4444            | 3333                | 2222           | 0000                 |
| relative        | approximate         | more or less   | absolute             |
| physical        | direct              | indirect       | logical              |
| virtual         | direct              | indirect       | virtual              |
| CPU             | CU                  | IU             | MMU                  |
| memory register | hard disk           | pendrive       | relocation register  |
| dynamic linking | registers           | array of words | dynamic loading      |
| used            | regular             | recursive      | unused               |
| dynamic         | temporary           | interruptive   | static               |
| dll             | recursive routine   | exe file       | stub                 |
| swapping        | roll in and out     | libraries      | overlays             |
| swapping        | dynamic             | index          | addressing           |
| gaps            | free space          | words          | holes                |
| merging         | grouping            | fixing         | fragmentation        |
| segmentation    | sp                  | swapping       | paging               |
| pages           | segments            | bytes          | frames               |
| pages           | segments            | bytes          | pages                |
| segment number  | frame number        | offset         | page number          |
| segment number  | frame number        | offset         | offset               |
| segment table   | hash table          | pointing table | page table           |
| software        | os                  | kernel         | hardware             |
| associate       | direct & associativ | pointing       | direct               |
| associate       | direct & associativ | pointing       | associate            |
| associate       | direct & associativ | pointing       | direct & associative |
| segmentation    | sp                  | swapping       | segmentation         |
| pages           | segments            | bytes          | segments             |

|                |                     |               |               |
|----------------|---------------------|---------------|---------------|
| system         | cpu                 | manager       | user          |
| (p,d)          | (v,d)               | (k,d)         | (s,d)         |
| base           | offset              | page number   | limit         |
| base           | offset              | page number   | base          |
| associate      | direct & associativ | pointing      | direct        |
| 2              | 3                   | 4             | 3             |
| pages          | segments            | bytes         | frames        |
| physical       | rom                 | eprom         | virtual       |
| eerom          | main                | eprom         | main          |
| 2              | 3                   | 4             | 3             |
| 2              | 3                   | 4             | 3             |
| bargain        | quarrel             | order         | demand        |
| segment        | pages               | segments      | page          |
| segment        | pages               | segments      | page          |
| 2              | 3                   | 4             | 2             |
| memory mapping | paging              | segmentation  | copy on write |
| swapping       | logging             | locking       | replacement   |
| FIFO           | LRU                 | NRU           | optimal       |
| FIFO           | LRU                 | NRU           | FIFO          |
| FIFO           | LRU                 | NRU           | optimal       |
| FIFO           | LRU                 | NRU           | NRU           |
| index          | number              | identity      | label         |
| lowest         | moderate            | below average | highest       |
| head           | top                 | bottom        | head          |
| 2              | 3                   | 4             | 2             |
| smashing       | mocking             | breaking      | thrashing     |
| segment fault  | memory fault        | address fault | page fault    |
| pff            | lpr algorithm       | gpl algorithm | working set   |
| pff            | lpr algorithm       | gpl algorithm | lpr algorithm |
| pff            | lpr algorithm       | gpl algorithm | pff           |
|                |                     |               |               |

Processor Scheduling : preemptive scheduling : - Scheduling Criteria – Scheduling Algorithms – FCFS- SJF- Priority – RoundRobin –Multilevel Queue – Multilevel Feedback Queue . Multiprocess schedule: Real time schedule, Algorithm evaluation: Deterministic Modeling, Queue Model, Simulation

## Process Scheduling

The act of determining which process in the ready state should be moved to the running state is known as Process Scheduling.

The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs. For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of CPU.

Schedulers fall into one of the two general categories :

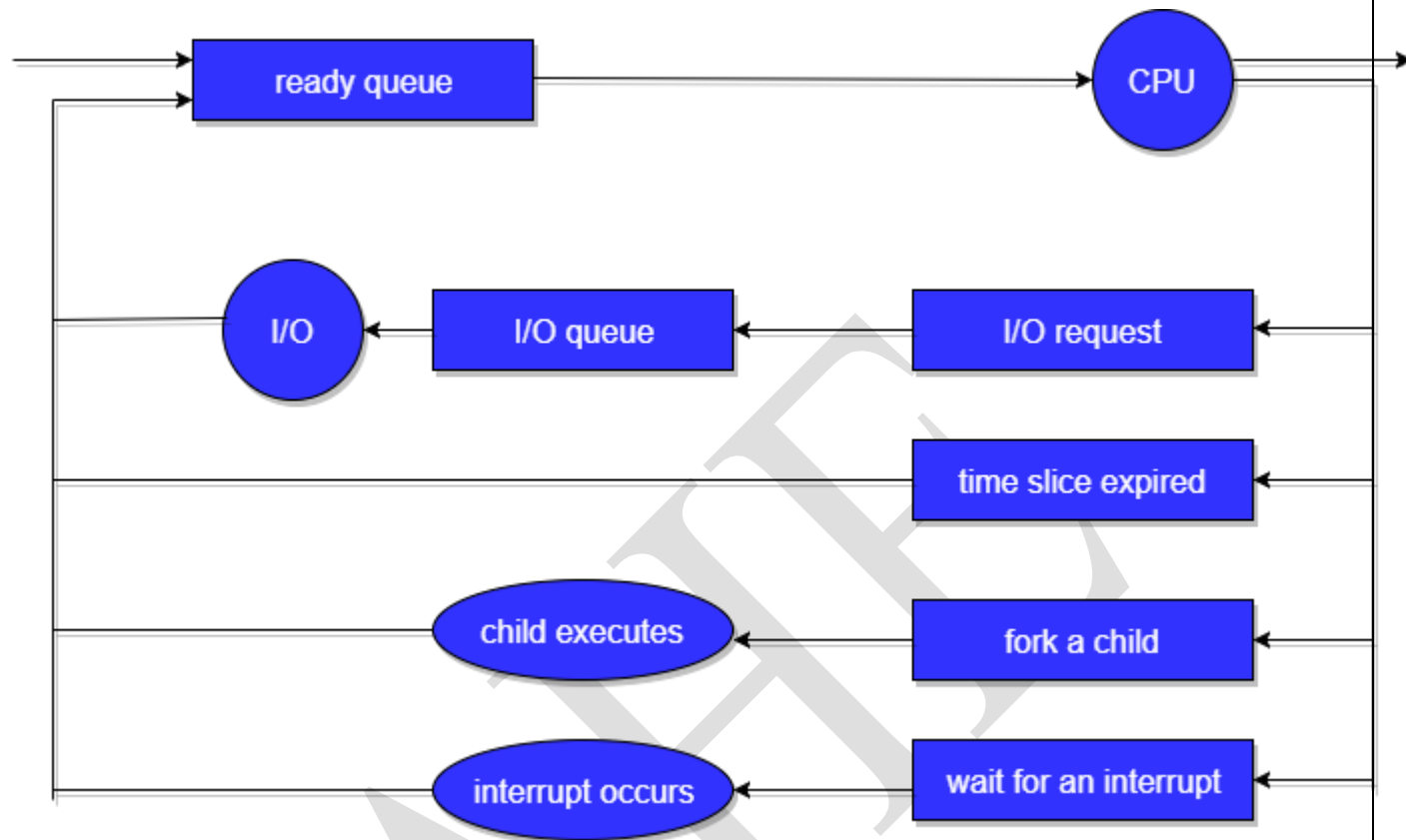
- **Non pre-emptive scheduling.** When the currently executing process gives up the CPU voluntarily.
- **Pre-emptive scheduling.** When the operating system decides to favour another process, pre-empting the currently executing process.

## Scheduling Queues

- All processes when enters into the system are stored in the **job queue**.
- Processes in the Ready state are placed in the **ready queue**.
- Processes waiting for a device to become available are placed in **device queues**. There are unique device queues for each I/O device available.

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

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



In the first two cases, the process eventually switches from the waiting state to the ready state, and is then put back in the ready queue. A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

### Types of Schedulers

There are three types of schedulers available :

#### 1. Long Term Scheduler :

Long term scheduler runs less frequently. Long Term Schedulers decide which program must get into the job queue. From the job queue, the Job Processor, selects processes and loads them into the memory for execution. Primary aim of the Job Scheduler is to maintain a good degree of Multiprogramming. An optimal degree of Multiprogramming means the average rate of process creation is equal to the average departure rate of processes from the execution memory.

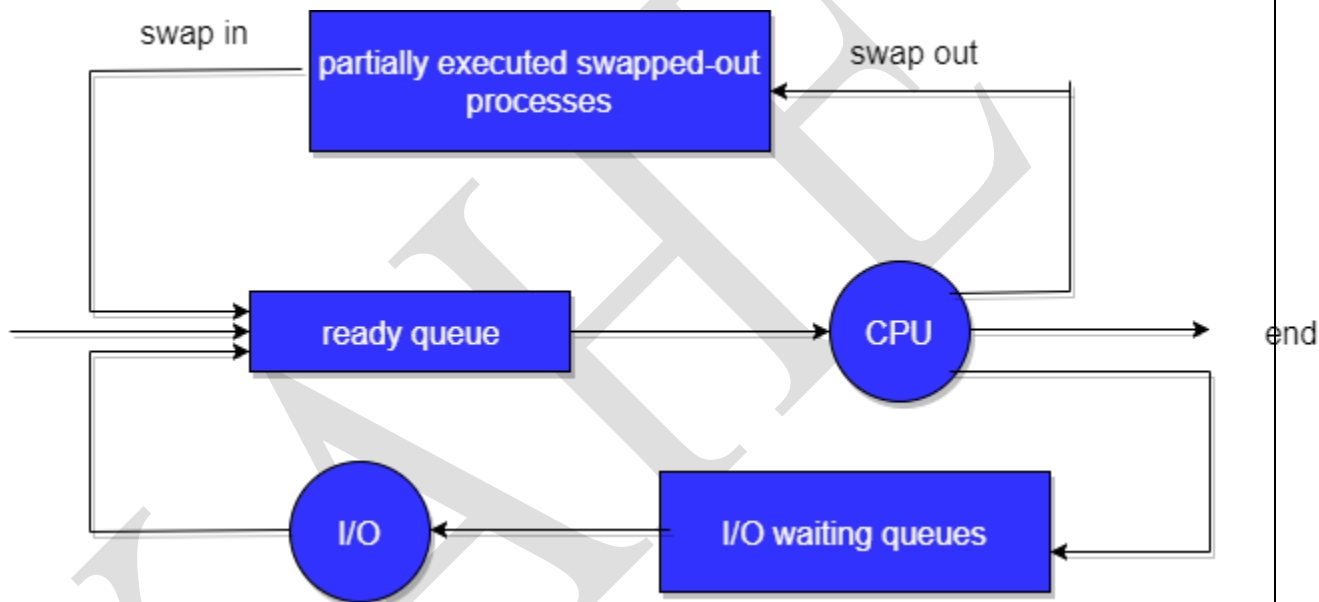
#### 2. Short Term Scheduler :

This is also known as CPU Scheduler and runs very frequently. The primary aim of this scheduler is to enhance CPU performance and increase process execution rate.

### 3. Medium Term Scheduler :

This scheduler removes the processes from memory (and from active contention for the CPU), and thus reduces the degree of multiprogramming. At some later time, the process can be reintroduced into memory and its execution can be continued where it left off. This scheme is called **swapping**. The process is swapped out, and is later swapped in, by the medium term scheduler.

Swapping may be necessary to improve the process mix, or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up. This complete process is described in the below diagram:



Addition of medium-term scheduling to the queueing diagram.

### Context Switch

- Switching the CPU to another process requires **saving** the state of the old process and **loading** the saved state for the new process. This task is known as a **context switch**.
- The **context** of a process is represented in the **Process Control Block(PCB)** of a process; it includes the value of the CPU registers, the process state and memory-management information. When a context switch occurs, the Kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.
- Context switch time is **pure overhead**, because the **system does no useful work while switching**. Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions (such as a single instruction to load or store all registers). Typical speeds range from 1 to 1000 microseconds.
- Context Switching has become such a performance **bottleneck** that programmers are using new structures (threads) to avoid it whenever possible.

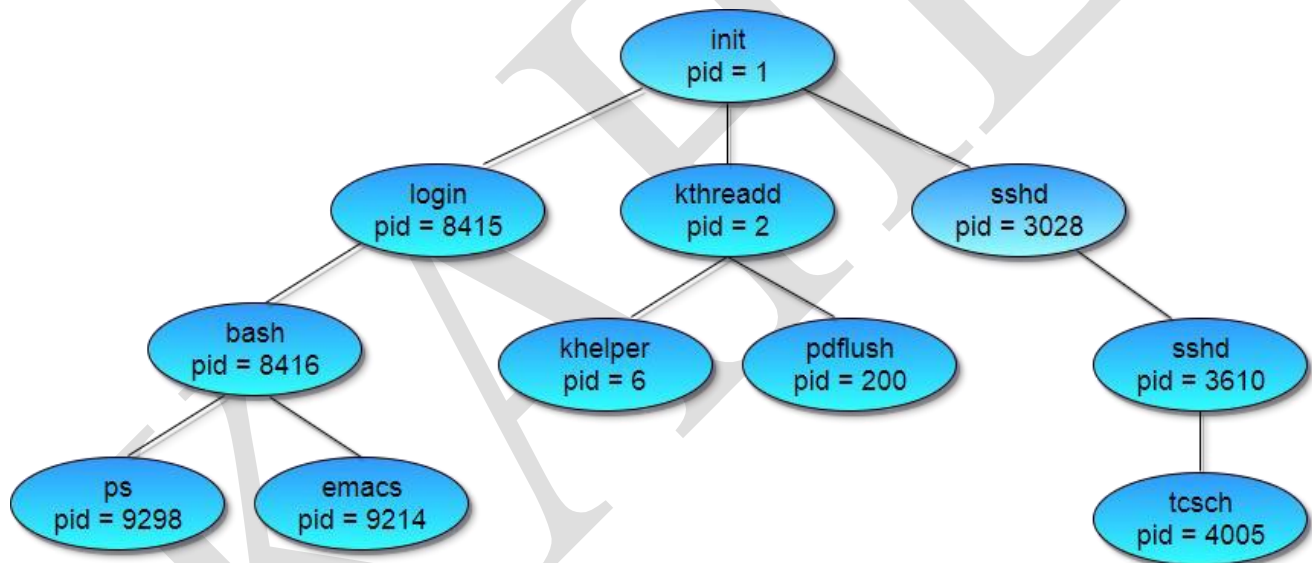
## Operations on Process

### Process Creation

Through appropriate system calls, such as fork or spawn, processes may create other processes. The process which creates other process, is termed the **parent** of the other process, while the created sub-process is termed its **child**.

Each process is given an integer identifier, termed as process identifier, or PID. The parent PID (PPID) is also stored for each process.

On a typical UNIX systems the process scheduler is termed as sched, and is given PID 0. The first thing done by it at system start-up time is to launch init, which gives that process PID 1. Further Init launches all the system daemons and user logins, and becomes the ultimate parent of all other processes.



A child process may receive some amount of shared resources with its parent depending on system implementation. To prevent runaway children from consuming all of a certain system resource, child processes may or may not be limited to a subset of the resources originally allocated to the parent.

There are two options for the parent process after creating the child :

- Wait for the child process to terminate before proceeding. Parent process makes a wait() system call, for either a specific child process or for any particular child process, which causes the parent process to block until the wait() returns. UNIX shells normally wait for their children to complete before issuing a new prompt.
- Run concurrently with the child, continuing to process without waiting. When a UNIX shell runs a process as a background task, this is the operation seen. It is also possible for the parent to run for a while, and then wait for the child later, which might occur in a sort of a parallel processing operation.

There are also two possibilities in terms of the address space of the new process:

1. The child process is a duplicate of the parent process.
2. The child process has a program loaded into it.

To illustrate these different implementations, let us consider the **UNIX** operating system. In UNIX, each process is identified by its **process identifier**, which is a unique integer. A new process is created by the **fork** system call. The new process consists of a copy of the address space of the original process. This mechanism allows the parent process to communicate easily with its child process. Both processes (the parent and the child) continue execution at the instruction after the fork system call, with one difference: **The return code for the fork system call is zero for the new(child) process, whereas the(non zero) process identifier of the child is returned to the parent.**

Typically, the **execvp system call** is used after the fork system call by one of the two processes to replace the process memory space with a new program. The execvp system call loads a binary file into memory - destroying the memory image of the program containing the execvp system call – and starts its execution. In this manner the two processes are able to communicate, and then to go their separate ways.

Below is a C program to illustrate forking a separate process using UNIX(made using Ubuntu):

```
#include<stdio.h>

void main(int argc, char *argv[])
{
    int pid;

    /* Fork another process */
    pid=fork();

    if(pid<0)
    {
        //Error occurred
        fprintf(stderr, "Fork Failed");
        exit(-1);
    }
    else if (pid == 0)
    {
        //Child process
        execvp("/bin/ls","ls",NULL);
    }
    else
    {
        //Parent process
        //Parent will wait for the child to complete
        wait(NULL);
        printf("Child complete");
        exit(0);
    }
}
```

**Gate Numerical Tip:** if fork is called for  $n$  times, the number of child process or new process created are:  $2^n - 1$ .

### Process Termination

By making the `exit()` system call, typically returning an int, processes may request their own termination. This int is passed along to the parent if it is doing a `wait()`, and is typically zero on successful completion and some non-zero code in the event of any problem.

Processes may also be terminated by the system for a variety of reasons, including :

- The inability of the system to deliver the necessary system resources.
- In response to a KILL command or other unhandled process interrupts.
- A parent may kill its children if the task assigned to them is no longer needed i.e. if the need of having a child terminates.
- If the parent exits, the system may or may not allow the child to continue without a parent (In UNIX systems, orphaned processes are generally inherited by `init`, which then proceeds to kill them.)

When a process ends, all of its system resources are freed up, open files flushed and closed, etc. The process termination status and execution times are returned to the parent if the parent is waiting for the child to terminate, or eventually returned to `init` if the process already became an orphan.

The processes which are trying to terminate but cannot do so because their parent is not waiting for them are termed **zombies**. These are eventually inherited by `init` as orphans and killed off.

### Operating System Scheduling algorithms

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter –

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin (RR) Scheduling
- Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive** or **preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

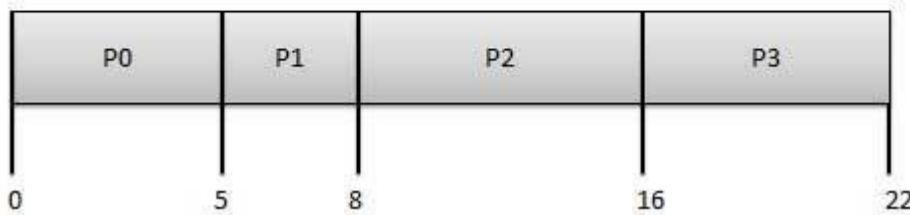
#### First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.



- 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 as follows –

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:  $(0+4+6+13) / 4 = 5.75$

#### Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processor should know in advance how much time process will take.

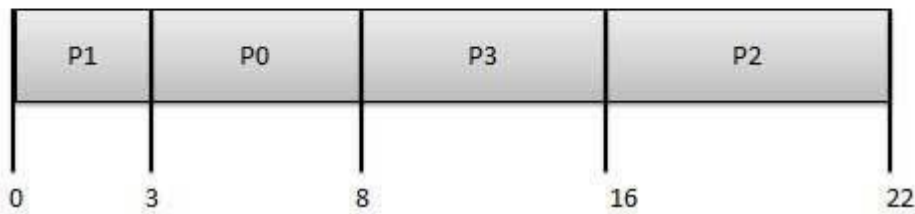
## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.Sc Maths  
COURSE CODE: 17MMU404B

COURSE NAME: OPERATING SYSTEMS: LINUX  
UNIT: III Process Scheduling

BATCH-2017-2020

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



Wait time of each process is as follows –

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

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- 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 served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

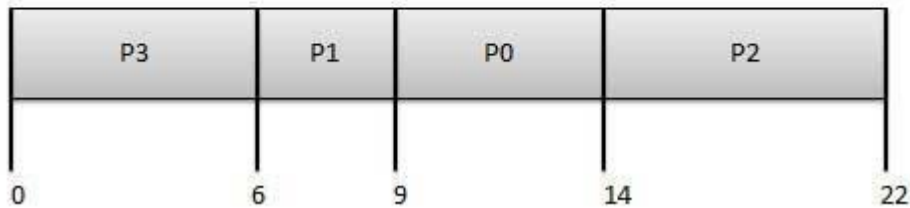
## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.Sc Maths  
COURSE CODE: 17MMU404B

COURSE NAME: OPERATING SYSTEMS: LINUX  
UNIT: III Process Scheduling

BATCH-2017-2020

| Process | Arrival Time | Execute Time | Priority | Service Time |
|---------|--------------|--------------|----------|--------------|
| P0      | 0            | 5            | 1        | 9            |
| P1      | 1            | 3            | 2        | 6            |
| P2      | 2            | 8            | 1        | 14           |
| P3      | 3            | 6            | 3        | 0            |



**Wait time** of each process is as follows –

**Process**                      **Wait Time : Service Time - Arrival Time**

P0                                       $9 - 0 = 9$

P1                                       $6 - 1 = 5$

P2                                       $14 - 2 = 12$

P3                                       $0 - 0 = 0$

Average Wait Time:  $(9+5+12+0) / 4 = 6.5$

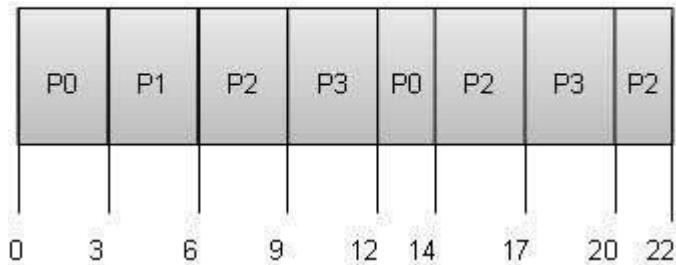
### Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

### Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum = 3



**Wait time** of each process is as follows –

**Process**                      **Wait Time : Service Time - Arrival Time**

P0                                       $(0 - 0) + (12 - 3) = 9$

P1                                       $(3 - 1) = 2$

P2                                       $(6 - 2) + (14 - 9) + (20 - 17) = 12$

P3                                       $(9 - 3) + (17 - 12) = 11$

Average Wait Time:  $(9+2+12+11) / 4 = 8.5$

### Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

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

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

### **Operating System | Multiple-Processor Scheduling**

1

In multiple-processor scheduling **multiple CPU's** are available and hence **Load Sharing** becomes possible. However multiple processor scheduling is more **complex** as compared to single processor scheduling. In multiple processor scheduling there are cases when the processors are identical i.e. **HOMOGENEOUS**, in terms of their functionality, we can use any processor available to run any process in the queue.

### Approaches to Multiple-Processor Scheduling –

One approach is when all the scheduling decisions and I/O processing are handled by a single processor which is called the **Master Server** and the other processors execute only the **user code**. This is simple and reduces the need of data sharing. This entire scenario is called **Asymmetric Multiprocessing**.

A second approach uses **Symmetric Multiprocessing** where each processor is **self scheduling**. All processes may be in a common ready queue or each processor may have its own private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.

### Processor Affinity –

Processor Affinity means a process has an **affinity** for the processor on which it is currently running. When a process runs on a specific processor there are certain effects on the cache memory. The data most recently accessed by the process populates the cache for the processor and as a result successive memory access by the process are often satisfied in the cache memory. Now if the process migrates to another processor, the contents of the cache memory must be invalidated for the first processor and the cache for the second processor must be repopulated. Because of the high cost of invalidating and repopulating caches, most of the SMP (symmetric multiprocessing) systems try to avoid migration of processes from one processor to another and try to keep a process running on the same processor. This is known as **PROCESSOR AFFINITY**.

There are two types of processor affinity:

1. **Soft Affinity** – When an operating system has a policy of attempting to keep a process running on the same processor but not guaranteeing it will do so, this situation is called soft affinity.
2. **Hard Affinity** – Some systems such as Linux also provide some system calls that support Hard Affinity which allows a process to migrate between processors.

### Load Balancing –

Load Balancing is the **phenomena** which keeps the **workload** evenly **distributed** across all processors in an SMP system. Load balancing is necessary only on systems where each processor has its own private queue of process which are eligible to execute. Load balancing is unnecessary because once a processor becomes idle it immediately extracts a runnable process from the common run queue. On SMP (symmetric multiprocessing), it is important to keep the workload balanced among all processors to fully utilize the benefits of having more than one processor else one or more processors will sit idle while other processors have high workloads along with lists of processors awaiting the CPU.

There are two general approaches to load balancing :

1. **Push Migration** – In push migration a task routinely checks the load on each processor and if it finds an imbalance then it evenly distributes load on each processor by moving the processes from overloaded to idle or less busy processors.
2. **Pull Migration** – Pull Migration occurs when an idle processor pulls a waiting task from a busy processor for its execution.

### **Multicore Processors –**

In multicore processors **multiple processor** cores are placed on the same physical chip. Each core has a register set to maintain its architectural state and thus appears to the operating system as a separate physical processor. **SMP systems** that use multicore processors are faster and consume **less power** than systems in which each processor has its own physical chip.

However multicore processors may **complicate** the scheduling problems. When processor accesses memory then it spends a significant amount of time waiting for the data to become available. This situation is called **MEMORY STALL**. It occurs for various reasons such as cache miss, which is accessing the data that is not in the cache memory. In such cases the processor can spend up to fifty percent of its time waiting for data to become available from the memory. To solve this problem recent hardware designs have implemented multithreaded processor cores in which two or more hardware threads are assigned to each core. Therefore if one thread stalls while waiting for the memory, core can switch to another thread.

There are two ways to multithread a processor :

1. **Coarse-Grained Multithreading** – In coarse grained multithreading a thread executes on a processor until a long latency event such as a memory stall occurs, because of the delay caused by the long latency event, the processor must switch to another thread to begin execution. The cost of switching between threads is high as the instruction pipeline must be terminated before the other thread can begin execution on the processor core. Once this new thread begins execution it begins filling the pipeline with its instructions.
2. **Fine-Grained Multithreading** – This multithreading switches between threads at a much finer level mainly at the boundary of an instruction cycle. The architectural design of fine grained systems include logic for thread switching and as a result the cost of switching between threads is small.

### **Virtualization and Threading –**

In this type of **multiple-processor** scheduling even a single CPU system acts like a multiple-processor system. In a system with Virtualization, the virtualization presents one or more virtual CPU's to each of virtual machines running on the system and then schedules the use of physical CPU'S among the virtual machines. Most virtualized environments have one host operating system and many guest operating systems. The host operating system creates and manages the virtual machines and each virtual machine has a guest operating system installed and applications running within that guest. Each guest operating system may be assigned for specific use cases, applications, and users, including time sharing or even real-time operation. Any guest operating-system scheduling algorithm that assumes a certain amount of progress in a given amount of time will be negatively impacted by the virtualization. In a time sharing operating system that tries to allot 100 milliseconds to each time slice to give users a reasonable response time. A given 100 millisecond time slice may take much more than 100 milliseconds of virtual CPU time. Depending on how busy the system is, the time slice may take a second or more which results in a very poor response time for users logged into that virtual machine. The net effect of such scheduling layering is that individual virtualized operating systems receive only a portion of the available CPU cycles, even though they believe they are receiving all cycles and that they are scheduling all of those cycles. Commonly, the time-of-day clocks in virtual machines are incorrect because timers take no longer to trigger than they would on dedicated CPU's.

**Virtualizations** can thus undo the good scheduling-algorithm efforts of the operating systems within virtual machines.

### **Real Time Schedule**

#### **Real-time operating systems**

If you are serious about meeting hard deadlines or designing a safety critical system, you will likely need to run a **real-time operating system** on a system that is dedicated to those tasks and minimizes all other forms of interference. For example, you may not be able to afford wasting time servicing disk interrupts and you certainly will not want to move memory pages back and forth between the disk and memory for fear of the time it will take to retrieve them if they are suddenly needed. You also will need to ensure that your operating system has preemptable system calls since you don't want a process held up because the operating system is tied up servicing a system call.

A real-time operating system has a well-specified maximum time for each action that it performs to support applications with precise timing needs. Systems that can guarantee these maximum times are called **hard real-time** systems. Those that meet these times most of the time are called **soft real-time** systems. Deploying an airbag in response to a sensor being actuated is a case where you would want a hard real-time system. Decoding video frames is an example of where a soft real-time system will suffice. Real-time systems will usually have the following characteristics:

- Priority-based scheduler
- Guaranteed maximum time to service interrupts
- Ability to ensure that processes are fully loaded into memory and stay in memory
- Consistent and efficient memory allocation
- Preemptable system calls

#### **Process types**

As we start to use terms such as *compute time* and *deadline*, it helps to see how these terms relate to different categories of processes:

1. **Terminating processes:** A terminating process may be considered as one that runs and then exits (terminates). We are interested in the amount of time that it takes it to run to completion. Its deadline is the time that at which it should complete all its work and its compute time is the amount of CPU time it needs.
2. **Nonterminating processes:** For processes such as video and audio servers as well as editors, we are not interested in the terminating time of these processes but rather in the time between events. For example, we would like our audio server to fill a 4K byte audio buffer every 500 milliseconds or we would like our video server to provide a new video frame every 33.3 milliseconds. For these processes, the compute time is the CPU time that the process needs to compute its periodic event and the deadline is the time at which it must have the results ready. Nonterminating processes may be divided into two classes:
  - **Periodic:** A periodic process has a fixed frequency at which it needs to run. For example, a video decompressor may have to run 30 times per second at 33.3 millisecond intervals.
  - **Aperiodic:** Aperiodic processes have no fixed, cyclical, frequency between events. Event interrupts may occur sporadically and event computation times may vary dramatically. For



purposes of scheduling, we use the shortest period and the longest computation time to play it safe.

### How much can we do?

The CPU cannot work magic. If we want to have our system process four video streams at the same time at 30 frames per second and processing a single frame requires 40 milliseconds of CPU time, we will be forced to fail in our scheduling needs. There is just not enough CPU time to go around. If  $C$  represents our computation time and  $D$  represents the deadline, the following relation must hold:

$$C \leq D$$

This assures us that we will have enough CPU time to complete the task. Moreover, for periodic tasks, the deadline must be within the period. If the period of the process is  $T$ , the following relation must now hold:

$$C \leq D \leq T$$

Let's now look at a few popular algorithms for scheduling processes with real-time constraints.

### Earliest deadline scheduling

Earliest deadline scheduling is simple in concept. Every process tells the operating system scheduler its absolute time deadline. The scheduling algorithm simply allows the process that is in the greatest danger of missing its deadline to run first. Generally, this means that one process will run to completion if it has an earlier deadline than another. The only time a process would be preempted would be when a new process with an even shorter deadline becomes ready to run. To determine whether all the scheduled processes are capable of having their deadlines met, the following condition must hold :

$$\sum_i \frac{C_i}{T_i} \leq 1$$

This simply tells us sum of all the percentages of CPU time used per process has to be less than or equal to 100%.

### Least slack scheduling

This method is similar to shortest remaining time scheduling with the concept of a deadline thrown in. The goal is to pick the process that we can least afford to delay. This differs from earliest deadline scheduling because we're not looking only at the deadline, but at the amount of time we can procrastinate (work on something else) before we will have to put 100% of the CPU resource to finishing the task. Least slack is computed as the time to the deadline minus the amount of computation. For example, suppose that our remaining computation time,  $C$ , is 5 msec. and the deadline,  $D$ , is 20 msec. from now. The slack time is  $D - C$ , or 15 msec. The scheduler will compare this slack time with the slack times of other processes in the system and run the one with the lowest slack time.

The effect of least slack scheduling is significantly different from that of earliest deadline scheduling. With earliest deadline, we will always work on the process with the nearest deadline, delaying all the



other processes. With least slack scheduling, we get a balanced result in that we attempt to keep the differences from deadlines balanced among processes. If the CPU has no problem meeting all deadlines, both scheduling policies will work satisfactorily. If there is too much of a workload and some deadlines must be missed, earliest deadline will satisfy the processes with the earliest deadlines (assuming all processes arrived at the same time) because it started working on them early. Processes with later deadlines will get delayed significantly. With least slack scheduling, all deadlines will be missed, but they all will be missed by roughly the same amount of time. Which is better? It depends on the applications. The same scheduling constraint applies to Least Slack scheduling as to Earliest Deadline First scheduling.

### **Rate monotonic analysis**

**Rate monotonic analysis** is a technique for assigning static priorities to periodic processes. As such, it is not a scheduler but a mechanism for governing the behavior of a preemptive priority scheduler. A conventional priority scheduler is used with this system, where the highest priority ready process will always get scheduled, preempting any lower priority processes.

A scheduler that is aware of rate monotonic scheduling would be provided with process timing parameters (period of execution) when the process is created and compute a suitable priority for the process. Most schedulers that support priority scheduling (e.g., Windows, Linux, Solaris, FreeBSD, NetBSD) do not perform rate monotonic analysis but only allow fixed priorities, so it is up to the user to assign proper priority levels for all real-time processes on the system. To do this properly, the user must be aware of all the real-time processes that will be running at any given time and each process' frequency of execution ( $1/T$ , where  $T$  is the period). To determine whether all scheduled processes can have their real-time demands met, the system has to also know each process' compute needs per period ( $C$ ) and check that the following condition holds:

$$\sum_i \frac{C_i}{T_i} < \ln 2$$

To assign a rate monotonic priority, one simply uses the frequency information for each process. If a process is an aperiodic process, the worst-case (fastest) frequency should be used. The highest frequency (smallest period) process gets the highest priority and successively lower frequency processes get lower priorities.

Scheduling is performed by a simple priority scheduler. At each quantum, the highest priority ready process gets to run. Processes at the same priority level run round-robin.

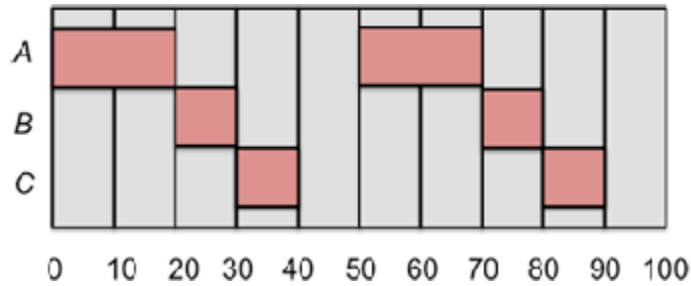
Here is an example of rate monotonic priority assignment. Suppose we have the following processes:

A runs every 50 msec for 20 msec

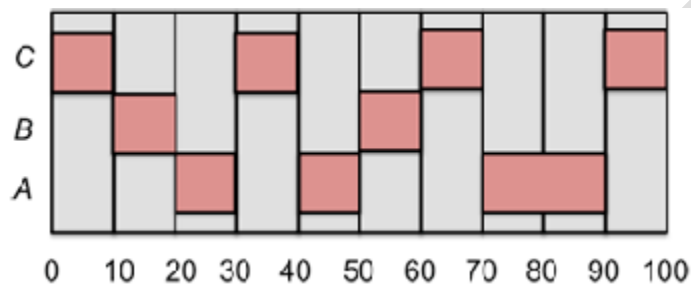
B runs every 50 msec for 10 msec

C runs every 30 msec for 10 msec

Rate-monotonic assignment requires that the highest frequency process(es) ( $B$  and  $C$ ) get the highest priority and  $A$ , having the lowest frequency of execution, gets a lower priority. If we do not follow the rules and give  $A$  the highest priority,  $B$  the next highest, and  $C$  the lowest, the CPU will run processes in the following order:



This does not give us the desired performance because, while processes *A* and *B* get scheduled acceptably, process *C* is late the second time it is scheduled and misses an entire period! Now let us reverse the priorities as rate monotonic assignment would dictate:



The scheduler can now satisfactorily meet the real-time requirements these tasks. Rate monotonic priority assignment is guaranteed to be optimal. If processes cannot be scheduled using rate monotonic assignment, the processes cannot be properly scheduled with any other static priority assignment.

### Deterministic Modeling

This evaluation method takes a predetermined workload and evaluates each algorithm using that workload.

Assume we are presented with the following processes, which all arrive at time zero.

| Process | Burst Time |
|---------|------------|
| P1      | 9          |
| P2      | 33         |
| P3      | 2          |
| P4      | 5          |
| P5      | 14         |

Which of the following algorithms will perform best on this workload?

First Come First Served (FCFS), Non Preemptive Shortest Job First (SJF) and Round Robin (RR). Assume a quantum of 8 milliseconds.

Before looking at the [answers](#), try to calculate the figures for each algorithm.

The advantages of deterministic modeling is that it is exact and fast to compute. The disadvantage is that it is only applicable to the workload that you use to test. As an example, use the above workload but assume P1 only has a burst time of 8 milliseconds. What does this do to the average waiting time?

Of course, the workload might be typical and scale up but generally deterministic modeling is too specific and requires too much knowledge about the workload.

### Queuing Models

Another method of evaluating scheduling algorithms is to use queuing theory. Using data from real processes we can arrive at a probability distribution for the length of a burst time and the I/O times for a process. We can now generate these times with a certain distribution.

We can also generate arrival times for processes (arrival time distribution).

If we define a queue for the CPU and a queue for each I/O device we can test the various scheduling algorithms using queuing theory.

Knowing the arrival rates and the service rates we can calculate various figures such as average queue length, average wait time, CPU utilization etc.

One useful formula is **Little's Formula**.

$$n = \lambda w$$

Where

$n$  is the average queue length

$\lambda$  is the average arrival rate for new processes (e.g. five a second)

$w$  is the average waiting time in the queue

Knowing two of these values we can, obviously, calculate the third. For example, if we know that eight processes arrive every second and there are normally sixteen processes in the queue we can compute that the average waiting time per process is two seconds.

The main disadvantage of using queuing models is that it is not always easy to define realistic distribution times and we have to make assumptions. This results in the model only being an approximation of what actually happens.

### Simulations

Rather than using queuing models we simulate a computer. A Variable, representing a clock is incremented. At each increment the state of the simulation is updated.

Statistics are gathered at each clock tick so that the system performance can be analysed.

The data to drive the simulation can be generated in the same way as the queuing model, although this leads to similar problems.

Alternatively, we can use trace data. This is data collected from real processes on real machines and is fed into the simulation. This can often provide good results and good comparisons over a range of scheduling algorithms.

However, simulations can take a long time to run, can take a long time to implement and the trace data may be difficult to collect and require large amounts of storage.

### **Implementation**

The best way to compare algorithms is to implement them on real machines. This will give the best results but does have a number of disadvantages.

- It is expensive as the algorithm has to be written and then implemented on real hardware.
- If typical workloads are to be monitored, the scheduling algorithm must be used in a live situation. Users may not be happy with an environment that is constantly changing.
- If we find a scheduling algorithm that performs well there is no guarantee that this state will continue if the workload or environment changes.

### **POSSIBLE QUESTIONS**

**PART –B**

**(Each Question carries 2 Marks)**

1. Define : Process
2. Define: Scheduler
3. What is preemptive scheduling?
4. What is non preemptive scheduling?
5. List the scheduling algorithms
6. What is simulation?
7. What is modeling?
8. What is deterministic modeling?
9. What is real time schedule?
10. What is queue model?

**PART –C**

**(Each Question carries 6 Marks)**

1. Write short notes on FCFS scheduling algorithm.
2. Explain the shortest job first scheduling algorithm.
3. Write a algorithm for round robin and explain
4. Write a algorithm for priority queue and explain
5. Explain the multilevel queue algorithm
6. Discuss the concept of real time schedule
7. Discuss the concept of deterministic modeling in detail
8. Explain the queue model.
9. What is simulation? Discuss in detail
10. Discuss any two scheduling algorithms.

**Semester : III****Subject : Operating Systems: Linux**

| S.NO     | Question   | Opt1            |
|----------|--|-----------------|
| UNIT-III |  |                 |
|          |  |                 |
| 1        | _____ is the basis of multiprogrammed operating system.  | RR scheduling   |
| 2        | A _____ is executed until it must wait, typically for the completion of some i/o request             | reverse         |
| 3        | _____ is a fundamental operating system function.  | RR              |
| 4        | Process execution begins with a _____  | CPU burst       |
| 5        | The operating system must select one of the processes in the ready queue to be executed by the _____ | nonpreemptive   |
| 6        | When scheduling takes place only under circumstances 1 and 4 called _____                            | variable class  |
| 7        | Another component involved in the CPU scheduling function is the _____                               | central edge    |
| 8        | One measure of work is the number of processes completed per time unit called _____                  | throughput      |
| 9        | Which of the following is the simplest scheduling discipline?  | FCFS scheduling |
| 10       | In which scheduling, processes are dispatched according to their arrival time on the ready queue?    | FCFS scheduling |
| 11       | In which scheduling, processes are dispatched FIFO but are given a limited amount of CPU time?       | FIFO scheduling |
| 12       | Which scheduling is effective in time sharing environments   | FIFO scheduling |
| 13       | Variable size blocks are called  | Pages           |
| 14       | Which scheduling is effective in time sharing environments   | FIFO scheduling |
| 15       | Which of the following is non-preemptive scheduling?   | RR scheduling   |

|    |   |  |
|----|---|--|
| 16 | The interval from the time of submission of a process to the time of completion is the _____  | Queues   |
| 17 | The simplest CPU scheduling algorithm is the  | FCS  |
| 18 | Multiprogramming was made possible by _____.  | input/output units that operate independently of the CPU |
| 19 | The SJF algorithm is a special case of the general _____ algorithm  | FCS  |
| 20 | _____ scheduling algorithm is designed especially for time sharing systems.   | CFS  |
| 21 | The seek optimization strategy in which there is no reordering of the queue is called _____.  | FCFS   |
| 22 | A major problem with priority scheduling algorithms is  | tail   |
| 23 | If the time quantum is very small the RR approach is called _____   | Queues   |
| 24 | The seek optimization strategy in which the disk arm is positioned next at the request (inward or outward) that minimizes arm movement is called _____. | FCFS   |
| 25 | If several identical processors are available then _____ can occur.   | heterogeneous  |
| 26 | The high priority process would be waiting for a lower priority one to finish is called _____   | resources inversion                                      |
| 27 | _____ systems are required to complete a critical task within a guaranteed amount of time.  | hard real time   |
| 28 | The scheduler than either admits a process guaranteeing that the process will complete on time known as _____   | Priority inversion                                       |
| 29 | _____ uses the the given algorithm and the system workload to produce a formula.  | deterministic modelling                                  |
| 30 | If no thread is found the dispatcher will execute a special thread called _____   | variable class   |
| 31 | Deadlocks can be described more precisely in terms of a directed graph called   | resource graph   |

|    |   |  |
|----|---|--|
| 32 | _____ is the set of methods for ensuring that at least one of the necessary condition.  | Deadlock prevention                            |
| 33 | _____ is possible to construct an algorithm that ensures that the system will never enter the deadlock state.                       | Deadlock prevention                            |
| 34 | A system is in a safe state only if there exists a _____  | Safe state                                     |
| 35 | A critical section is a program segment _____.  | which should run in a certain specified amount |
| 36 | In addition to the request and assignment edges, we introduce a new type of edge called _____                                       | central edge                                   |
| 37 | The deadlock avoidance algorithm are described in next system but is less efficient than the resource allocation graph called _____ | Deadlock prevention                            |
| 38 | CPU Scheduling is the basis of _____ operating system.  | single programmed                              |
| 39 | Scheduling is a fundamental _____ function.   | computer                                       |
| 40 | Another component involved in the CPU _____ function is the dispatcher  | processing                                     |
| 41 | A major problem for priority _____ is starvation.   | sort algorithms                                |
| 42 | The seek _____ strategy in which there is no reordering of the queue is called SSTF   | processing                                     |
| 43 | The high _____ would be waiting for a lower priority one to finish is called priority inversion                                     | performance                                    |
| 44 | A _____ is a program segment where shared resources are accessed.   | critical section                               |
| 45 | If no thread is found, the _____ will execute a special thread called idle thread.  | degrader                                       |
| 46 | _____ execution begins with a CPU Burst.  | Process  |
| 47 | SJF Scheduling is an example for _____ scheduling.  | non- preemptive                                |
| 48 | _____ is the simplest CPU scheduling algorithm.   | FCFS   |
| 49 | Segments are called _____ blocks.   | equal size                                     |



|    |  |           |
|----|--|-----------|
| 50 | _____ can be described more precisely in terms of a directed graph.                          | semaphore |
| 51 | The interval from the time of submission of a process to the time of completion is the _____ | Queues    |

g

**of Higher Education**  
**ment of Mathematics**

Class : II B.Sc Maths

Subject Code:

17MMU404B

| Opt2                 | Opt3                | Opt4           | Answer               |
|----------------------|---------------------|----------------|----------------------|
|                      |                     |                |                      |
| Self Scheduling      | CPU sceduling       | throughput     | CPU sceduling        |
| deadlock avoidance   | deadlock            | process        | process              |
| CPU                  | Scheduling          | nonpreemptive  | Scheduling           |
| RR scheduling        | SJF scheduling      | SRT scheduling | CPU burst            |
| short term scheduler | long term scheduler | low level      | short term scheduler |
| real time class      | priority class      | nonpreemptive  | nonpreemptive        |
| dispatcher           | claim edge          | graph edge     | dispatcher           |
| variable class       | real time class     | priority class | throughput           |
| RR scheduling        | SJF scheduling      | SRT scheduling | FCFS schedulin       |
| RR scheduling        | SJF scheduling      | SRT scheduling | FCFS scheduling      |
| RR scheduling        | SJF scheduling      | SRT scheduling | RR scheduling        |
| RR scheduling        | SJF scheduling      | SRT scheduling | RR scheduling        |
| Segments             | Tables              | None           | Segments             |
| RR scheduling        | SJF scheduling      | SRT scheduling | RR scheduling        |
| SJF scheduling       | SRT scheduling      | None           | SJF scheduling       |

|                       |                                   |                          |                                   |
|-----------------------|-----------------------------------|--------------------------|-----------------------------------|
| Processor Sharing     | Sharing resources                 | turaround time           | turaround time                    |
| SJS                   | FCFS                              | DFG                      | FCFS                              |
| operating systems     | both (a) & (b) above              | neither (a) or (b) above | both (a) & (b) above              |
| SJS                   | Roundrobin                        | FCSC                     | Roundrobin                        |
| FSCS                  | priority                          | Round Robin              | Round Robin                       |
| SSTF                  | SCAN                              | C-SCAN                   | FCFS                              |
| Starvation            | time first                        | time quantum             | Starvation                        |
| Processor Sharing     | Sharing resources                 | Context switching        | Processor Sharing                 |
| SSTF                  | SCAN                              | C-SCAN                   | SSTF                              |
| homogeneous           | load sharing                      | UMA                      | load sharing                      |
| Priority inversion    | priority                          | Priority inheritance     | Priority inversion                |
| Priority inversion    | load sharing                      | Priority inheritance     | hard real time                    |
| resources reservation | load sharing                      | Sharing resources        | resources reservation             |
| scheduling process    | Analytic evaluation               | Queuing model            | Analytic evaluation               |
| real time class       | priority class                    | idle thread              | idle thread                       |
| system graph          | system resources allocation graph | request graph            | system resources allocation graph |

|                        |                                     |   |                                     |
|------------------------|-------------------------------------|---|-------------------------------------|
| deadlock avoidance     | handling deadlock                   | resource deadlock   | Deadlock prevention                 |
| deadlock avoidance     | handling deadlock                   | resource deadlock   | deadlock avoidance                  |
| unsafe state           | normal                              | deadlock  | Safe state                          |
| which avoids deadlocks | where shared resources are accessed | which must be enclosed by a pair of semaphore operations, P and V | where shared resources are accessed |
| graph edge             | claim edge                          | system edge   | claim edge                          |
| deadlock avoidance     | bankers algorithm                   | bankers allocation  | bankers algorithm                   |
| multi programmed       | multi system                        | multi disks   | multi programmed                    |
| operating system       | system resource                     | disk  | operating system                    |
| mathematical           | arithmetic                          | scheduling  | scheduling                          |
| scheduling algorithms  | search algorithms                   | manage algorithms   | scheduling algorithms               |
| scheduling             | optimization                        | implementation  | optimization                        |
| priority               | patent                              | graph edge  | priority                            |
| sub section            | cross section                       | class section   | critical section                    |
| scheduler              | dispatcher                          | redeemer  | dispatcher                          |
| Performance            | Purge                               | Put   | Process                             |
| preemptive             | emptive                             | prescheduling   | non- preemptive                     |
| LCFS                   | FCLS                                | LCFS  | FCFS                                |
| variable class         | variable size                       | big size  | variable size                       |

|                      |                   |                |                |
|----------------------|-------------------|----------------|----------------|
| deadlocks            | dumlocks          | starvation     | deadlocks      |
| Processor<br>Sharing | Sharing resources | turaround time | turaround time |

File systems: Introduction – File System Concepts – Access Methods – Directory structure – File Sharing – Allocation Methods – Free space management – Efficiency and performance – Recovery  
Disk Performance Optimization: Introduction – Disk structure – Disk scheduling – Disk management.

## **File System:**

### **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**

A File Structure should be according to a required format that the 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 have 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.

### **File Systems | Operating System**

A file is a collection of related information that is recorded on secondary storage. Or file is a collection of logically related entities. From user's perspective a file is the smallest allotment of logical secondary storage.

| <b>Attributes</b> | <b>Types</b> | <b>Operations</b> |
|-------------------|--------------|-------------------|
| Name              | Doc          | Create            |
| Type              | Exe          | Open              |
| Size              | Jpg          | Read              |
| Creation data     | Xis          | Write             |
| Author            | C            | Append            |
| Last modified     | Java         | Truncate          |
| protection        | class        | Delete            |
|                   |              | Close             |

There are various file types with their associated functions.



## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.Sc Maths  
COURSE CODE: 17MMU404B

COURSE NAME: OPERATING SYSTEMS: LINUX  
UNIT: IV File System  
BATCH-2017-2020

| File type      | Usual extension  | Function                                       |
|----------------|------------------|--|
| Executable     | exe,com,bin      | Read to run machine language program           |
| Object         | obj,o            | Compiled,machine language not linked           |
| Source code    | C,java,pas,asm,a | Source code in various languages               |
| Batch          | bat,sh           | Commands to the command interpreter            |
| Text           | txt,doc          | Textual data,documents                         |
| Word processor | Wp,tex,rrf,doc   | Various word processor formats                 |
| Archive        | arc,zip,tar      | Related files grouped into one file compressed |
|                |                  | for storage                                    |
| Multimedia     | mpeg,mov,rm      | File containing audio or a/v information       |

### FILE DIRECTORIES:

Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

### Information contained in a device directory are:

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner id
- Protection information

### Operation performed on directory are:

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

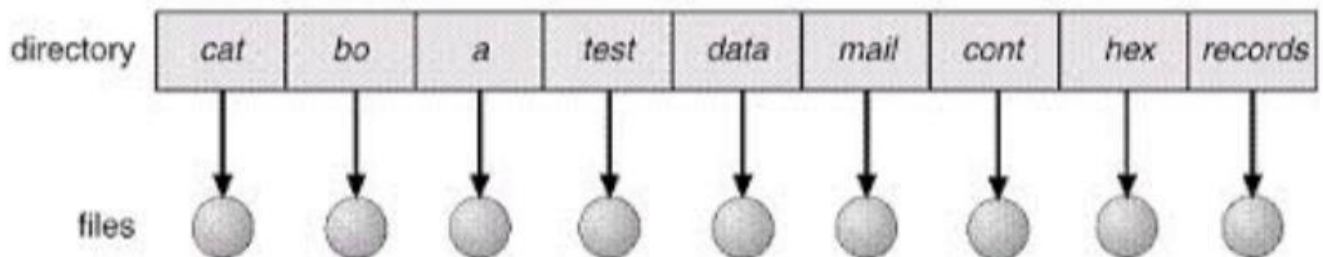
**Advantages of maintaining directories are:**

- **Efficiency:** A file can be located more quickly.
- **Naming:** It becomes convenient for users as two users can have same name for different files or may have different name for same file.
- **Grouping:** Logical grouping of files can be done by properties e.g. all java programs, all games etc.

**SINGLE-LEVEL DIRECTORY**

In this a single directory is maintained for all the users.

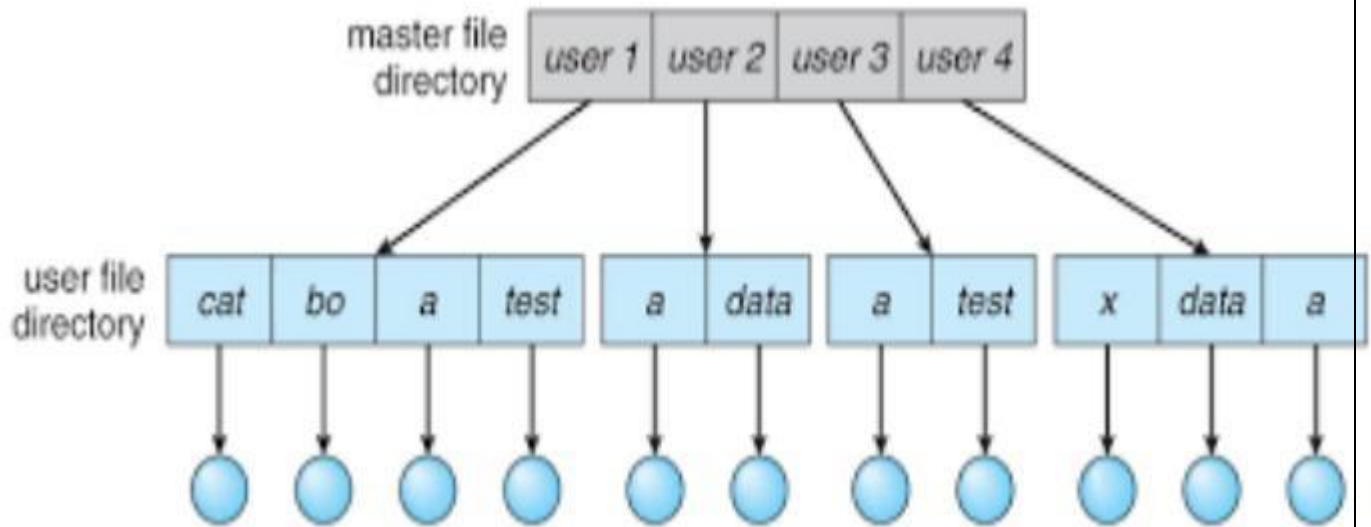
- **Naming problem:** Users cannot have same name for two files.
- **Grouping problem:** Users cannot group files according to their need.



**TWO-LEVEL DIRECTORY**

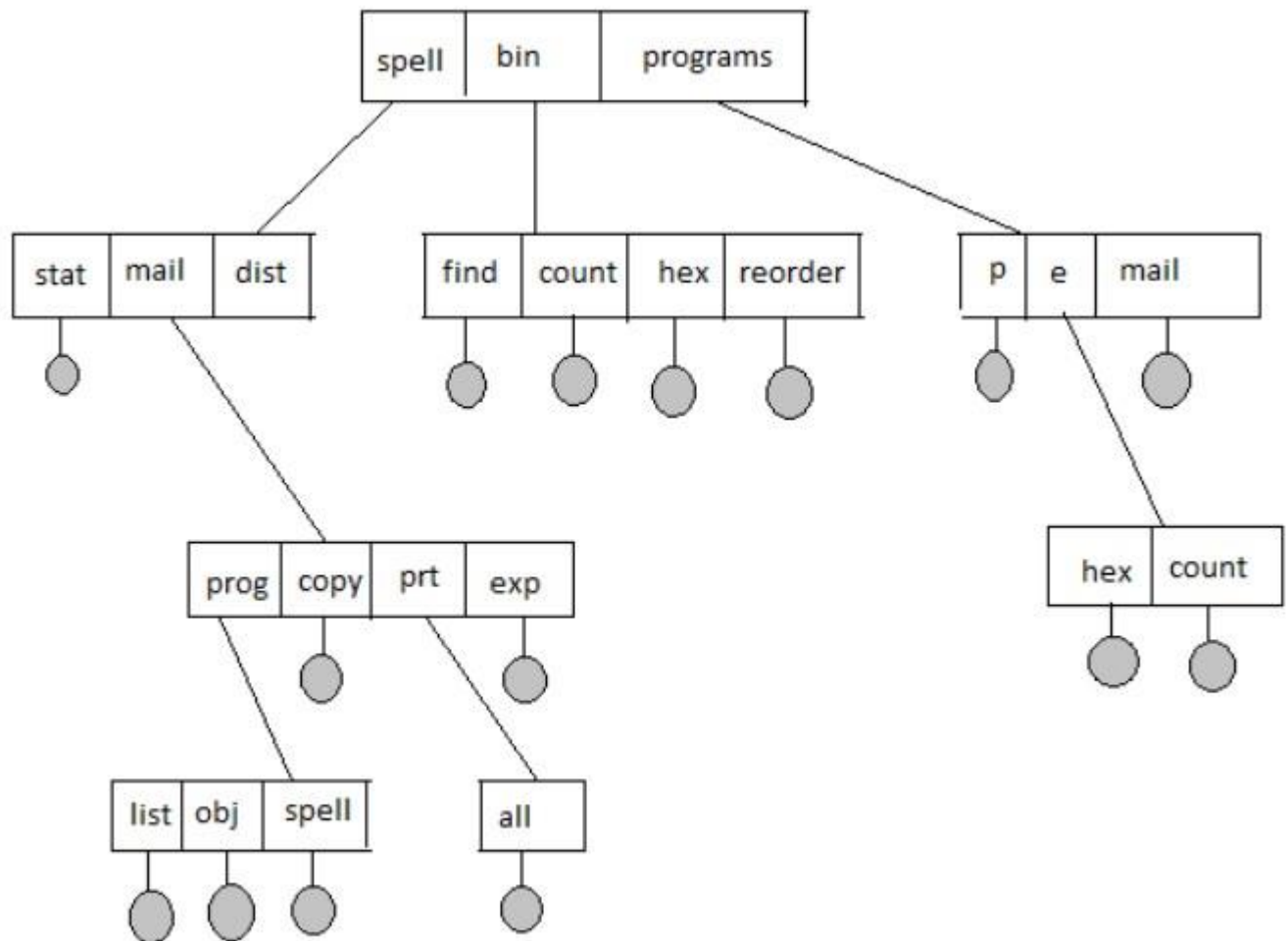
In this separate directories for each user is maintained.

- **Path name:** Due to two levels there is a path name for every file to locate that file.
- Now, we can have same file name for different user.
- Searching is efficient in this method.



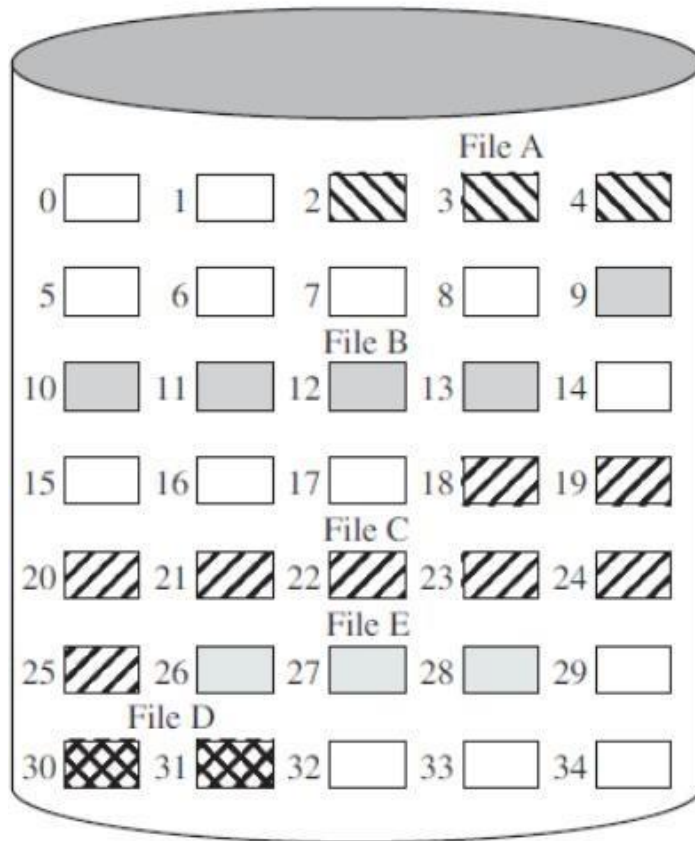
#### TREE-STRUCTURED DIRECTORY :

Directory is maintained in the form of a tree. Searching is efficient and also there is grouping capability. We have absolute or relative path name for a file.



## FILE ALLOCATION METHODS

**1. Continuous Allocation:** A single continuous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions. The file allocation table needs just a single entry for each file, showing the starting block and the length of the file. This method is best from the point of view of the individual sequential file. Multiple blocks can be read in at a time to improve I/O performance for sequential processing. It is also easy to retrieve a single block. For example, if a file starts at block  $b$ , and the  $i$ th block of the file is wanted, its location on secondary storage is simply  $b+i-1$ .



File allocation table

| File name | Start block | Length |
|-----------|-------------|--------|
| File A    | 2           | 3      |
| File B    | 9           | 5      |
| File C    | 18          | 8      |
| File D    | 30          | 2      |
| File E    | 26          | 3      |

### Disadvantage

- External fragmentation will occur, making it difficult to find contiguous blocks of space of sufficient length. Compaction algorithm will be necessary to free up additional space on disk.
- Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.

**2. Linked Allocation(Non-contiguous allocation) :** Allocation is on an individual block basis. Each block contains a pointer to the next block in the chain. Again the file table needs just a single entry for each file, showing the starting block and the length of the file. Although pre-allocation is possible, it is more common simply to allocate blocks as needed. Any free block can be added to the chain. The blocks need not be continuous. Increase in file size is always possible if free disk block is available. There is no external fragmentation because only one block at a time is needed but there can be internal fragmentation but it exists only in the last disk block of file.

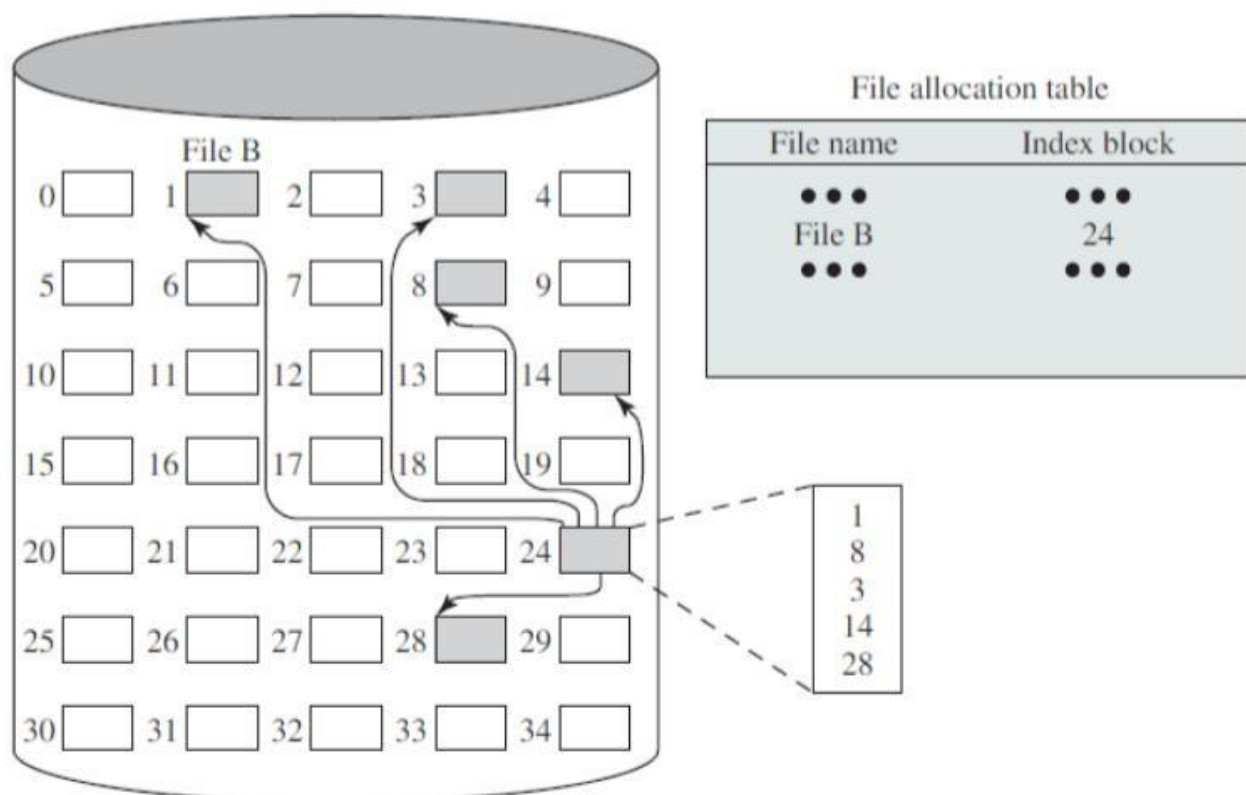
### Disadvantage:

- Internal fragmentation exists in last disk block of file.
- There is an overhead of maintaining the pointer in every disk block.
- If the pointer of any disk block is lost, the file will be truncated.

- It supports only the sequential access of files.

### 3. Indexed Allocation:

It addresses many of the problems of contiguous and chained allocation. In this case, the file allocation table contains a separate one-level index for each file: The index has one entry for each block allocated to the file. Allocation may be on the basis of fixed-size blocks or variable-sized blocks. Allocation by blocks eliminates external fragmentation, whereas allocation by variable-size blocks improves locality. This allocation technique supports both sequential and direct access to the file and thus is the most popular form of file allocation.



### Disk Free Space Management

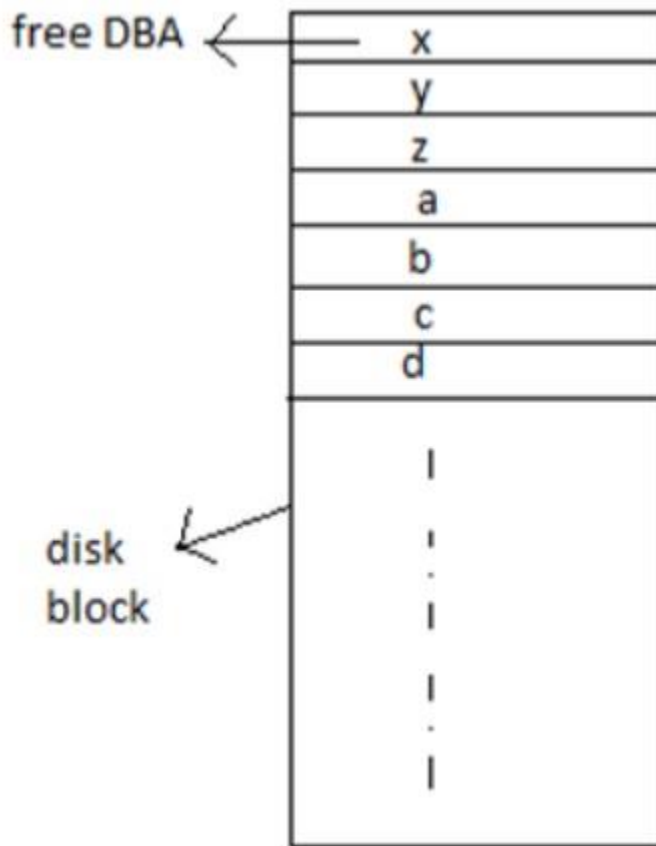
Just as the space that is allocated to files must be managed, so the space that is not currently allocated to any file must be managed. To perform any of the file allocation techniques, it is necessary to know what blocks on the disk are available. Thus we need a disk allocation table in addition to a file allocation table. The following are the approaches used for free space management.

1. **Bit Tables** : This method uses a vector containing one bit for each block on the disk. Each entry for a 0 corresponds to a free block and each 1 corresponds to a block in use.  
For example: 00011010111100110001

In this vector every bit correspond to a particular vector and 0 implies that, that particular block is free and 1 implies that the block is already occupied. A bit table has the advantage that it is

relatively easy to find one or a contiguous group of free blocks. Thus, a bit table works well with any of the file allocation methods. Another advantage is that it is as small as possible.

2. **Free Block List** : In this method, each block is assigned a number sequentially and the list of the numbers of all free blocks is maintained in a reserved block of the disk.



### Disk Scheduling Algorithms

#### 2.4

**Disk scheduling** is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

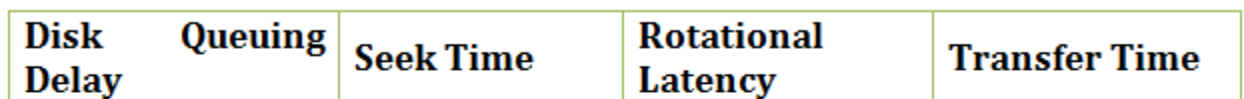
- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by disk controller. Thus other I/O requests need to wait in waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of computer system and thus need to be accessed in an efficient manner.



There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:** Disk Access Time is:

Disk Access Time = Seek Time +  
Rotational Latency +  
Transfer Time



← Disk Access Time →

← Disk Response Time →

- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

### **Disk Scheduling Algorithms**

1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time



- May not provide the best possible service
2. **SSTF:** In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

Advantages:

- Average Response Time decreases
- Throughput increases

Disadvantages:

- Overhead to calculate seek time in advance
  - Can cause Starvation for a request if it has higher seek time as compared to incoming requests
  - High variance of response time as SSTF favours only some requests
3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence also known as **elevator algorithm**. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantages:

- High throughput
- Low variance of response time
- Average response time

Disadvantages:

- Long waiting time for requests for locations just visited by disk arm
4. **CSCAN:** In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in CSAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Advantages:

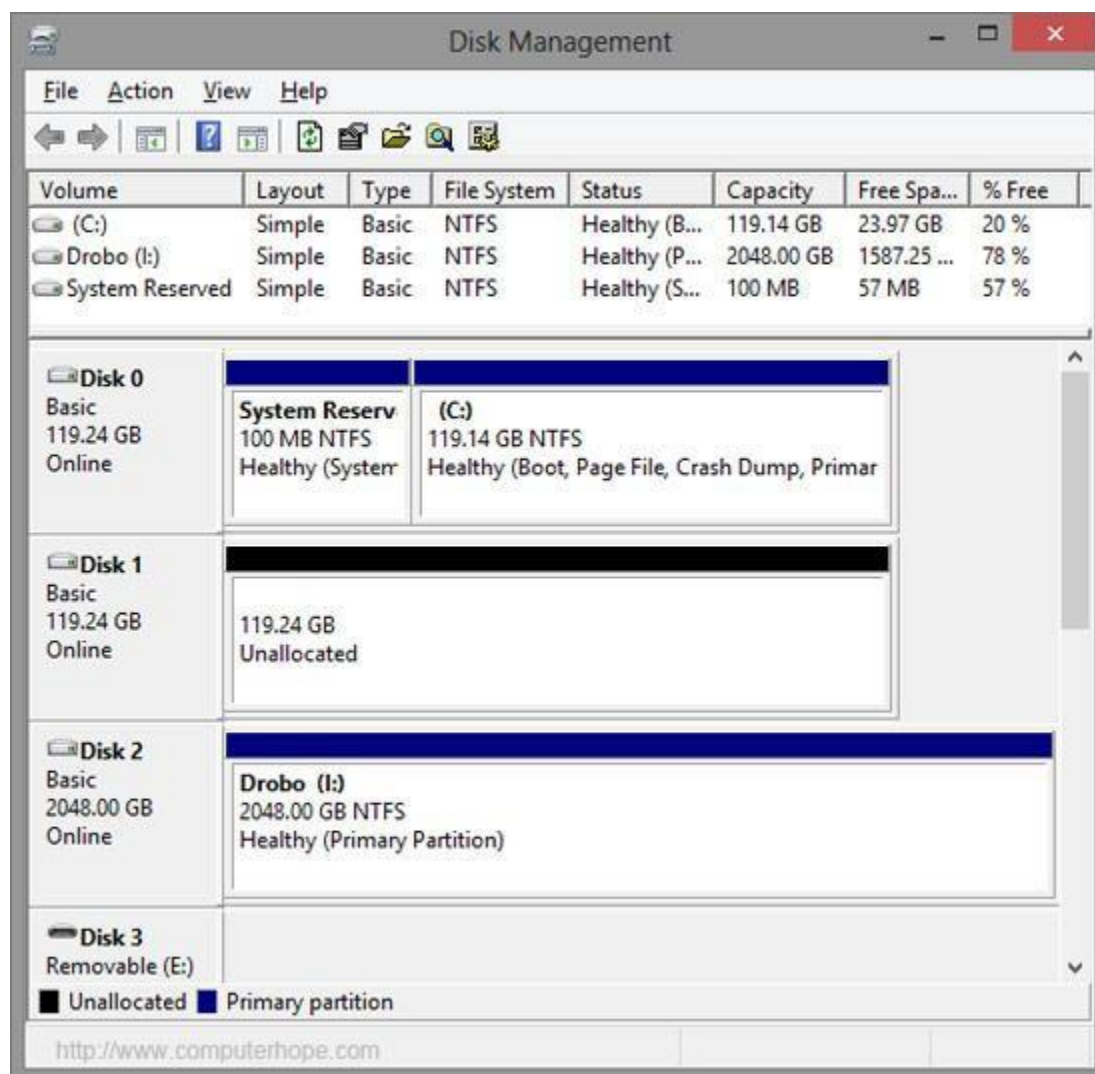
- Provides more uniform wait time compared to SCAN

5. **LOOK:** It is similar to the SCAN disk scheduling algorithm except the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.
6. **CLOOK:** As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm inspite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

### Disk Management

Updated: 11/10/2017 by Computer Hope

**Disk Management** is a [Microsoft Windows](#) utility first introduced in Windows XP as a replacement for the [fdisk command](#). It enables users to view and manage the [disk drives](#) installed in their computer and the [partitions](#) associated with those drives. As can be seen in the picture below, each drive is displayed followed by the layout, type, file system, status, capacity, free space, % free, and fault tolerance.



## How to open Windows Disk Management

1. Click the [Start](#) button and access the [Run](#) option. You can also press **Windows key + R** on the keyboard to open the Run option.
2. Type **diskmgmt.msc** and press Enter.

Tip: In Windows 8, you can type "diskmgmt.msc" directly on the Start screen to access Disk Management.

or

1. [Open the Control Panel](#).
2. Double-click on **Administrative Tools** if in Classic View or click **Performance and Maintenance** and then **Administrative Tools** if in Category View. Note: If you do not have [admin](#) rights to the computer this will not be available.
3. Once in the Administrative Tools window double-click **Computer Management** and then click **Disk Management** under the Storage section.

**POSSIBLE QUESTIONS**

**PART –B**

**(Each Question carries 2 Marks)**

- 1.What is file structure?
- 2.Define: Directory file
3. What is file Type?
- 4.List the file accessing mechanisms.
- 5.What is disk scheduling?
- 6.What is seek time?
- 7.Define: Rotational latency
- 8.What is shortest seek time first?
- 9.List the allocation methods
10. What is disk management?

**PART –C**

**(Each Question carries 6 Marks)**

- 1.Explain the various file accessing methods in detail.
- 2.Write short notes on file sharing.
- 3.Write short notes on disk structure.
- 4.Write short notes on disk scheduling.
- 5.What is disk management in detail?
- 6.Write short notes on free space management.
- 7.How do you achieve efficiency and performance of file system?
- 8.Write short notes on disk performance optimization.
- 9.Write a SCAN algorithm and explain.
10. Write a CSCAN algorithm and explain.

Semester : III

Subject : Operating Systems: Linux

| S.NO     | Question   | Opt1                  | Opt2               |
|----------|--|-----------------------|--------------------|
| UNIT -IV |  |                       |                    |
| 1        | The file system consist of -----<br>Distinct parts   | 2                     | 3                  |
| 2        | A File is a sequence of<br>character organized into lines  | Source                | Object             |
| 3        | A File is a sequence of<br>subroutines and functions   | Source                | Object             |
| 4        | The operating system keeps a small table called<br>the _____,containing<br>information about all open files                                | Show file table       | Visible file table |
| 5        | A file is executed in -----<br>extension   | External structure    | .bat               |
| 6        | The .bat file is a _____ containing<br>in ANCI format,command to the operating<br>system   | Binary file           | Batch file         |
| 7        | The file type is used to indicate the -----<br>----- of the file   | .txt                  | Internal structure |
| 8        | Information in the file is processed in the<br>order called-----   | Direct access         | Sequence access    |
| 9        | A file is made up of fixed length that allows the<br>program to read and write record rapidly in no<br>particular order is<br>called ----- | Direct access         | Sequence access    |
| 10       | Data cannot be written in secondary storage<br>unless written with in a -----<br>-----   | File                  | Swap space         |
| 11       | File attribute consist of -----  | Name,Type,Conte<br>nt | Name,type,Size     |
| 12       | The information about all files is kept in --<br>-----   | swap space            | operating system   |

|    |   |   |   |
|----|---|---|---|
| 13 | A file is a _____ type  | Abstract  | Primitive   |
| 14 | In UNIX Open system call returns -----<br>-----   | pointer to the entry in<br>the open file<br>table | pointer to the entry in<br>the<br>system wide table |
| 15 | The open file table has a -----<br>Associated with each file  | File content                                      | File permission                                     |
| 16 | The file name is generally split into which<br>of the two parts -----   | Name and type                                     | Name and<br>identifier                              |
| 17 | In the sequential access method,<br>information in the file is processed  | One disk after the<br>other                       | One record after<br>the other                       |
| 18 | Sequential access method _____,on<br>random access devices  | Works well  | Dosen't works<br>well                               |
| 19 | The direct access method is based on a ----<br>----- model of a file as-----<br>allow random access to any file block | Magnetic<br>tape,magnetic tapes                   | Tape,Tapes  |
| 20 | A relative block number is an index<br>relative to -----  | The beginning of<br>the file                      | The end of the file                                 |
| 21 | The index contains -----  | Name of all<br>content of file                    | Pointer to each<br>page                             |
| 22 | The directory can be viewed as a -----<br>---,that translate the file name into their directory<br>entries            | Symbol table                                      | Partition   |
| 23 | In the single level directory: -----<br>-----   | All files are contain in<br>different directories | All files are contained<br>in the same directory    |
| 24 | In the single level directory -----   | All directory must<br>have a unique<br>name       | All files must have a<br>unique<br>name             |
| 25 | In the two level directory structure -----<br>-----   | Each user has its own<br>user file<br>directory   | The system has its<br>own master file<br>directory  |
| 26 | When a user refers to a particular file -----<br>-----  | System MFD is<br>searched                         | His own UFD is<br>searched                          |

|    |  |   |   |
|----|--|---|---|
| 27 | The disadvantage of the two level directory structure is that  | It does not solve the name collision problem                | It solve the name collision problem       |
| 28 | In the tree structure directory -----<br>---   | The tree has the same directory                             | The tree has the leaf directory           |
| 29 | The three major methods of allocating disk space that are in wide use are -----<br>-----                     | Contiguous, Linked, Hashed                                  | Contiguous, Linked, Indexed               |
| 30 | In Contiguous allocation -----   | each file must occupy a set of contiguous block on the disk | Each file is a linked list of disk blocks |
| 31 | In linked allocation -----   | Each file must occupy a set of contiguous block on the disk | Each file is a linked list of disk blocks |
| 32 | In indexed allocation -----  | Each file must occupy a set of contiguous block on the disk | Each file is a linked list of disk blocks |
| 33 | One system where there are multiple operating system, the decision to load a particular one is done by ----- | Boot loader   | Boot strap                                |
| 34 | The VFS refers to -----  | Virtual File System   | Valid File System                         |
| 35 | The disadvantage of a linear list of directory entries is the -----  | Size of the linear list in the memory                       | Linear search to find a file              |
| 36 | One difficulty of contiguous allocation is -<br>-----  | Finding space for a new file                                | Ineffecient                               |

|    |  |  |   |
|----|--|--|---|
| 37 | To solve the problem of external fragmentation needs to be done periodically   | Compaction                               | Check                                     |
| 38 | If too little space is allocated to a file -----<br>-----  | The file will not work                   | There will not be any space               |
| 39 | A system program such as fsck -----<br>---- is a consistency checker   | UNIX                                     | Windows                                   |
| 40 | Each set of operations for performing a specific task is a -----   | Program                                  | Code                                      |
| 41 | Once the changes are written to the log, they are considered to be -----   | Committed                                | Aborted                                   |
| 42 | When an entire command transaction is completed,-----  | It is stored in the memory               | It is removed from the log file           |
| 43 | A circular buffer is -----   | Write to the end of its space            | Overwrite older value as it goes          |
| 44 | In ----- information is recorded magnetically on platters  | Magnetic disk                            | Electrical disk                           |
| 45 | The head of the magnetic disk are attached to a -----<br>that moves all the head as unit   | Spindle                                  | Disk arm                                  |
| 46 | The set of tracks that are at one arm position make up a -----   | Magnetic disk                            | Electrical disk                           |
| 47 | The time taken to move a disk arm to the desired cylinder is called as-----  | Positioning time                         | Random access ti                          |
| 48 | When a head damages the magnetic surface, it is known as -----   | Disk crash                               | Head crash                                |
| 49 | A floppy disk is designed to rotate -----<br>--- as compared to a hard disk drive  | Faster                                   | Slower                                    |
| 50 | The host controller is -----   | Controller built at the end of each disk | Controller at the computer end of the bus |
| 51 | The process of dividing a disk into sectors that the disk controller can read and write, before a disk can store data is known as----<br>----- | Partitioning                             | Swap space creation                       |



|    |  |              |                       |
|----|--|--------------|-----------------------|
| 52 | the data structure for a sector typically contains -----   | Header       | Data area             |
| 53 | The header and trailer of a sector contains information used by the disk controller such as -----.   | Main section | Error corecting codes |
| 54 | The two steps that the operating system takes to use a disk to hold its files are ----- and -----  | partitioning | Swap space creation   |
| 55 | The ----- program initializes all aspects of the system, from CPU registers to device controllers and the content of main memory, and then starts the operating system | Main         | Boot loader           |
| 56 | For most computers the boot strap is stored in-----  | RAM          | ROM                   |
| 57 | A disk that has a boot partition is called a - -----   | Start disk   | Destroyed blocks      |
| 58 | Defective sectors on disks are often known as -----  | Good blocks  | System disk           |
| 59 | Bad blocks are called as -----   | Good Sectors | Defective Sectors     |
| 60 | ROM got ----- file   | boot strap   | Data area             |
|    |  |              |                       |

Class : II B.Sc Maths

Subject Code: 17MMU404B

| Opt3                      | Opt4              | Answer                    |
|---------------------------|-------------------|---------------------------|
| 4                         | 5                 | 2                         |
| Text                      | Executable        | Text                      |
| Text                      | Executable        | Source                    |
| Open file ta              | Manage file table | Open file table           |
| .mdb                      | .in               | .bat                      |
| Text file                 | Word file         | Batch file                |
| Block structure           | Outer structure   | Internal structure        |
| Dynamic access            | Random access     | Sequence access           |
| Dyanamic access           | Random access     | Direct access             |
| Directory                 | Text format       | File                      |
| Seperate directory system | Name,identifier   | Name,Size,Type,identifier |
| Name,Size,Type,identifier | Hard disk         | Seperate directory system |

| Public                             | Private                                    | Abstract                                      |
|------------------------------------|--|---|
| A file to the process calling it   | pointer to the entry in the close file     | pointer to the entry in the open file table   |
| open count                         | Close count                                | open count                                    |
| Name and extension                 | Extension and type                         | Name and extension                            |
| One text document after the other  | One name after the other                   | One record after the other                    |
| Works slow                         | Works normal                               | Works well                                    |
| Disk,Disks                         | Tape,Disk                                  | Disk,Disks                                    |
| The last written position in file  | Middle of the file                         | The beginning of the file                     |
| Pointers to the various blocks     | Pointer to same page                       | Pointers to the various blocks                |
| Swap space                         | Cache                                      | Symbol table                                  |
| Depend on the operating system     | Depend on the file name                    | All files are contained in the same directory |
| All files must have a unique owner | All files must have a different names      | All files must have a unique name             |
| )Both a and b                      | Each user has its different file directory | Both a and b                                  |
| Both MFD and UFD are searched      | Every directory is searched                | Both MFD and UFD are searched                 |

|  |                                    |  |
|--|------------------------------------|--|
| It does not isolate users from one another                               | It isolates users from one another | It isolates users from one another                                       |
| The tree has the root directory  | The tree has no directory          | The tree has the root directory  |
| Linked,Hashed,Indexed  | Contiguous ,Linked                 | Contiguous,Linked,Indexed  |
| All the pointers to scattered  | All the files are blocked          | each file must occupy a set of contiguous block on the disk              |
| All the pointers to scattered  | All the files are blocked          | Each file is a linked list of disk blocks                                |
| All the pointers to scattered blocks are placed together in one location | All the files are blocked          | All the pointers to scattered blocks are placed together in one location |
| Process control block  | File control block                 | Boot loader  |
| Virtual Font System  | Virtual Function System            | Virtual File System  |
| It is not reliable   | It is not valid                    | Linear search to find a file   |
| Costly   | Time taking                        | Finding space for a new file   |

|                             |                               |  |
|-----------------------------|-------------------------------|--|
| Formatting                  | Replacing memory              | Compaction                                   |
| The file cannot be extended | file cannot be opened         | The file cannot be extended                  |
| Macintosh                   | Solaris                       | UNIX   |
| Transaction                 | Method                        | Transaction                                  |
| Completed                   | Finished                      | Committed                                    |
| It is redone                | It is deleted from the memory | It is removed from the log file              |
| both A and B                | overwrite new values          | both A and B                                 |
| Assemblies                  | Cylinders                     | Magnetic disk                                |
| Track                       | Pointer                       | Disk arm                                     |
| Assemblies                  | Cylinders                     | Cylinders                                    |
| Seek time                   | Rotational latency            | Seek time                                    |
| Magnetic damage             | All of these                  | Head crash                                   |
| At the same speed           | Normal speed                  | Slower                                       |
| Both a and b                | Controller at the system side | Controller at the computer end of the bus    |
| Low-level formatting        | Physical formatting           | Low-level formatting<br>,Physical formatting |

|               |                    |                               |
|---------------|--------------------|-------------------------------|
| Trailer       | Main section       | Header ,Data<br>area ,Trailer |
| Sector number | Disk identifier    | Sector number                 |
| Catching      | Logical formatting | partitioning                  |
| Boot strap    | ROM                | Boot strap                    |
| Cache         | Tertiary storage   | ROM                           |
| Boot disk     | Format disk        | System<br>disk,boot disk      |
| Bad blocks    | Semi blocks        | Bad blocks                    |
| boot disks    | boot strap         | Defective<br>Sectors          |
| head data     | random data        | boot strap                    |
|               |                    |                               |

Linux-The Operating System: Linux History, Linux features, Linux distributions, Linux's relationship to Unix, Overview of Linux Architecture, Installation, Start up scripts, system process (an overview), Linux Security, The Ext2 and Ext3 File Systems: General characteristics of the Ext3 File System, File permissions, User Management: Types of users, the powers of Root, Managing users (adding and deleting) : using the command line and GUI Tools.

Resource Management in Linux: File and Directory management, system calls for files process management, Signals, IPC:Pipes, FIFOs, System V IPC, Message Queues, System calls for processes, Memory Management, Library and System calls for Memory.

## Linux:

### What Is Linux?

In the simple language Linux is an operating system (OS). We all are familiar with other operating systems like Microsoft windows, Apple Mac OS, iOS, Google android, etc, just like them linux is also an operating system.

An operating system is a software that enables communication between computer hardware and software. It conveys input to get processed by the processor and brings output to the hardware to display it. This is the basic function of an operating system. Although, it performs many other important tasks, let's not talk about that.

Linux is around us since mid 90s. It can be used from wristwatches to supercomputers. It is everywhere in our phones, laptops, PCs, cars and even in refrigerators. It is very much famous among the developers and normal computer users.

### Structure Of Linux Operating System

An operating system is a collection of software, each designed for a specific function.

Linux OS has following components:

#### 1) Kernel

kernel is the core of the operating system. It establishes communication between devices and software. Moreover, it manages the system resources. Basically it has four responsibilities:

- **device management:** A system has many devices connected to it like CPU, memory device, sound cards, graphic cards, etc. A kernel stores all the data related to all the devices in device driver (without this kernel won't be able to control the devices). Thus kernel knows what a device can do and how to manipulate it to

bring out the best performance. It also manages communication between all the devices. Kernel has certain rules that has to be followed by all the devices.

- **Memory management:** Another function that kernel has to manage is the memory management. Kernel keeps a track of used and unused memory and make sure that processes shouldn't manipulate data of each other using virtual memory address.
- **Process management:** In process management kernel assign enough time and gives priorities to processes before handling CPU to other process. It also deals with security and ownership information.
- **Handling system calls:** Handling system calls means a programmer can write a query or ask the kernel to perform a task.

## 2) System Libraries

System libraries are special programs that helps in accessing the kernel's features. A kernel has to be triggered to perform a task and this triggering is done by the applications. But applications must know how to place a system call because each kernel has a different set of system calls. Programmers have developed standard library of procedures to communicate with kernel. Each operating system supports these standards and then these are transferred to system calls for that operating system.

Most well known system library for Linux is glibc (GNU C library).

## 3) System Tools

Linux OS has a set of utility tools which are usually simple commands. It is a software which GNU project has written and publish under their open source license so that software is freely available to everyone.

With the help of commands you can access your files, edit and manipulate data in your directories or files, change location of files or anything.

## 4) Development Tools

With the above three components your OS is running and working. But to update your system you have additional tools and libraries. These additional tools and libraries are written by the programmers and are called tool chain. A tool chain is a vital development tool used by the developers to produce a working application.

## 5) End User Tools

These end tools make a system unique for a user. End tools are not required for the operating system but are necessary for a user.

Some examples of end tools are graphic design tools, office suites, browsers, multimedia players, etc.

---

## Open Source Operating System

Most OS come in a compiled format means the main source code has run through a program called compiler that translates the source code into a language which is known to the computer.

Modifying this compiled code is really a tough job.

On the other hand, open source is completely different. The source code is included with the compiled version and allows modification by anyone having some knowledge. It gives us freedom to run the



program, freedom to change the code according to our use, freedom to redistribute its copies and freedom to distribute copies which are modified by us.

In short, Linux is an operating system that is "for the people, by the people".

### **Linux History**

---

#### **Evolution of Computer**

In earlier days, computers were as big as houses or parks. So you can imagine how difficult it was to operate them. Moreover, every computer has a different operating system which made it completely worse to operate on them. Every software was designed for a specific purpose and was unable to operate on other computer. It was extremely costly and normal people neither can afford it nor can understand it.

#### **Evolution of Unix**

In 1969, a team of developers of Bell Labs started a project to make a common software for all the computers and named it as 'Unix'. It was simple and elegant, used 'C' language instead of assembly language and its code was recyclable. As it was recyclable, a part of its code now commonly called 'kernel' was used to develop the operating system and other functions and could be used on different systems. Also its source code was open source.

Initially, Unix was only found in large organizations like government, university, or larger financial corporations with mainframes and minicomputers (PC is a microcomputer).

#### **Unix Expansion**

In eighties, many organizations like IBM, HP and dozen other companies started creating their own Unix. It result in a mess of Unix dialects. Then in 1983, Richard Stallman developed GNU project with the goal to make it freely available Unix like operating system and to be used by everyone. But his project failed in gaining popularity. Many other Unix like operating system came into existence but none of them was able to gain popularity.

#### **Evolution of Linux**

In 1991, Linus Torvalds a student at the university of Helsinki, Finland, thought to have a freely available academic version of Unix started writing its own code. Later this project became the Linux kernel. He wrote this program specially for his own PC as he wanted to use Unix 386 Intel computer but couldn't afford it. He did it on MINIX using GNU C compiler. GNU C compiler is still the main choice to compile Linux code but other compilers are also used like Intel C compiler.

He started it just for fun but ended up with such a large project. Firstly he wanted to name it as 'Freax' but later it became 'Linux'.

He published the Linux kernel under his own license and was restricted to use as commercially. Linux uses most of its tools from GNU software and are under GNU copyright. In 1992, he released the kernel under GNU General Public License.

#### **Linux Today**

Today, supercomputers, smart phones, desktop, web servers, tablet, laptops and home appliances like washing machines, DVD players, routers, modems, cars, refrigerators, etc use Linux OS.

### Linux Features

- **Multiuser capability:** Multiple users can access the same system resources like memory, hard disk, etc. But they have to use different terminals to operate.
- **Multitasking:** More than one function can be performed simultaneously by dividing the CPU time intelligently.
- **Portability:** Portability doesn't mean it is smaller in file size or can be carried in pen drives or memory cards. It means that it support different types of hardware.
- **Security:** It provides security in three ways namely authenticating (by assigning password and login ID), authorization (by assigning permission to read, write and execute) and encryption (converts file into an unreadable format).
- **Live CD/USB:** Almost all Linux distros provide live CD/USB so that users can run/try it without installing it.
- **Graphical User Interface (X Window system):** Linux is command line based OS but it can be converted to GUI based by installing packages.
- **Support's customized keyboard:** As it is used worldwide, hence supports different languages keyboards.
- **Application support:** It has its own software repository from where users can download and install many applications.
- **File System:** Provides hierarchical file system in which files and directories are arranged.
- **Open Source:** Linux code is freely available to all and is a community based development project.

### Why Use Linux

Linux is completely different from other operating systems in many ways.

- It is an open source OS which gives a great advantage to the programmers as they can design their own custom operating systems.
- It gives you a lot of option of programs having some different features so you can choose according to your need.
- A global development community look at different ways to enhance its security, hence it is highly secured and robust so you don't need an anti virus to scan it regularly. Companies like Google, Amazon and Facebook use linux in order to protect their servers as it is highly reliable and stable.
- Above all you don't have to pay for software and server licensing to install Linux, its absolutely free and you can install it on as many computers as you want.
- Its completely trouble free operating system and don't have an issue with viruses, malware and slowing down your computer.

### Linux Distributions (Distros)

Other operating systems like Microsoft combine each bit of codes internally and release it as a single package. You have to choose from one of the version they offer.

But Linux is different from them. Different parts of Linux are developed by different organizations.

Different parts include kernel, shell utilities, X server, system environment, graphical programs, etc. If you want you can access the codes of all these parts and assemble them yourself. But its not an easy task seeking a lot of time and all the parts has to be assembled correctly in order to work properly.

From here on distribution (also called as distros) comes into the picture. They assemble all these parts for us and give us a compiled operating system of Linux to install and use.

### **Linux Distributions List**

There are on an average six hundred Linux distributors providing different features. Here, we'll discuss about some of the popular Linux distros today.

#### **1) Ubuntu**

It came into existence in 2004 by Canonical and quickly became popular. Canonical wants Ubuntu to be used as easy graphical Linux desktop without the use of command line. It is the most well known Linux distribution. Ubuntu is a next version of Debian and easy to use for newbies. It comes with a lots of pre-installed apps and easy to use repositories libraries.

Earlier, Ubuntu uses GNOME2 desktop environment but now it has developed its own unity desktop environment. It releases every six months and currently working to expand to run on tablets and smartphones.

#### **2) Linux Mint**

Mint is based on Ubuntu and uses its repository software so some packages are common in both.

Earlier it was an alternative of Ubuntu because media codecs and proprietary software are included in mint but was absent in Ubuntu. But now it has its own popularity and it uses cinnamon and mate desktop instead of Ubuntu's unity desktop environment.

#### **3) Debian**

Debian has its existence since 1993 and releases its versions much slowly then Ubuntu and mint.

This makes it one of the most stable Linux distributor.

Ubuntu is based on Debian and was founded to improve the core bits of Debian more quickly and make it more user friendly. Every release name of Debian is based on the name of the movie Toy Story.

#### **4) Red Hat Enterprise / CentOS**

Red hat is a commercial Linux distributor. There products are red hat enterprise Linux (RHEL) and Fedora which are freely available. RHEL is well tested before release and supported till seven years after the release, whereas, fedora provides faster update and without any support.

Red hat uses trademark law to prevent their software from being redistributed. CentOS is a community project that uses red hat enterprise Linux code but removes all its trademark and make it freely available. In other words, it is a free version of RHEL and provide a stable platform for a long time.

#### **5) Fedora**

It is a project that mainly focuses on free software and provides latest version of software. It doesn't make its own desktop environment but used 'upstream' software. By default it has GNOME3 desktop environment. It is less stable but provides the latest stuff.

---

### **Choosing a Linux Distro** **Distribution**

### **Why To Use**

## KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II B.Sc Maths  
COURSE CODE: 17MMU404B

COURSE NAME: OPERATING SYSTEMS: LINUX  
UNIT: V LINUX  
BATCH-2017-2020

|                    |  |
|--------------------|--|
| Ubuntu             | It works like Mac OS and easy to use.  |
| Linux mint         | It works like windows and should be use by new comers.                             |
| Debian             | It provides stability but not recommended to a new user.                           |
| Fedora             | If you want to use red hat and latest software.                                    |
| Red hat enterprise | To be used commercially.   |
| CentOS             | If you want to use red hat but without its trademark.                              |
| OpenSUSE           | It works same as Fedora but slightly older and more stable.                        |
| Arch Linux         | It is not for the beginners because every package has to be installed by yourself. |

### Unix Vs Linux

Today Linux is in great demand. You can see the use of Linux everywhere. It's dominating on our servers, desktop, smartphones and even used in some electrical devices like refrigerators.

Some people think Unix and Linux as synonyms, but that's not true. Many operating systems were developed to be like Unix but none of them got the popularity as Linux. Linux is the clone of Unix. It has several features similar to Unix, still have some key differences. Before Linux and Windows, computer world was dominated by Unix. Unix is a copyrighted name and IBM AIX, HP-UX and Sun Solaris are only Unix operating system remained till date.

| <u>Difference between Linux and Unix</u> |  |   |
|--|--|---|
| Comparison                               | Linux  | Unix  |
| Definition                               | It is an open-source operating system which is <i>freely available to everyone</i> .                             | It is an operating system which <i>can be only used by its copyrighters</i> .     |
| Examples                                 | It has different distros like Ubuntu, Redhat, Fedora, etc  | IBM AIX, HP-UX and Sun Solaris.   |
| Users                                    | Nowadays, Linux is in great demand. Anyone can use Linux whether a home user, developer or a student.            | It was developed mainly for servers, workstations and mainframes.                 |
| Usage                                    | Linux is used everywhere from servers, PC, smartphones, tablets to mainframes and supercomputers.                | It is used in servers, workstations and PCs.                                      |
| Cost                                     | Linux is freely distributed, downloaded, and distributed through magazines also. And priced distros of Linux are | Unix copyright vendors decide different costs for their respective Unix Operating |

## KARPAGAM ACADEMY OF HIGHER EDUCATION

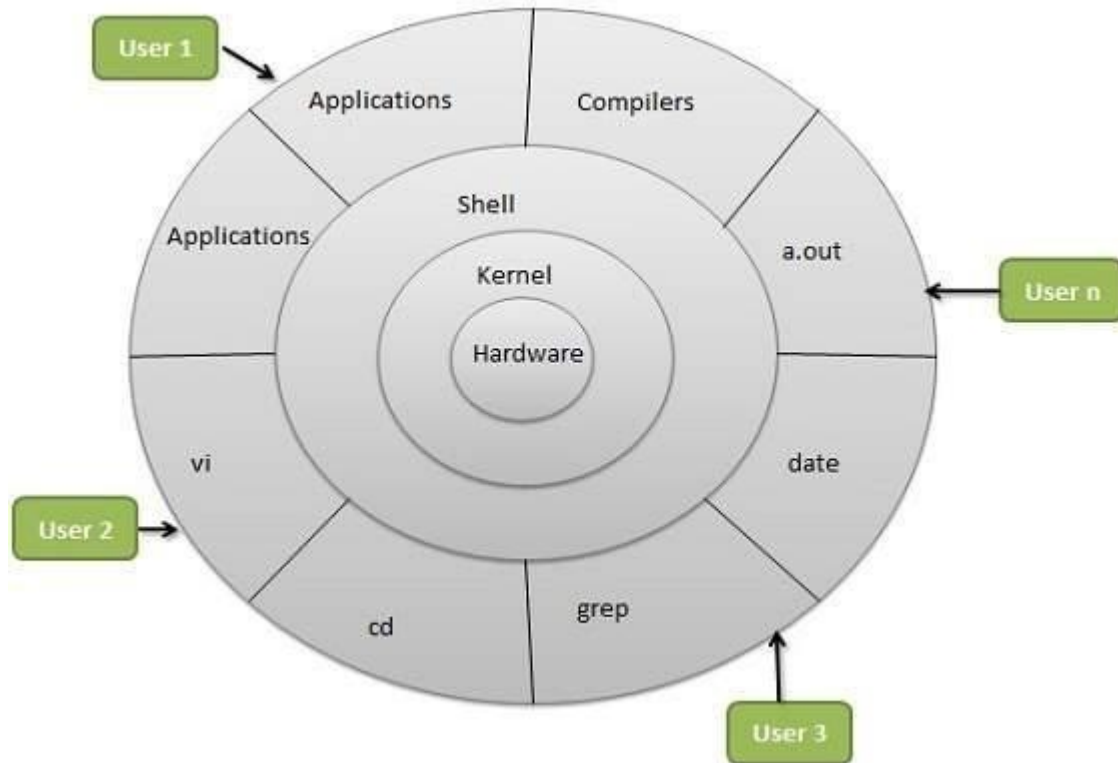
CLASS: II B.Sc Maths  
COURSE CODE: 17MMU404B

COURSE NAME: OPERATING SYSTEMS: LINUX  
UNIT: V LINUX  
BATCH-2017-2020

|                              |   |   |
|------------------------------|---|---|
|                              | also cheaper than Windows.  | systems.  |
| Development                  | As it is open source, it is developed by sharing and collaboration of codes by world-wide developers.   | Unix was developed by AT&T Labs, various commercial vendors and non-profit organizations.                               |
| Manufacturer                 | Linux kernel is developed by the community of developers from different parts of the world. Although the father of Linux, Linus Torvalds oversees things.         | Unix has three distributions IBM AIX, HP-UX and Sun Solaris. Apple also uses Unix to make OSX operating system.         |
| GUI                          | Linux is command based but some distros provide GUI based Linux. Gnome and KDE are mostly used GUI.   | Initially it was command based OS, but later Common Desktop Environment was created. Most Unix distributions use Gnome. |
| Interface                    | The default interface is BASH (Bourne Again SHell). But some distros have developed their own interfaces.   | It originally used Bourne shell. But is also compatible with other GUIs.  |
| File system support          | Linux supports more file system than Unix.  | It also supports file system but lesser than Linux.   |
| Coding                       | Linux is a Unix clone, behaves like Unix but doesn't contain its code.  | Unix contain a completely different coding developed by AT&T Labs.  |
| Operating system             | Linux is just the kernel.   | Unix is a complete package of Operating system.   |
| Security                     | It provides higher security. Linux has about 60-100 viruses listed till date.   | Unix is also highly secured. It has about 85-120 viruses listed till date   |
| Error detection and solution | As Linux is open-source, whenever a user post any kind of threat, developers from all over the world start working on it. And hence, it provides faster solution. | In Unix, users have to wait for some time for the problem to be resolved.   |

### Architecture

The following illustration shows the architecture of a Linux system –



The architecture of a Linux System consists of the following layers –

- **Hardware layer** – Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** – It is the 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. The shell takes commands from the user and executes kernel's functions.
- **Utilities** – Utility programs that provide the user most of the functionalities of an operating systems.

### Introduction to Linux security principles

#### Introduction

Security should be one of the foremost thoughts at all stages of setting up your Linux computer. To implement a good security policy on a machine requires a good knowledge of the fundamentals of Linux as well as some of the applications and protocols that are used.

Security of Linux is a massive subject and there are many complete books on the subject. I couldn't put everything in this one tutorial, but this does give a basic introduction to security and how the techniques, and tools can be used to provide additional security on a Linux computer. Hopefully this will provide sufficient information to be able to investigate other sources of information.

#### Why do we need security?

Although Linux users are must less prone to viruses than some other major operating systems, there are still many security issues facing Linux users and administrators.



One of the most important steps in any task is to identify why you are doing it. Rather than just saying we need to make a system secure you need to consider what is meant by secure, what risks there are associated with any data that's available, what impact your security measures will have on your users. Without first considering any of these factors how else will you know if you've met your goal of making a system secure.

### **Security requirements**

After establishing why security is to be implemented you should consider the aspects of security that are required. The main security requirements are:

**Authorisation** - Only allow those that need access to the data

**Authenticity** - Verifying they are who they say they are

**Privacy / Confidentiality** - Ensure personal information is not being compromised

**Integrity** - Ensuring that the data has not been tampered with

**Non-repudiation** - Confirmation that data is received. The ability to prove it in court

**Availability** - Ensure that the system can perform it's required function

### **Imposed requirements**

Some security requirements are not ones that are directly under your control but are instead imposed upon you. These may be legal requirements (e.g. Data Protection Act 1998), compliance with standards (e.g. ISO 7984-2 International Standards Organisation Security Standard), or corporate policy. If you handle credit card transactions then you may be required to comply with minimum security standards as described by the Payment Card Industry (PCI).

Some of these standards are very vague (e.g. the Data Protection Act just specifies that appropriate security should be in place) whereas some may be more specific (e.g. a corporate policy may insist on a minimum length of passwords etc.).

### **Knowing the enemy**

Before being able to effectively protect a computer system you need to know who it is that is trying to attack your systems and what they are trying to do. I have shown some examples by answering a few questions about those who could potentially attack a computer system.

1. Who wants to?
2. Why are they doing this?
3. What do they try and achieve?
4. How do they do it?

### **Hackers, crackers and phreakers**

These words are commonly used when referring to security attacks, however the meanings are often misinterpreted or understood. I have taken these in order of how easy the term is to explain so as to avoid confusing these together. Note that other people may have different meanings when they use these terms.

**Phreakers** - Also known as Phone Phreakers, this term originates from what could be considered to be the earliest form of attacks against electronic systems. It's earliest for was to bypass the systems used in telephone systems allowing free or reduced price international phone calls. One of the earliest forms of

this was when the American pay phone system used a certain frequency signal to indicate that a coin had been placed in the phone. It was discovered that the frequency of the signal was 2600 Hz, which was also the same frequency emitted from a toy whistle distributed with a popular make of cereals. By blowing the whistle into the phone when a request was made for payment the Phreaker could fool the operating into thinking that money had been deposited in the pay phone.

**Crackers** - These are people that gain unauthorised access to a computer. When people refer to hackers breaking into a computer then they are really referring to crackers.

**Hackers** - Using the traditional meaning of the word Hacker is not meant to imply any kind of illegal or immoral activities. The true meaning is of a computer enthusiast that understands the inner workings of a system and uses that knowledge to "hack" together programs etc. to perform a function. This was different to the traditional techniques or programming that are designed to follow a set structure and procedure to produce a finished piece of software. Due to incorrect use, including by the press, the word hacker has now come to take on two meanings. One is it's original meaning and the other is that of anyone who tries to penetrate a computer (crackers) or those who cause intentional disruption or damage (none-physical) to computer systems.

Throughout this tutorial I will normally refer to the perpetrator as an attacker, regardless of which of these categories she comes under, however where I do refer to a hacker I will normally mean the newer of these meanings.

Whilst some may object to my use of the word hacker, my justification is to turn to the definition held in the Oxford English dictionary which describes the popular use of the language and is considered a definitive guide to the English language:

"Hacker - computer enthusiast, esp. one gaining unauthorised access to files"  
*The Oxford Popular Dictionary, Parragon, 1995*

### **The stereotypes - why be a hacker?**

By understanding the reasons for the attacks gives a basis for what protection can be used to protect the data. I have therefore taken a few examples of reasons for hackers. This includes the stereotypical examples and some that you may not necessarily think about. This is by no means complete, it does however highlight that there are different reasons that someone would want to attack your system.

**Just for fun** - Typically someone in further or higher education that uses the college or universities computer facilities to attack another computer over the Internet. Whilst there are indeed a number of attackers that match this description it is important to recognise that these are not the only type of hackers. This person will typically have limited resources and normally does it, just for fun; or to prove their intelligence etc. However they may be part of a larger group united using the Internet. Whilst many do not intend to commit malicious damage they may discredit your company name, they may cause accidental damage, and may open the door for others.

**Commercial espionage / sabotage** - Whilst espionage normally congers up the image of James Bond fighting a host of bad guys the reality is much less dramatic. There is potentially a risk from competitors wanting to gain a competitive edge. For example if you are bidding for a contract and your competitor is able to find out details of your bid, they could easily undercut you and win the contract. Alternatively by putting your web page out of action, customers could be encouraged to try the competition. This kind of attacker normally has a lot of resources, both financial and in man power, at it's disposal and



has very specific targets. If your organisation is involved in military contracts there may be a real Ernst Stavro Blofeld trying to steal the technology to take over the world.

**Fighting a cause** - Other groups that may wish to attack your company are those that are fighting for a cause or defending a belief. Whilst there are a number of obvious extremist groups such as terrorists or the extremist animal rights groups this could equally apply to less controversial areas where someone has a different opinion.

**Disgruntled employees** - So far I have mentioned attackers external to the organisation, however it is sometimes the case that the greater risk lies from employees within the organisation. These could already have authorised access to a computer, and already be inside the firewall. They could then use that access against the organisation and exploit other holes in the system. Whilst these people can have different motives one of the most obvious is for someone that has been fired, disciplined or who is not satisfied with their current standing in the organisation. Defending against the internal employee can be more challenging as methods need to be found to limit access without preventing others for performing their job. To tighten up security to the point where employees cannot do their job properly is an indirect Denial of Service.

**Unintentional user error** - Whilst normal users may not be trying to cause any damage to the system it's possible that they could cause some accidental damage to data. By limiting a users access user errors can be contained to a reasonable extent. This could be in the form of a programming error as well as incorrectly typing instructions into a program.

### **Linux File Systems: Ext2 vs Ext3 vs Ext4**

by Ramesh Natarajan on May 16, 2011

[Tweet](#)

ext2, ext3 and ext4 are all filesystems created for Linux. This article explains the following:

- High level difference between these filesystems.
- How to create these filesystems.
- How to convert from one filesystem type to another.

### **Ext2**

- Ext2 stands for second extended file system.
- It was introduced in 1993. Developed by Rémy Card.
- This was developed to overcome the limitation of the original ext file system.
- Ext2 does not have journaling feature.
- On flash drives, usb drives, ext2 is recommended, as it doesn't need to do the over head of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext2 file system size can be from 2 TB to 32 TB

### **Ext3**

- Ext3 stands for third extended file system.
- It was introduced in 2001. Developed by Stephen Tweedie.
- Starting from Linux Kernel 2.4.15 ext3 was available.

- The main benefit of ext3 is that it allows journaling.
- Journaling has a dedicated area in the file system, where all the changes are tracked. When the system crashes, the possibility of file system corruption is less because of journaling.
- Maximum individual file size can be from 16 GB to 2 TB
- Overall ext3 file system size can be from 2 TB to 32 TB
- There are three types of journaling available in ext3 file system.
  - Journal – Metadata and content are saved in the journal.
  - Ordered – Only metadata is saved in the journal. Metadata are journaled only after writing the content to disk. This is the default.
  - Writeback – Only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to the disk.
- You can convert a ext2 file system to ext3 file system directly (without backup/restore).

## Ext4

- Ext4 stands for fourth extended file system.
- It was introduced in 2008.
- Starting from Linux Kernel 2.6.19 ext4 was available.
- Supports huge individual file size and overall file system size.
- Maximum individual file size can be from 16 GB to 16 TB
- Overall maximum ext4 file system size is 1 EB (exabyte). 1 EB = 1024 PB (petabyte). 1 PB = 1024 TB (terabyte).
- Directory can contain a maximum of 64,000 subdirectories (as opposed to 32,000 in ext3)
- You can also mount an existing ext3 fs as ext4 fs (without having to upgrade it).
- Several other new features are introduced in ext4: multiblock allocation, delayed allocation, journal checksum, fast fsck, etc. All you need to know is that these new features have improved the performance and reliability of the filesystem when compared to ext3.
- In ext4, you also have the option of turning the journaling feature "off".

## Understanding Linux File Permissions

Although there are already a lot of good security features built into Linux-based systems, one very important potential vulnerability can exist when local access is granted - - that is file permission based issues resulting from a user not assigning the correct permissions to files and directories. So based upon the need for proper permissions, I will go over the ways to assign permissions and show you some examples where modification may be necessary.

### Basic File Permissions

#### Permission Groups

Each file and directory has three user based permission groups:

- **owner** - The Owner permissions apply only to the owner of the file or directory, they will not impact the actions of other users.
- **group** - The Group permissions apply only to the group that has been assigned to the file or directory, they will not effect the actions of other users.
- **all users** - The All Users permissions apply to all other users on the system, this is the permission group that you want to watch the most.

## Permission Types

Each file or directory has three basic permission types:

- **read** - The Read permission refers to a user's capability to read the contents of the file.
- **write** - The Write permissions refer to a user's capability to write or modify a file or directory.
- **execute** - The Execute permission affects a user's capability to execute a file or view the contents of a directory.

## Viewing the Permissions

You can view the permissions by checking the file or directory permissions in your favorite GUI File Manager (which I will not cover here) or by reviewing the output of the `ls -l` command while in the terminal and while working in the directory which contains the file or folder.

The permission in the command line is displayed as: `-rwxrwxrwx 1 owner:group`

### 1. User rights/Permissions

1. The first character that I marked with an underscore is the special permission flag that can vary.
  2. The following set of three characters (rwx) is for the owner permissions.
  3. The second set of three characters (rwx) is for the Group permissions.
  4. The third set of three characters (rwx) is for the All Users permissions.
2. Following that grouping since the integer/number displays the number of hardlinks to the file.
  3. The last piece is the Owner and Group assignment formatted as Owner:Group.

## Modifying the Permissions

When in the command line, the permissions are edited by using the command **chmod**. You can assign the permissions explicitly or by using a binary reference as described below.

### Explicitly Defining Permissions

To explicitly define permissions you will need to reference the Permission Group and Permission Types.

The Permission Groups used are:

**u** - Owner

**g** - Group

**o** - Others

**a** - All users

The potential Assignment Operators are + (plus) and - (minus); these are used to tell the system whether to add or remove the specific permissions.

The Permission Types that are used are:

- **r** - Read
- **w** - Write

- **x** - Execute

So for an example, let's say I have a file named file1 that currently has the permissions set to **\_rw\_rw\_rw**, which means that the owner, group and all users have read and write permission. Now we want to remove the read and write permissions from the all users group.

To make this modification you would invoke the command: **chmod a-rw file1**

To add the permissions above you would invoke the command: **chmod a+rw file1**

As you can see, if you want to grant those permissions you would change the minus character to a plus to add those permissions.

### Using Binary References to Set permissions

Now that you understand the permissions groups and types this one should feel natural. To set the permission using binary references you must first understand that the input is done by entering three integers/numbers.

A sample permission string would be **chmod 640 file1**, which means that the owner has read and write permissions, the group has read permissions, and all other user have no rights to the file.

The first number represents the Owner permission; the second represents the Group permissions; and the last number represents the permissions for all other users. The numbers are a binary representation of the rwx string.

- **r** = 4
- **w** = 2
- **x** = 1

You add the numbers to get the integer/number representing the permissions you wish to set. You will need to include the binary permissions for each of the three permission groups.

So to set a file to permissions on file1 to read **\_rwxr\_\_\_\_\_**, you would enter **chmod 740 file1**.

### **Owners and Groups**

I have made several references to Owners and Groups above, but have not yet told you how to assign or change the Owner and Group assigned to a file or directory.

You use the **chown** command to change owner and group assignments, the syntax is simple **chown owner:group filename**, so to change the owner of file1 to user1 and the group to family you would enter **chown user1:family file1**.

### **Advanced Permissions**

The special permissions flag can be marked with any of the following:

- **\_** - no special permissions
- **d** - directory
- **l** - The file or directory is a symbolic link
- **s** - This indicated the setuid/setgid permissions. This is not set displayed in the special permission part of the permissions display, but is represented as a **s** in the read portion of the owner or group permissions.

- **t** - This indicates the sticky bit permissions. This is not set displayed in the special permission part of the permissions display, but is represented as a **t** in the executable portion of the all users permissions

## Setuid/Setgid Special Permissions

The setuid/setgid permissions are used to tell the system to run an executable as the owner with the owner's permissions.

Be careful using setuid/setgid bits in permissions. If you incorrectly assign permissions to a file owned by root with the setuid/setgid bit set, then you can open your system to intrusion.

You can only assign the setuid/setgid bit by explicitly defining permissions. The character for the setuid/setgid bit is **s**.

So do set the setuid/setgid bit on file2.sh you would issue the command **chmod g+s file2.sh**.

## Sticky Bit Special Permissions

The sticky bit can be very useful in shared environment because when it has been assigned to the permissions on a directory it sets it so only file owner can rename or delete the said file.

You can only assign the sticky bit by explicitly defining permissions. The character for the sticky bit is **t**.

To set the sticky bit on a directory named dir1 you would issue the command **chmod +t dir1**.

## When Permissions Are Important

To some users of Mac- or Windows-based computers you don't think about permissions, but those environments don't focus so aggressively on user based rights on files unless you are in a corporate environment. But now you are running a Linux-based system and permission based security is simplified and can be easily used to restrict access as you please.

So I will show you some documents and folders that you want to focus on and show you how the optimal permissions should be set.

- **home directories**- The users' home directories are important because you do not want other users to be able to view and modify the files in another user's documents or desktop. To remedy this you will want the directory to have the **drwx** (700) permissions, so let's say we want to enforce the correct permissions on the user user1's home directory that can be done by issuing the command **chmod 700 /home/user1**.
- **bootloader configuration files**- If you decide to implement password to boot specific operating systems then you will want to remove read and write permissions from the configuration file from all users but root. To do you can change the permissions of the file to 700.
- **system and daemon configuration files**- It is very important to restrict rights to system and daemon configuration files to restrict users from editing the contents, it may not be advisable to restrict read permissions, but restricting write permissions is a must. In these cases it may be best to modify the rights to 644.
- **firewall scripts** - It may not always be necessary to block all users from reading the firewall file, but it is advisable to restrict the users from writing to the file. In this case the firewall script is run by the root user automatically on boot, so all other users need no rights, so you can assign the 700 permissions.

## Linux User Management

User management includes everything from creating a user to deleting a user on your system. User management can be done in three ways on a Linux system.

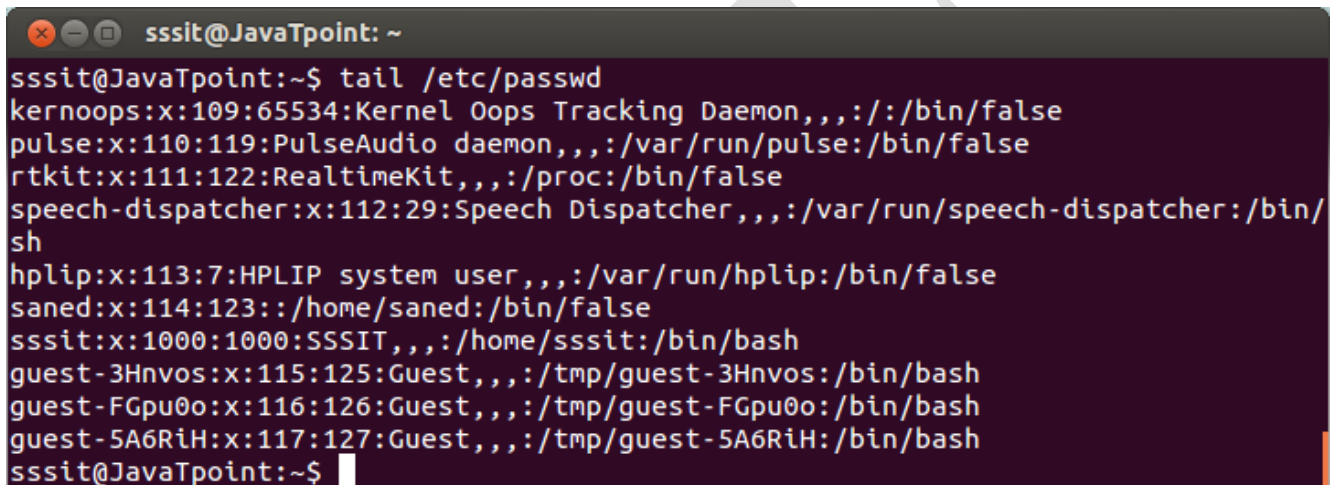
**Graphical tools** are easy and suitable for new users, as it makes sure you'll not run into any trouble.

**Command line tools** includes commands like useradd, userdel, passwd, etc. These are mostly used by the server administrators.

Third and very rare tool is to **edit the local configuration files** directly using vi.

1. /etc/passwd

The local user database in Linux is /etc/passwd directory.



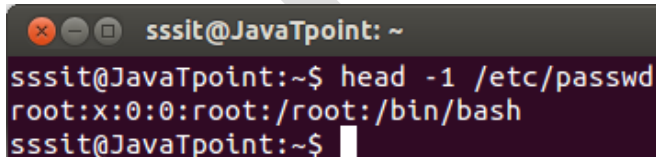
```
sssit@JavaTpoint: ~  
sssit@JavaTpoint:~$ tail /etc/passwd  
kernoops:x:109:65534:Kernel Oops Tracking Daemon,,,:/bin/false  
pulse:x:110:119:PulseAudio daemon,,,:/var/run/pulse:/bin/false  
rtkit:x:111:122:RealtimeKit,,,:/proc:/bin/false  
speech-dispatcher:x:112:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh  
hplip:x:113:7:HPLIP system user,,,:/var/run/hplip:/bin/false  
saned:x:114:123::/home/saned:/bin/false  
sssit:x:1000:1000:SSSIT,,,:/home/sssit:/bin/bash  
guest-3Hnvos:x:115:125:Guest,,,:/tmp/guest-3Hnvos:/bin/bash  
guest-FGpu0o:x:116:126:Guest,,,:/tmp/guest-FGpu0o:/bin/bash  
guest-5A6RiH:x:117:127:Guest,,,:/tmp/guest-5A6RiH:/bin/bash  
sssit@JavaTpoint:~$
```

Look at the above snapshot, it has seven columns separated by a colon. Starting from the left columns denotes username, an x, user id, primary group id, a description, name of home directory and a login shell.

---

### root

The root user is the superuser and have all the powers for creating a user, deleting a user and can even login with the other user's account. The root user always has userid 0.



```
sssit@JavaTpoint: ~  
sssit@JavaTpoint:~$ head -1 /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
sssit@JavaTpoint:~$
```

---

### useradd

With useradd commands you can add a user.

### **Syntax:**



1. `useradd -m -d /home/<userName> -c "<userName>" <userName>`

## Example:

1. `useradd -m -d /home/xyz -c "xyz" xyz`

```
root@JavaTpoint: ~  
root@JavaTpoint:~# useradd -m -d /home/xyz -c "xyz" xyz  
root@JavaTpoint:~# tail -2 /etc/passwd  
akki:x:1003:1003::/home/akki:/bin/sh  
xyz:x:1004:1004:xyz:/home/xyz:/bin/sh  
root@JavaTpoint:~#
```

Look at the above snapshot, we have created a user **xyz** along with creating a home directory (-m), setting the name of home directory (-d), and a description (-c).

The 'xyz' received **userid** as 1004 and **primary group id** as 1004.

## /etc/default/useradd

File /etc/default/useradd contains some user default options. The command **useradd -D** can be used to display this file.

## Syntax:

1. `useradd -D`

```
root@JavaTpoint: ~  
root@JavaTpoint:~# useradd -D  
GROUP=100  
HOME=/home  
INACTIVE=-1  
EXPIRE=  
SHELL=/bin/sh  
SKEL=/etc/skel  
CREATE_MAIL_SPOOL=no  
root@JavaTpoint:~#
```

## userdel

To delete a user account **userdel** command is used.

## Syntax:

1. `userdel -r <userName>`

```
root@JavaTpoint: ~  
root@JavaTpoint:~# tail -1 /etc/passwd  
xyz:x:1004:1004:xyz:/home/xyz:/bin/sh  
root@JavaTpoint:~#  
root@JavaTpoint:~# userdel -r xyz  
root@JavaTpoint:~# tail -1 /etc/passwd  
akki:x:1003:1003::/home/akki:/bin/sh  
root@JavaTpoint:~#
```

**Example:**

1. userdel -r xyz

Look at the above snapshot, first we have shown the xyz user account with 'tail' command. To delete it, command "**userdel -r xyz**" is passed.

To recheck, again 'tail' command is passed and as you can see no xyz user account is displayed.

Hence, it is deleted.

**usermod**

The command usermod is used to modify the properties of an existing user.

**Syntax:**

1. usermod -c <newName> <oldName>

**Example:**

1. usermod -c 'jhonny' john

```
root@JavaTpoint: ~  
root@JavaTpoint:~# tail -1 /etc/passwd  
john:x:1002:1002:john taylor:/home/john:/bin/sh  
root@JavaTpoint:~#  
root@JavaTpoint:~# usermod -c 'jhonny' john  
root@JavaTpoint:~# tail -1 /etc/passwd  
john:x:1002:1002:johnny:/home/john:/bin/sh  
root@JavaTpoint:~#
```

Look at the above snapshot, user name **john** is replaced by the new user name **jhonny**

**/etc/skel/**



The /etc/skel/ contains some hidden files which have profile settings and default values for applications. Hence, it serves as a default home directory and user profile. While using useradd -m option, the /etc/skel/ is copied to the newly created directory.

```
root@JavaTpoint: ~  
root@JavaTpoint:~# ls -la /etc/skel  
total 40  
drwxr-xr-x  2 root root  4096 Aug 18  2012 .  
drwxr-xr-x 128 root root 12288 Jul  2 17:50 ..  
-rw-r--r--  1 root root   220 Apr  3  2012 .bash_logout  
-rw-r--r--  1 root root  3486 Apr  3  2012 .bashrc  
-rw-r--r--  1 root root  8445 Apr 16  2012 examples.desktop  
-rw-r--r--  1 root root   675 Apr  3  2012 .profile  
root@JavaTpoint:~#
```

Look at the above snapshot, files of /etc/skel/ is listed.

## Deleting Home Directories

By using **userdel -r** option, you can delete home directory along with user account.

### **Syntax:**

1. userdel -r <userName>

### **Example:**

1. userdel -r john

```
root@JavaTpoint: ~  
root@JavaTpoint:~# ls -ld /home/john  
drwxr-xr-x 2 john john 4096 Jul  2 17:49 /home/john  
root@JavaTpoint:~# userdel -r john  
root@JavaTpoint:~# ls -ld /home/john  
ls: cannot access /home/john: No such file or directory  
root@JavaTpoint:~#
```

Look at the above snapshot, both home directory as well as user account john is deleted.

## Login Shell

The /etc/passwd file also tells about the login shell for the user.

```
root@JavaTpoint: ~  
root@JavaTpoint:~# tail -2 /etc/passwd  
jtp:x:1001:1001:,,,:/home/jtp:/bin/ksh  
guest-on3hSB:x:118:128:Guest,,,:/tmp/guest-on3hSB:/bin/bash  
root@JavaTpoint:~#
```

Look at the above snapshot, user guest will log in with **/bin/bash** shell and user jtp will log in with **/bin/ksh** shell.

You can change the shell mode with **usermod** command for a user.

#### Syntax:

1. `usermod -s <newShell> <userName>`

#### Example:

1. `usermod -s /bin/bash jtp`

```
root@JavaTpoint: ~  
root@JavaTpoint:~# usermod -s /bin/bash jtp  
root@JavaTpoint:~# tail -2 /etc/passwd  
jtp:x:1001:1001:,,,:/home/jtp:/bin/bash  
guest-on3hSB:x:118:128:Guest,,,:/tmp/guest-on3hSB:/bin/bash  
root@JavaTpoint:~#
```

Look at the above snapshot, shell of jtp is changed to **/bin/bash** from **/bin/ksh**.

#### chsh

Users can change their login shell with **chsh** command.

Both the command **chsh** and **chsh -s** will work to change the shell.

#### Syntax:

1. `chsh`

```
sssit@JavaTpoint: ~  
sssit@JavaTpoint:~$ chsh  
Password:  
Changing the login shell for sssit  
Enter the new value, or press ENTER for the default  
Login Shell [/bin/sh]: /bin/bash  
sssit@JavaTpoint:~$
```

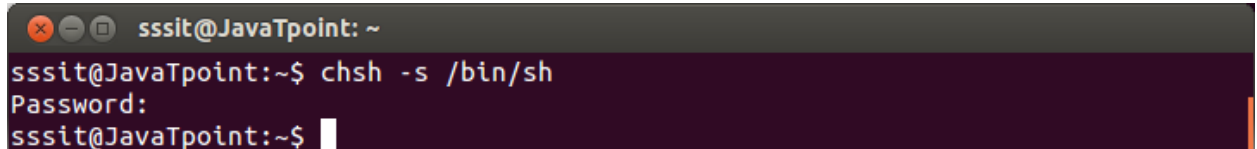
Look at the above snapshot, command **chsh** has changed the sssit login shell from **/bin/sh** to **/bin/bash**.

### Syntax:

1. `chsh -s <newShell>`

### Example:

1. `chsh -s /bin/sh`



```
sssit@JavaTpoint: ~  
sssit@JavaTpoint:~$ chsh -s /bin/sh  
Password:  
sssit@JavaTpoint:~$
```

## Resource Management

### Starting a Process

When you start a process (run a command), there are two ways you can run it –

- Foreground Processes
- Background Processes

### Foreground Processes

By default, every process that you start runs in the foreground. It gets its input from the keyboard and sends its output to the screen.

You can see this happen with the **ls** command. If you wish to list all the files in your current directory, you can use the following command –

```
$ls ch*.doc
```

This would display all the files, the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc ch010.doc ch02.doc ch03-2.doc  
ch04-1.doc ch040.doc ch05.doc ch06-2.doc  
ch01-2.doc ch02-1.doc
```

The process runs in the foreground, the output is directed to my screen, and if the **ls** command wants any input (which it does not), it waits for it from the keyboard.

While a program is running in the foreground and is time-consuming, no other commands can be run (start any other processes) because the prompt would not be available until the program finishes processing and comes out.

### Background Processes

A background process runs without being connected to your keyboard. If the background process requires any keyboard input, it waits.

The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!

The simplest way to start a background process is to add an ampersand (&) at the end of the command.

```
$ls ch*.doc &
```

This displays all those files the names of which start with **ch** and end with **.doc** –

```
ch01-1.doc ch010.doc ch02.doc ch03-2.doc
ch04-1.doc ch040.doc ch05.doc ch06-2.doc
ch01-2.doc ch02-1.doc
```

Here, if the **ls** command wants any input (which it does not), it goes into a stop state until we move it into the foreground and give it the data from the keyboard.

That first line contains information about the background process - the job number and the process ID. You need to know the job number to manipulate it between the background and the foreground.

Press the Enter key and you will see the following –

```
[1] + Done      ls ch*.doc &
$
```

The first line tells you that the **ls** command background process finishes successfully. The second is a prompt for another command.

### Listing Running Processes

It is easy to see your own processes by running the **ps** (process status) command as follows –

```
$ps
PID  TTY  TIME  CMD
18358 ttty3 00:00:00 sh
18361 ttty3 00:01:31 abiword
18789 ttty3 00:00:00 ps
```

One of the most commonly used flags for **ps** is the **-f** ( f for full) option, which provides more information as shown in the following example –

```
$ps -f
UID    PID PPID C STIME TTY TIME CMD
amrood 6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood 6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood  3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood  6892 3662 4 10:51:50 pts/6 0:00 ps -f
```

Here is the description of all the fields displayed by **ps -f** command –

| S.No. | Column & Description   |
|-------|--|
| 1     | <b>UID</b><br>User ID that this process belongs to (the person running it) |
| 2     | <b>PID</b><br>Process ID   |

- 3     **PPID**  
Parent process ID (the ID of the process that started it)
- 4     **C**  
CPU utilization of process
- 5     **STIME**  
Process start time
- 6     **TTY**  
Terminal type associated with the process
- 7     **TIME**  
CPU time taken by the process
- 8     **CMD**  
The command that started this process

There are other options which can be used along with **ps** command –

| S.No. | Option & Description   |
|-------|--|
| 1     | <b>-a</b><br>Shows information about all users                   |
| 2     | <b>-x</b><br>Shows information about processes without terminals |
| 3     | <b>-u</b><br>Shows additional information like -f option         |
| 4     | <b>-e</b><br>Displays extended information                       |

### **Stopping Processes**

Ending a process can be done in several different ways. Often, from a console-based command, sending a CTRL + C keystroke (the default interrupt character) will exit the command. This works when the process is running in the foreground mode.

If a process is running in the background, you should get its Job ID using the **ps** command. After that, you can use the **kill** command to kill the process as follows –

```
$ps -f
UID      PID PPID C STIME TTY TIME CMD
amrood 6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood 6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood 3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood 6892 3662 4 10:51:50 pts/6 0:00 ps -f
$kill 6738
Terminated
```

Here, the **kill** command terminates the **first\_one** process. If a process ignores a regular kill command, you can use **kill -9** followed by the process ID as follows –

```
$kill -9 6738
Terminated
```

### Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

Most of the commands that you run have the shell as their parent. Check the **ps -f** example where this command listed both the process ID and the parent process ID.

### Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

### Daemon Processes

Daemons are system-related background processes that often run with the permissions of root and services requests from other processes.

A daemon has no controlling terminal. It cannot open **/dev/tty**. If you do a "**ps -ef**" and look at the **tty** field, all daemons will have a **?** for the **tty**.

To be precise, a daemon is a process that runs in the background, usually waiting for something to happen that it is capable of working with. For example, a printer daemon waiting for print commands.

If you have a program that calls for lengthy processing, then it's worth to make it a daemon and run it in the background.

### The top Command

The **top** command is a very useful tool for quickly showing processes sorted by various criteria.

It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

Here is the simple syntax to run top command and to see the statistics of CPU utilization by different processes –

\$top

### **Job ID Versus Process ID**

Background and suspended processes are usually manipulated via **job number (job ID)**. This number is different from the process ID and is used because it is shorter.

In addition, a job can consist of multiple processes running in a series or at the same time, in parallel. Using the job ID is easier than tracking individual processes.

### **Linux Memory Management – Virtual Memory and Demand Paging**

Memory management is one of the most complex activity done by Linux kernel. It has various concepts/issues associated with it.

### **Virtual Memory**

The concept of virtual memory is one of the very powerful aspects of memory management. Since the initial era of computers the need of memory more than the existing physical memory has been felt. Over the years, many solutions were used to overcome this issue and the most successful of them has been the concept of virtual memory.

Virtual memory makes your system appear as if it has more memory than it actually has. This may sound interesting and may prompt one to ask how is this possible. So, let's understand the concept.

- To start, we must first understand that virtual memory is a layer of memory addresses that map to physical addresses.
- In virtual memory model, when a processor executes a program instruction, it reads the instruction from virtual memory and executes it.
- But before executing the instruction, it first converts the virtual memory address into physical address.
- This conversion is done based on the mapping of virtual to physical addresses that is done based on the mapping information contained in the page tables (that are maintained by OS).

The virtual and physical memory is divided into fixed length chunks known as pages. In this paged model, a virtual address can be divided into two parts :

- An offset (Lowest 12 bits)
- A virtual page frame number (rest of the bits)

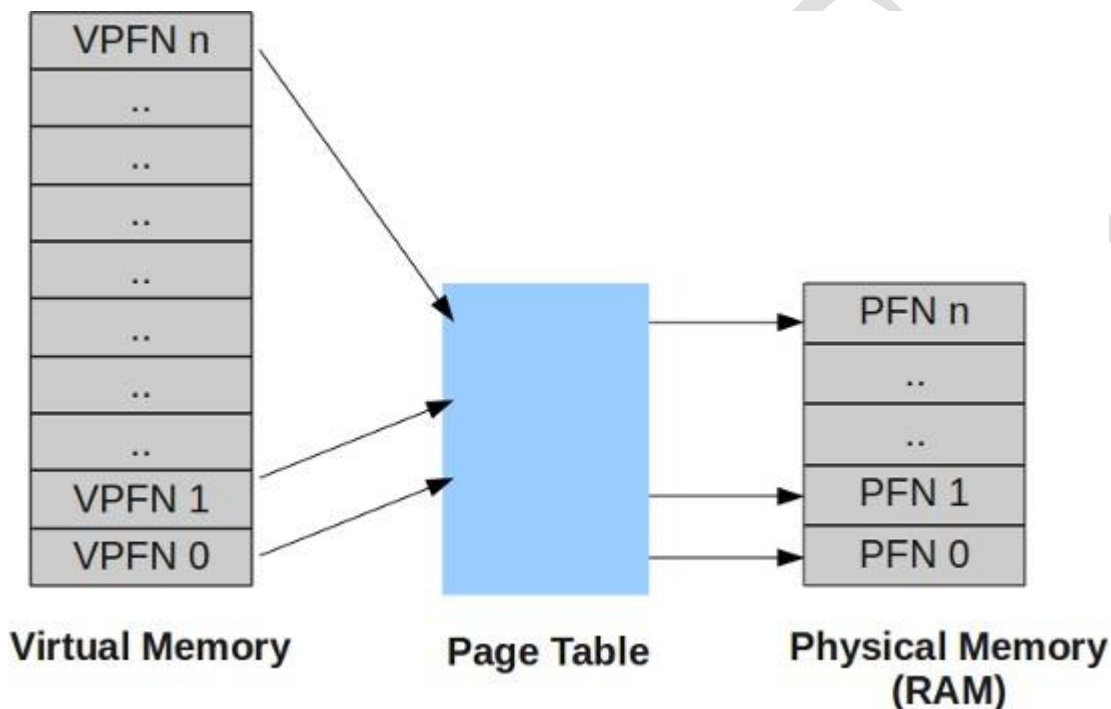
When ever the processor encounters a virtual address, it extracts the virtual page frame number out of it. Then it translates this virtual page frame number into a physical page frame number and the offset parts helps it to go to the exact address in the physical page. This translation of addresses is done through the page tables.

Theoretically we can consider a page table to contain the following information :

- A flag that describes whether the entry is valid or not
- The physical page frame number as described by this entry
- Access information regarding the page (like read-only, read-write etc)

A page table is accessed through virtual page frame number using it as offset for entries in the page table. For example, a virtual page frame number of „2“ points to the entry „1“ in the page table (the ~~of~~ numbers begin with „0“).

In the image below, VPFN stands for Virtual page frame number, and PFN indicates the physical page frame number.



It may happen that a processor goes to a processes page table entry with a virtual page frame number and finds the entry as invalid. In this case it is the processor's responsibility to pass the control to kernel and ask it to fix the problem. Different processors pass the control in different ways but this phenomenon is known as a „page fault“. But if the entry was valid then processor takes the physical page frame number, multiplies with the size of the page to get the base address of the physical page and then adds the offset to get to the exact physical address.

So now we understand that through the concept of virtual memory, each process thinks that it has all range of virtual address at its disposal and hence this concepts make the system appear as if it has more physical memory than actually available.

### **Demand Paging**

In the previous sectioned we learned that if the processor goes to the processes page table with a virtual page frame number for which no entry was present in the table then two cases arise.

1. Either the process has tried to access an invalid memory address



2. The physical page corresponding to the virtual address was not loaded into physical memory

Out of the two cases above, the case 1 is the case where the process tries to memory address which it is not allowed. In this case a page fault is generated and the kernel terminates the process.

While in case „2“, as already explained, the physical page corresponding to the virtual address is not yet loaded into physical memory. In this case also a page fault is generated and the kernel then tries to bring the required memory page into physical memory from hard disk.

Since this operation of bringing a page from hard disk into physical memory is time consuming so by this time a context switch between processes happens and some other process is brought into execution. Meanwhile the page of the earlier process is brought into physical memory and the page tables are updated and then this process is brought back into execution again from the same instruction that caused the „page fault“.

\*\*\*\*\*

**POSSIBLE QUESTIONS**

**PART –B**

**(Each Question carries 2 Marks)**

1. List the feature of Linux.
2. What is kernel?
3. What is a signal?
4. Define : Pipes
5. What is system call?
6. How do you add and delete users?
7. Define: Library
8. What is security?
9. How do you install Linux?
10. What is start up scripts?

**PART –C**

**(Each Question carries 6 Marks)**

1. Explain about OS Protection and Security.
2. Explain in detail about Policy mechanism..
3. Discuss about Internal access authorization.
4. Describe the Security in Operating System.
5. Explain in detail about Authentication.
6. Discuss about the protection for Linux Files and Directories
7. Describe the process of protecting of Operating System.
8. Discuss about the Configuration of User Authentication
9. Describe the process of Threats in Operating System.
10. Discuss about Policy versus Mechanism

Semester : III

Subject : Operating Systems: Linux

| S.NO | Question   | Opt1            |
|------|--|-----------------|
| UNIT |  |                 |
| 1    | _____developed the Linux Operating System.   | Linus Torvalds  |
| 2    | The first Linux kernel run only on _____Microprocessors.                                 | 80586 Intel     |
| 3    | Linux looks and feels much like _____Operating System.                                   | Windows         |
| 4    | Linux is an _____Operating System.   | Close Source    |
| 5    | The Linux programming interface adheres to the _____semantics                            | SVR4 UNIX       |
| 6    | Linux is designed to be compliant with the _____documents                                | Purge           |
| 7    | Linux supports _____Operating System.  | Multitasking    |
| 8    | _____reads input command and translates it to the operating system.                      | Vinix Shell     |
| 9    | Linux was originally programmed with a _____compatible filesystem                        | Minimum         |
| 10   | VFS stands for _____File System.   | Vertical        |
| 11   | The _____is a low-level systems software whose main role is to manage hardware resources | Windows GUI     |
| 12   | _____is a boot loader for Linux.   | LILO            |
| 13   | CLI stands for _____Interface.   | Close Line      |
| 14   | Linux Virtual File System is designed around _____principles.                            | Object based    |
| 15   | A _____is an interpreter for the information from the terminal device.                   | line discipline |

|    |  |                   |
|----|--|-------------------|
| 16 | Linux's security model is closely related to _____security mechanism.    | Relational        |
| 17 | PAM stands for _____Authentication Module.                               | Pure              |
| 18 | The standard on-disk file system used by Linux is called _____.          | ex2fs             |
| 19 | Linux implements dynamic linking through a spacial _____library .        | Locker            |
| 20 | The _____service allocates memory on demand in Linux.                    | calloc            |
| 21 | Linux memory management uses a _____algorithm to manage physical memory. | Bubble sort       |
| 22 | Linux kernel supports symmetric _____hardware.                           | multi processing  |
| 23 | A _____ mechanism allows device drivers to reserve hardware resources.   | module management |
| 24 | The _____ management allows modules to be loaded into memory.            | module            |
| 25 | The Linux kernel is distributed under General _____.                     | Public License.   |
| 26 | _____is a popular distribution of commercial Linux.                      | White Hat         |
| 27 | The _____is an array of pointers to kernel file structures.              | memory table      |
| 28 | _____are a potential problem in kernel routine.                          | Page faults       |
| 29 | Linux interrupts are a problem only if _____exist.                       | Cross Sections    |
| 30 | Linux uses _____algorithm for time-sharing process.                      | Sorting           |
| 31 | Windows 2000 is _____bit operating system.                               | 34                |
| 32 | Windows 2000 is _____ operating system.                                  | multi kernel      |
| 33 | Windows 2000 provides better _____support.                               | OS                |

|    |  |           |
|----|--|-----------|
| 34 | Windows 2000 supports upto _____ of RAM.                             | 4 gb      |
| 35 | DLL stands _____ Link Library.                                       | Databse   |
| 36 | HAL stands for Hardware _____ Layer.                                 | Absolute  |
| 37 | LPC stands for _____ Procedure Call.                                 | Lower     |
| 38 | Windows 2000 supports many languages because it has _____ API.       | NLS       |
| 39 | The kernel of Windows 2000 is _____ oriented.                        | Procedure |
| 40 | A _____ object acts as gate to control the number of threads.        | semaphore |
| 41 | _____ objects control dispatching and synchronization.               | Databse   |
| 42 | APC stands for _____ Procedure Call.                                 | Absolute  |
| 43 | _____ can occur when a thread enters the ready or wait state.        | Symmetric |
| 44 | The _____ dispatcher creates an exception record.                    | Extreme   |
| 45 | The kernel uses an _____ dispatch table to bind each interrupt.      | Interrupt |
| 46 | DPC stands for _____ Procedure Call.                                 | Database  |
| 47 | The job of the _____ manager is to supervise the use of all objects. | Object    |
| 48 | _____ are a standardized interface to all kinds of objects.          | Hampers   |
| 49 | The design of _____ Memory manager supports a paging mechanism.      | Virtue    |
| 50 | _____ is responsible for cache management and network drivers.       | Network   |
| 51 | VDM stands for Virtual _____ Machine.                                | Databse   |

|    |   |              |
|----|---|--------------|
| 52 | _____routines call the appropriate Win32 subroutines, converting 16-bit addresses into 32-bit ones                              | Skeleton     |
| 53 | _____FAT file system has solved the size and fragmentation problems   | 16-bit       |
| 54 | NTFS uses _____cluster numbers as disk addresses.   | Logical      |
| 55 | In MS-DOS and UNIX, each directory uses a datastructure called a _____ Tree   | AVL          |
| 56 | NTFS divides the file's data into _____ units which are blocks of 16 contiguous clusters  | Single       |
| 57 | The _____is a protocol provided by Windows 2000 to communicate between Windows 2000 server and other clients over the Internet. | TCP/IP       |
| 58 | The Common Object Model is a mechanism for interprocess communication that was developed for                                    | IBM          |
| 59 | _____protocol was introduced by IBM in 1985 as networking protocol for up to 254 machines                                       | WebDAV       |
| 60 | Winsock is a _____ layer interface that is largely compatible with UNIX sockets   | Applications |

**Higher Education****Department of Mathematics**

Class : II B.Sc Maths

Subject Code:

17MMU404B

| Opt2            | Opt3                       | Opt4              | Answer                     |
|-----------------|----------------------------|-------------------|----------------------------|
| V               |                            |                   |                            |
| Lion Torvalds   | Lamp Torvalds              | Lin Torvals       | Linus Torvalds             |
| 80386 Intel     | 80186 AMD                  | 8085 Intel        | 80386 Intel                |
| Windows         | UNIX                       | Apple Mac         | UNIX                       |
| Open Source     | Program Source             | None of the above | Open Source                |
| SOR4 UNIX       | SXR4 UNIX                  | SYR4 UNIX         | SVR4 UNIX                  |
| Prefix          | Postfix                    | POSIX             | POSIX                      |
| Multi User      | Multiuser and Multitasking | Single User       | Multiuser and Multitasking |
| Unix Hell       | Unix Shell                 | Cinix Shell       | Unix Shell                 |
| Maximum         | Minix                      | DOS               | Minix                      |
| Virtual         | vector                     | VINIX             | Virtual                    |
| Linux Kernel    | Minix System               | Vinix System      | Linux Kernel               |
| LIXO            | LIFO                       | LIMO              | LILO                       |
| Command Line    | Common Line                | Cinix Line        | Command Line               |
| Object Oriented | Structured                 | none              | Object oriented            |
| Line discipline | Character discipline       | normalization     | line discipline            |

|                   |                     |                   |                     |
|-------------------|---------------------|-------------------|---------------------|
| Transactional     | Object Oriented     | Text              | Text                |
| Purge             | Pipeline            | Pluggable         | Pluggable           |
| expfs             | ex67fp              | extfp             | ex2fs               |
| linker            | liquid              | Lava              | linker              |
| kmalloc           | dalloc              | qalloc            | kmalloc             |
| Sort-heap         | buddy-heap          | heap              | buddy-heap          |
| single processing | Image processing    | Data processing   | multi processing    |
| driver management | conflict-resolution | memory management | conflict-resolution |
| memory            | processor           | file              | module              |
| Private Lincense  | Corporate License   | Domain License    | Public License.     |
| Black Hat         | Grey Hat            | Red Hat           | Red Hat             |
| file table        | module table        | schedule table    | file table          |
| Processor faults  | Pivot faults        | Pipeline faults   | Page faults         |
| Middle sections   | Critical Sections   | Dummy Table       | Critical sections   |
| Swapping          | Searching           | credit-based      | credit-based        |
| 35                | 32                  | 64                | 32                  |
| multi tasking     | multi threading     | multi OS          | multi tasking       |
| CUI               | networking          | platform          | networking          |



|             |              |                  |              |
|-------------|--------------|------------------|--------------|
| 8 gb        | 16 gb        | 64 gb            | 64 gb        |
| Distributed | Dynamic      | DoS              | Dynamic      |
| Abstraction | Ambient      | Apple Mac        | Abstraction  |
| Local       | Less         | Little           | Local        |
| NPI         | NIT          | NQT              | NLS          |
| Object      | Function     | Structure        | Object       |
| Super       | Semaphore    | Mutual Exclusion | semaphore    |
| Distributed | Dispatcher   | Scheduler        | Dispatcher   |
| Abstraction | Asynchronous | Synchronous      | Asynchronous |
| Scheduling  | Sampling     | Superphore       | Scheduling   |
| Exception   | Error        | extfp            | Exception    |
| Interior    | Inter        | Inner            | Interrupt    |
| Deferred    | Dispatcher   | Distributed      | Deferred     |
| memory      | processor    | Printer          | Object       |
| Huckles     | Handles      | Heaps            | Handles      |
| Virtual     | vector       | Vento            | Virtual      |
| Driver      | I/O Manager  | Interrupt        | I/O Manager  |
| DOS         | Drive        | Distributed      | DOS          |

|              |                |              |             |
|--------------|----------------|--------------|-------------|
| Kernel       | Stub           | GID function | Stub        |
| 8-bit        | 64-bit         | 32-bit       | 32-bit      |
| Physical     | Virtual        | Static       | Logical     |
| BinarySearch | P              | B+           | B+          |
| Compression  | Two            | Compaction   | Compression |
| DLC          | AppleTalk      | PPTP         | PPTP        |
| UNIX         | LINUX          | Windows      | Windows     |
| NetBIOS      | Novell Netware | DHCP         | NetBIOS     |
| Network      | Session        | Data-link    | Session     |

Reg.No \_\_\_\_\_

[17MMU404B]

**KARPAGAM ACADEMY OF HIGHER EDUCATION**  
(Deemed to be University Established Under Section 3 of UGC Act 1956)

COIMBATORE – 641 021

**B.SC DEGREE EXAMINATION**

(For the candidates admitted in 2017 onwards)

**Fourth Semester**

**FIRST INTERNAL TEST**

**OPERATING SYSTEMS: LINUX**

**Maximum : 50Marks**

**Time : 2 Hours**  
**DATE & SESSION:**

**PART-A (20 \* 1 = 20Marks )**

**Answer ALL the questions**

1. Process states are \_\_\_\_\_  
a) Submit, Ready, Block    b) Submit, Run, Ready    **c) Ready, Run, Block**    d) Run, Stop
2. What is a shell?  
a) It is a hardware component    **b) It is a command interpreter**  
c) It is a part in compiler    d) It is a tool in CPU scheduling
3. The only state transition initiated by the user process itself is  
a) Dispatch    b) Timer run out    **c) Block**    d) Suspend
4. Which scheduling is effective in time sharing environments?  
a) FIFO scheduling    **b) RR scheduling**    c) SJF scheduling    d) SRT scheduling
5. \_\_\_\_\_ were the first computers used to tackle many commercial & scientific application.  
**a) Mainframe computer system**    b) Mainframe computer service  
c) Multiframe computing system    d) Multiframe cloud service
6. \_\_\_\_\_ contains the address of an instruction to be fetched from memory.  
a) Program counter (PC)    **b) Instruction register (IR)**  
c) Control registers    d) Status registers
7. In Banker's Algorithm \_\_\_\_\_ conditions are allowed.  
**a) mutual-exclusion**    b) Time-sharing    c) race condition  
d) cooperating processes
8. Systems designed for graceful degradation are also called \_\_\_\_\_.  
a) graceful degradation    b) Economy of scale  
**c) fault tolerant**    d) Increased throughput
9. OS chooses one process whenever CPU is \_\_\_\_\_.  
a) Waiting    **b) Idle**    c) Busy    d) Ready
10. On systems with multiple command interpreters to choose from, the interpreters are

Known as \_\_\_\_\_

- a) GUI                      **b) Shells**                      c) Signal                      d) Command

11. Block state transition is

- a) Running Ready                      b) Ready Running  
**c) Running Blocked**                      d) Blocked Ready

12. Process is said to be blocked if

- a) It is the first process in ready list  
**b) It is waiting for some event to happen**  
c) It currently has CPU  
d) It could use a CPU if available

13. A Semaphore is a \_\_\_\_\_ variable whose value can be accessed and altered only by the operations P & V

- a) Predefined                      **b) Protected**  
c) User defined                      d) Built-in

14. Process states are \_\_\_\_\_

- a) Submit, Ready, Block                      b) Submit, Run, Ready  
**c) Ready, Run, Block**                      d) Ready, Submit, Block

15. Which access a process should have to modify any information in a segment?

- a) Write**                      b) Read                      c) Execute                      d) Append

16. Critical section can also be called

- a) Critical region                      **b) Mutual exclusion**  
c) Parallelism                      d) Critical part

17. How many child processes a process can have?

- a) 0                      b) 1                      c) 2                      d) Any number

18. \_\_\_\_\_ allows the OS to locate all key information about a process

- a) FAT                      **b) PCB**                      c) Process Table                      d) None

19. Ready list contains

- a) List of processes blocked itself to do some operation  
**b) List of processes ready to run**  
c) List of Submitted processes  
d) List of Running processes

20. The only state transition initiated by the user process itself is

- a) Dispatch                      b) Timer run out                      **c) Block**                      d) Segment

### Part – B(3X 2= 6 Marks)

#### Answer ALL Questions

## 21. Define Segmentation.

**Segmentation** is one of the most common ways to achieve memory protection. In a computer system using segmentation, an instruction operand that refers to a memory location includes a value that identifies a segment and an offset within that segment.

## 22. What is Process Control?

**Process Control Block (PCB**, also called Task Controlling Block, Entry of the **Process Table**, Task Struct, or Switchframe) is a data structure in the operating system kernel containing the information needed to manage the scheduling of a particular **process**.

## 23. What are all the basic conditions for Deadlock?

A **deadlock situation** on a resource can arise if and only if **all** of the following **conditions** hold simultaneously in a system: Mutual exclusion: At least one resource must be held in a non-shareable mode. Otherwise, the processes would not be prevented from using the resource when necessary.

**Part – C(3X 8= 24 Marks)**  
**Answer ALL Questions**

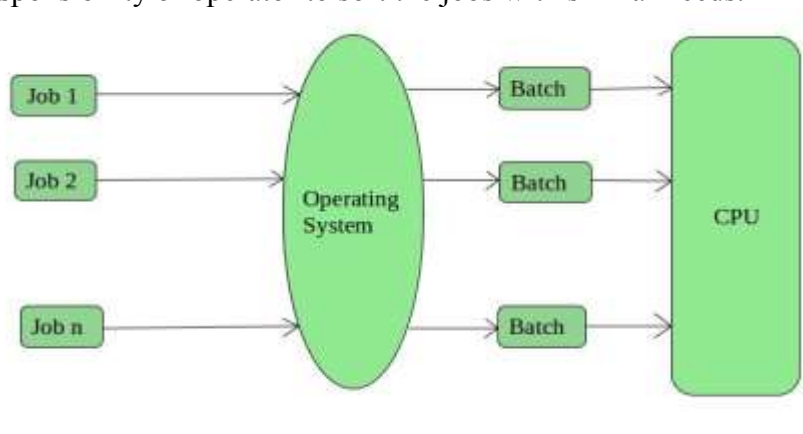
### 24.a. Explain the Types of Operating System.

An Operating System performs all the basic tasks like managing file, process, and memory. Thus operating system acts as manager of all the resources, i.e. **resource manager**. Thus operating system becomes an interface between user and machine.

**Types of Operating Systems:** Some of the widely used operating systems are as follows-

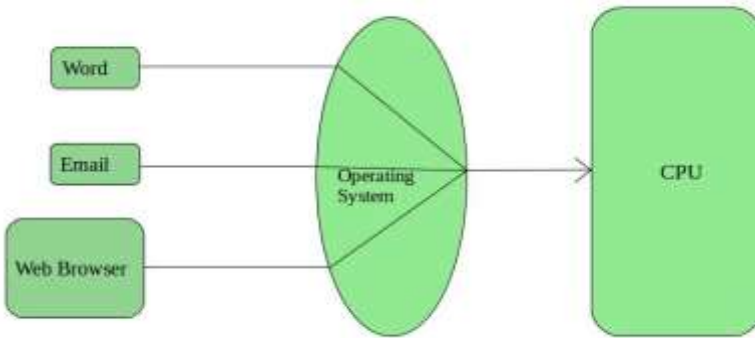
#### 1. Batch Operating System –

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



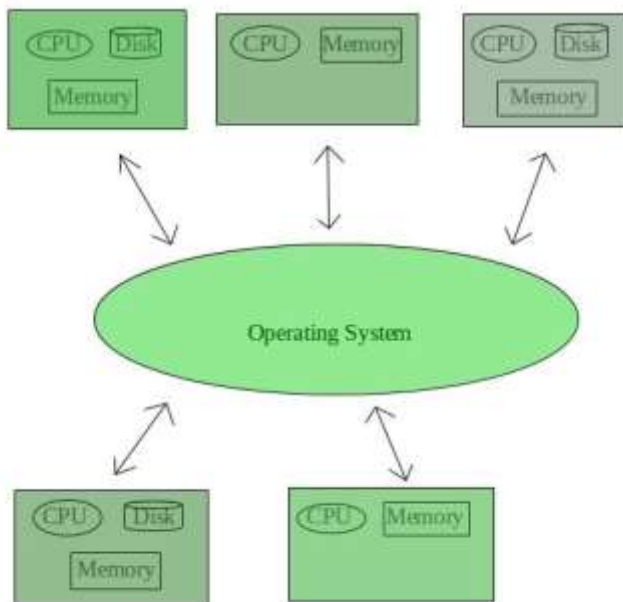
#### 2. Time-Sharing Operating Systems –

Each task has given some time to execute, so that all the tasks work smoothly. Each user gets time of CPU as they use single system. These systems are also known as Multitasking Systems. The task can be from single user or from different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to next task.



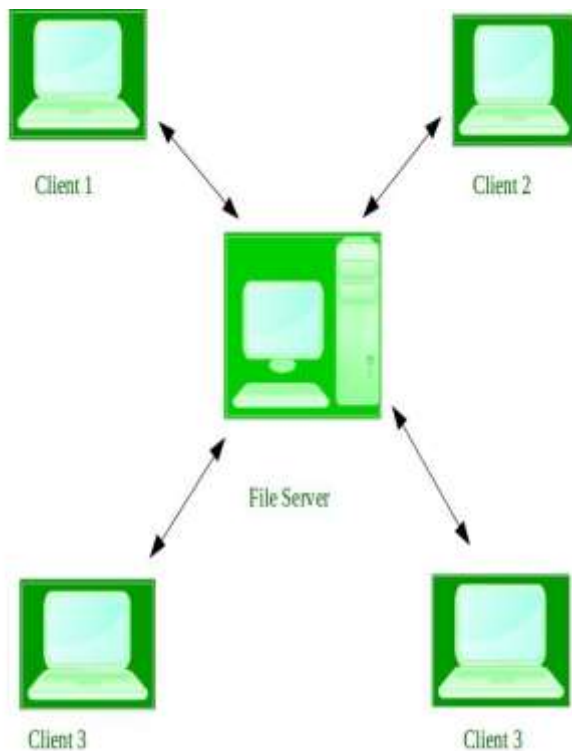
### 3. Distributed Operating System –

These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all-over the world and, that too, with a great pace. Various autonomous interconnected computers communicate each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred as **loosely coupled systems** or distributed systems. These systems processors differ in sizes and functions. The major benefit of working with these types of operating system is that it is always possible that one user can access the files or software which are not actually present on his system but on some other system connected within this network i.e., remote access is enabled within the devices connected in that network.



### 4. Network Operating System –

These systems runs on a server and provides the capability to manage data, users, groups, security, applications, and other networking functions. These type of operating systems allows shared access of files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections etc. and that's why these computers are popularly known as **tightly coupled systems**.



## 5. Real-Time Operating System –

These types of OSs serve the real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

**Real-time systems** are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots etc.

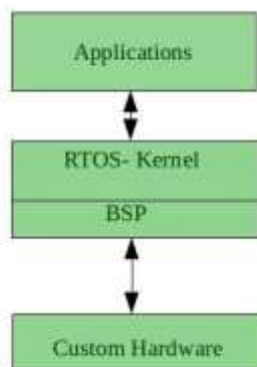
**Two types of Real-Time Operating System which are as follows:**

- **Hard Real-Time Systems:**

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

- **Soft Real-Time Systems:**

These OSs are for applications where the time-constraint is less strict.



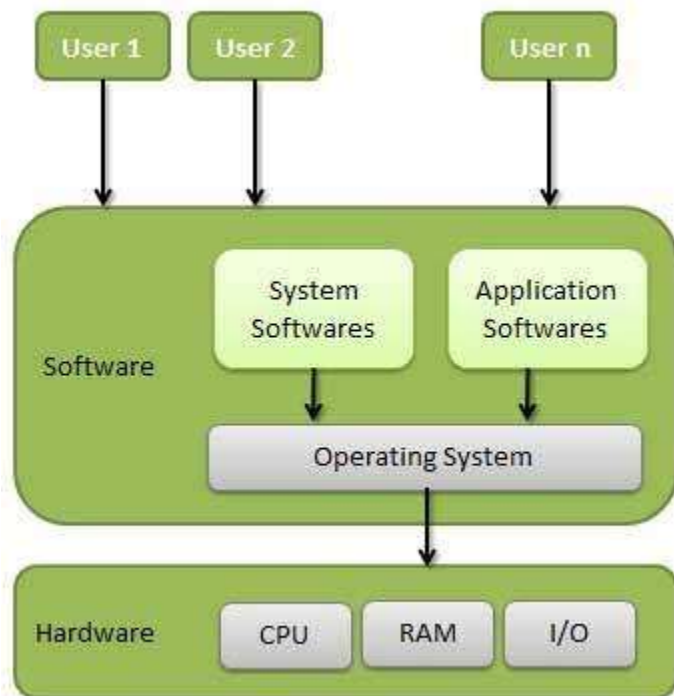
## b. Explain Basic OS Functions.

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

### Section 1.01 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 accessed directly by the CPU. For a program to be executed, it must be in the main memory. An Operating System does the following activities for memory management –

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

### Processor Management

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

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

### Device Management

An Operating System manages device communication via their respective drivers. It 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.

An 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 an Operating System performs –

- **Security** – By means of password and similar other techniques, it prevents 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 softwares and users** – Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

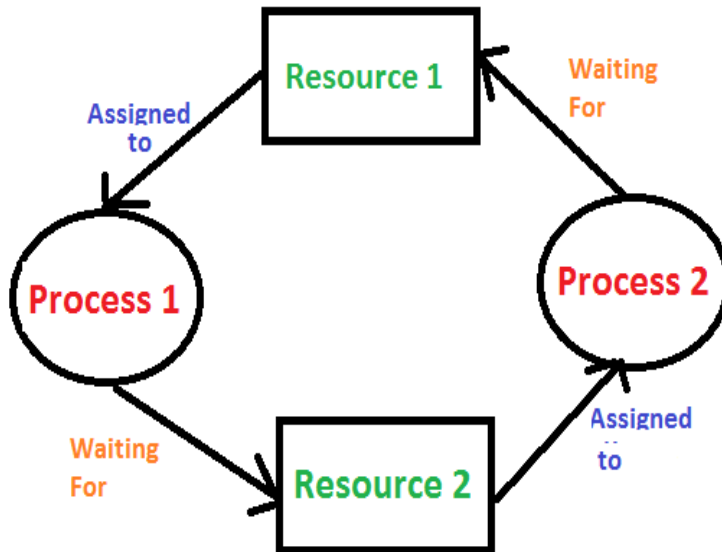
## 25.a. Explain about Deadlock and its process

A process in operating systems uses different resources and uses resources in following way.

- 1) Requests a resource
- 2) Use the resource
- 2) Releases the resource

**Deadlock** is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Consider an example when two trains are coming toward each other on same track and there is only one track, none of the trains can move once they are in front of each other. Similar situation occurs in operating systems when there are two or more processes hold some resources and wait for resources held by other(s).



**Deadlock can arise if following four conditions hold simultaneously (Necessary Conditions)**

**Mutual Exclusion:** One or more than one resource are non-sharable (Only one process can use at a time)

**Hold and Wait:** A process is holding at least one resource and waiting for resources.

**No Preemption:** A resource cannot be taken from a process unless the process releases the resource.

**Circular Wait:** A set of processes are waiting for each other in circular form.

### Methods for handling deadlock

There are three ways to handle deadlock

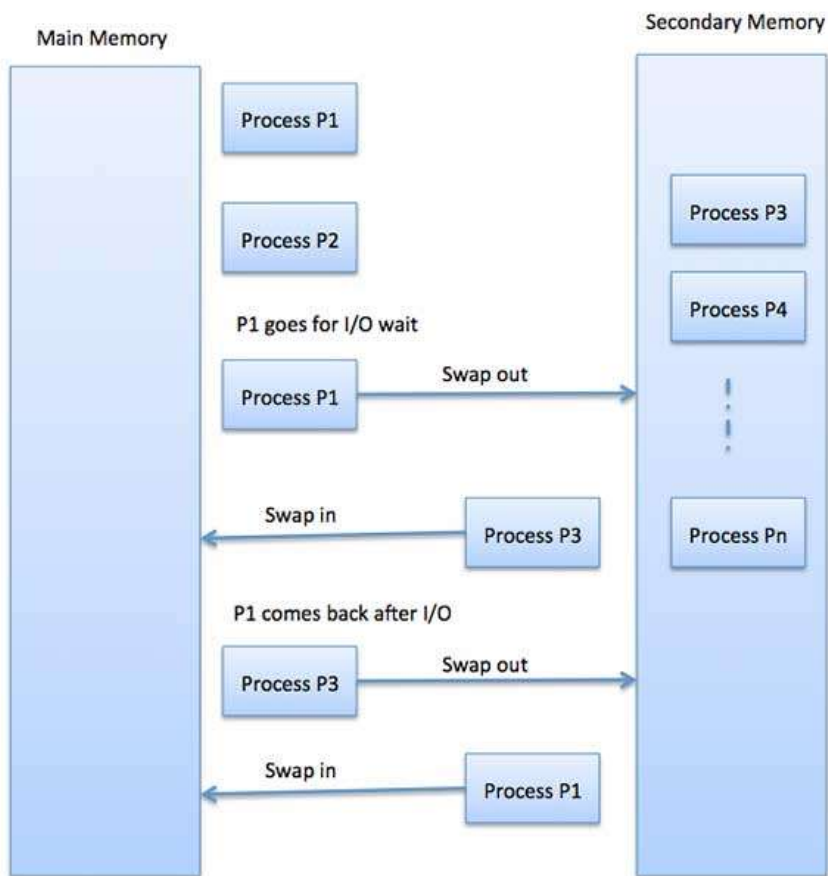
- 1) Deadlock prevention or avoidance: The idea is to not let the system into deadlock state.
- 2) Deadlock detection and recovery: Let deadlock occur, then do preemption to handle it once occurred.
- 3) Ignore the problem all together: If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

### b. Discuss about Swapping of two process.

## Swapping

Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction.**



The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

2048KB / 1024KB per second  
= 2 seconds  
= 2000 milliseconds

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

## 26.a. Explain in detail about Multiprogramming Systems

### Multiprogramming

In a multiprogramming system there are one or more programs loaded in main memory which are ready to execute. Only one program at a time is able to get the CPU for executing its instructions (i.e., there is at most one process running on the system) while all the others are waiting their turn. The main idea of multiprogramming is to maximize the use of CPU time. Indeed, suppose the currently running process is performing an I/O task (which, by definition, does not need the CPU to be accomplished). Then, the OS may interrupt that process and give the control to one of the other in-main-memory programs that are ready to execute (i.e. *process context switching*). In this way, no CPU time is wasted by the system waiting for the I/O task to be completed, and a running process keeps executing until either it voluntarily releases the CPU or when it blocks for an I/O operation. Therefore, the ultimate goal of multiprogramming is to keep the CPU busy as long as there are processes ready to execute. Note that in order for such a system to function properly, the OS must be able to load multiple programs into separate areas of the main memory and provide the required protection to avoid the chance of one process being modified by another one. Other problems that need to be addressed when having multiple programs in memory is *fragmentation* as programs enter or leave the main memory. Another issue that needs to be handled as well is that large programs may not fit at once in memory which can be solved by using *pagination* and *virtual memory*. Please, refer to [this article](#) for more details on that. Finally, note that if there are N ready processes and all of those are highly CPU-bound (i.e., they mostly execute CPU tasks and none or very few I/O operations), in the very worst case one program might wait all the other N-1 ones to complete before executing..

### b. Discuss about Memory Allocation Techniques.

Memory Allocation

Main memory usually has two partitions –

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

Operating system uses the following memory allocation mechanism.

| S.N. | Memory Allocation & Description   |
|------|---|
| 1    | <p><b>Single-partition allocation</b></p> <p>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.</p> |
| 2    | <p><b>Multiple-partition allocation</b></p> <p>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.</p>   |

Reg.No -----

[17MMU404B]

**Karpagam Academy Of Higher Education**  
(Deemed to be University Established Under Section 3 of UGC Act 1956)  
**Coimbatore – 64 021**  
**B.SC Degree Examination**  
(For the candidates admitted from 2017 onwards)  
Fourth Semester  
**Second Internal Exam**  
**OPERATING SYSTEMS: LINUX**

**Time: 2 Hours**

**Date & Session: 08.03.19 & AN**

**Maximum: 50 Marks**

**Class: II B.SC (Maths)**

**Part – A (20 X 1 = 20 Marks)**

**Answer ALL Questions**

1. \_\_\_\_\_ is a code that locates library routine.  
a) stub                      b) dll                      c) recursive routine                      d) exe file
2. \_\_\_\_\_ can be used to manage large memory requirement for a process  
a) overlays                      b) swapping                      c) roll in and out                      d) libraries
3. \_\_\_\_\_ error is raised in memory  
a) addressing                      b) swapping                      c) dynamic                      d) index
4. Set of \_\_\_\_\_ are scattered throughout the memory  
a) holes                      b) gaps                      c) free space                      d) words
5. \_\_\_\_\_ can be internal and external  
a) fragmentation                      b) merging                      c) grouping                      d) fixing
6. \_\_\_\_\_ is used to divide a process into fixed size chunks  
a) paging                      b) segmentation                      c) sp                      d) swapping
7. In paging physical memory is divided into \_\_\_\_\_  
a) frames                      b) pages                      c) segments                      d) bytes
8. In paging virtual memory is divided into \_\_\_\_\_  
a) frames                      b) pages                      c) segments                      d) bytes
9. \_\_\_\_\_ is first of virtual address in paging  
a) page number                      b) segment number                      c) frame number                      d) offset
10. \_\_\_\_\_ is second part of virtual address in paging  
a) page number                      b) segment number                      c) frame number                      d) offset
11. Page mapping entries are found in \_\_\_\_\_  
a) page table                      b) segment table                      c) hash table                      d) pointing table
12. \_\_\_\_\_ is the basis of multiprogrammed operating system.  
a) page table                      b) segment table                      c) hash table                      d) pointing table

- a) RR scheduling                      b) Self Scheduling                      c) CPU scheduling                      d) throughput
13. A \_\_\_\_\_ is executed until it must wait, typically for the completion of some i/o request
- a) reverse                      b) deadlock avoidance                      c) deadlock                      d) process
14. \_\_\_\_\_ is a fundamental operating system function.
- a) RR                      b) CPU                      c) Scheduling                      d) nonpreemptive
15. Process execution begins with a \_\_\_\_\_
- a) CPU burst                      b) RR scheduling                      c) SJF scheduling                      d) SRT scheduling
16. The operating system must select one of the processes in the ready queue to be executed by the \_\_\_\_\_
- a) nonpreemptive                      b) short term scheduler                      c) long term scheduler                      d) low level
17. When scheduling takes place only under circumstances 1 and 4 called \_\_\_\_\_
- a) variable class                      b) real time class                      c) priority class                      d) nonpreemptive class
18. Another component involved in the CPU scheduling function is the \_\_\_\_\_
- a) central edge                      b) dispatcher                      c) claim edge                      d) graph edge
19. One measure of work is the number of processes completed per time unit called \_\_\_\_\_
- a) throughput                      b) variable class                      c) real time class                      d) priority class
20. Which of the following is the simplest scheduling discipline?
- a) FCFS scheduling                      b) RR scheduling                      c) SJF scheduling                      d) SRT scheduling

#### PART-B

#### ANSWER THE FOLLOWING

21. Define Semaphore.
22. What is Process Scheduling?
23. What is Paging?

#### PART-C

#### ANSWER THE FOLLOWING

24. a. Explain the Round Robin Scheduling Algorithm with an example  
(OR)  
b. Explain the Shortest Job First (SJF) Scheduling Algorithm with an example
25. a. Explain the concept of Physical address space in detail.  
(OR)  
b. Explain in detail about Segmentation.
26. a. Explain about Virtual address space.  
(OR)  
b. Discuss about Paging in detail.



**Karpagam Academy Of Higher Education**  
(Deemed to be University Established Under Section 3 of UGC Act 1956)  
**Coimbatore – 64 021**  
**B.SC Degree Examination**  
(For the candidates admitted from 2017 onwards)  
Fourth Semester  
**Second Internal Exam**  
**OPERATING SYSTEMS: LINUX**

**Time: 2 Hours**

**Maximum: 50 Marks**

**Date & Session: 08.03.19 & AN**

**Class: II B.SC (Maths)**

**Part – A (20 X 1 = 20 Marks)**  
**Answer ALL Questions**

1. \_\_\_\_\_ is a code that locates library routine.  
a) stub                      b) **dll**                      c) recursive routine                      d) exe file
2. \_\_\_\_\_ can be used to manage large memory requirement for a process  
a) overlays                      b) **swapping**                      c) roll in and out                      d) libraries
3. \_\_\_\_\_ error is raised in memory  
a) **addressing**                      b) swapping                      c) dynamic                      d) index
4. Set of \_\_\_\_\_ are scattered throughout the memory  
a) holes                      b) gaps                      c) **free space**                      d) words
5. \_\_\_\_\_ can be internal and external  
a) **fragmentation**                      b) merging                      c) grouping                      d) fixing
6. \_\_\_\_\_ is used to divide a process into fixed size chunks  
a) **paging**                      b) segmentation                      c) sp                      d) swapping
7. In paging physical memory is divided into \_\_\_\_\_  
a) **frames**                      b) pages                      c) segments                      d) bytes
8. In paging virtual memory is divided into \_\_\_\_\_  
a) frames                      b) pages                      c) **segments**                      d) bytes
9. \_\_\_\_\_ is first of virtual address in paging  
a) **page number**                      b) segment number                      c) frame number                      d) offset
10. \_\_\_\_\_ is second part of virtual address in paging  
a) page number                      b) segment number                      c) **frame number**                      d) offset
11. Page mapping entries are found in \_\_\_\_\_  
a) page table                      b) **segment table**                      c) hash table                      d) pointing table
12. \_\_\_\_\_ is the basis of multiprogrammed operating system.  
a) RR scheduling                      b) Self Scheduling                      c) **CPU scheduling**                      d) throughput
13. A \_\_\_\_\_ is executed until it must wait, typically for the completion of some i/o request  
a) reverse                      b) deadlock avoidance                      c) **deadlock**                      d) process
14. \_\_\_\_\_ is a fundamental operating system function.

- a) RR                      b) CPU                      c) Scheduling                      d) nonpreemptive
15. Process execution begins with a \_\_\_\_\_  
a) CPU burst                      b) RR scheduling                      c) **SJF scheduling**                      d) SRT scheduling
16. The operating system must select one of the processes in the ready queue to be executed by the \_\_\_\_\_  
a) nonpreemptive                      b) **short term scheduler**                      c) long term scheduler                      d) low level
17. When scheduling takes place only under circumstances 1 and 4 called \_\_\_\_\_  
a) variable class                      b) real time class                      c) **priority class**                      d) nonpreemptive
18. Another component involved in the CPU scheduling function is the \_\_\_\_\_  
a) central edge                      b) dispatcher                      c) claim edge                      d) **graph edge**
19. One measure of work is the number of processes completed per time unit called \_\_\_\_\_  
a) throughput                      b) variable class                      c) **real time class**                      d) priority class
20. Which of the following is the simplest scheduling discipline?  
a) **FCFS scheduling**                      b) RR scheduling                      c) SJF scheduling                      d) SRT scheduling

## PART-B

### ANSWER THE FOLLOWING

#### 21. Define Semaphore.

Semaphore is simply a variable. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.

#### 22. What is Process Scheduling?

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 systems**.

#### 23. What is Paging?

**Paging** is a method of writing data to, and reading it from, secondary storage for use in primary storage, also known as main **memory**. ... In a **memory** management system that takes advantage of **paging**, the **OS** reads data from secondary storage in blocks called pages, all of which have identical size.

## PART-C

### ANSWER THE FOLLOWING

#### 24. a. Explain the Round Robin Scheduling Algorithm with an example

Round Robin is a **CPU scheduling algorithm** where each process is assigned a fixed time slot in a cyclic way.

- It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.
- One of the most commonly used technique in CPU scheduling as a core.
- It is preemptive as processes are assigned CPU only for a fixed slice of time at most.
- The disadvantage of it is more overhead of context switching.

#### Illustration:

##### Example 1

Assume there are 5 processes with process ID and burst time given below

| PID | Burst Time |
|-----|------------|
| P1  | 6          |
| P2  | 5          |
| P3  | 2          |
| P4  | 3          |
| P5  | 7          |

- Time quantum: 2
- Assume that all process arrives at 0.

Now, we will calculate average waiting time for these processes to complete.

### Solution –

We can represent execution of above processes using GANTT chart as shown below –

#### Gantt Chart:

|    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|
| P1 | P2 | P3 | P4 | P5 | P1 | P2 | P4 | P5 | P1 | P2 | P5 |    |
| 0  | 2  | 4  | 6  | 8  | 10 | 12 | 14 | 15 | 17 | 19 | 20 | 23 |

Round Robin Example Gantt Chart – 1

#### Explanation:

- First p1 process is picked from the ready queue and executes for 2 per unit time (time slice = 2).

If arrival time is not available, it behaves like FCFS with time slice.

- After P2 is executed for 2 per unit time, P3 is picked up from the ready queue. Since P3 burst time is 2 so it will finish the process execution at once.

- Like P1 & P2 process execution, P4 and p5 will execute 2 time slices and then again it will start from P1 same as above.

Waiting time = Turn Around Time – Burst Time

$$P1 = 19 - 6 = 13$$

$$P2 = 20 - 5 = 15$$

$$P3 = 6 - 2 = 4$$

$$P4 = 15 - 3 = 12$$

$$P5 = 23 - 7 = 16$$

$$\text{Average waiting time} = (13+15+4+12+16) / 5 = 12$$

### b. Explain the Shortest Job First (SJF) Scheduling Algorithm with an example

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

- Shortest Job first has the advantage of having minimum average waiting time among all scheduling algorithms.

- It is a Greedy Algorithm.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of aging.
- It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

#### Algorithm:

1- Sort all the processes in increasing order according to burst time.

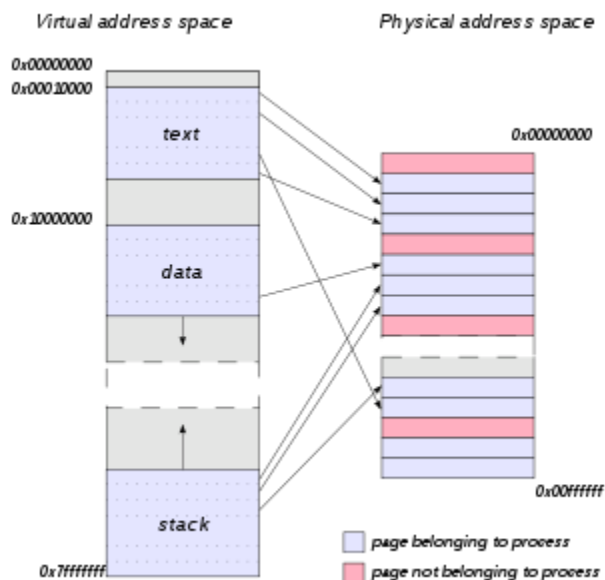
2- Then simply, apply **FCFS**.

#### How to compute below times in SJF using a program?

1. Completion Time: Time at which process completes its execution.
2. Turn Around Time: Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. Waiting Time(W.T): Time Difference between turn around time and burst time. Waiting Time = Turn Around Time – Burst Time

#### 25.a. Explain the concept of Physical address space in detail.

**Physical Address** identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by MMU before they are used. The term Physical Address Space is used for all physical addresses corresponding to the logical addresses in a Logical address space.



#### b. Explain in detail about Segmentation.

In Operating Systems, Segmentation is a memory management technique in which, the memory is divided into the variable size parts. Each part is known as segment which can be allocated to a process.

The details about each segment are stored in a table called as segment table. Segment table is stored in one (or many) of the segments.

Segment table contains mainly two information about segment:

1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

### Why Segmentation is required?

Till now, we were using Paging as our main memory management technique. Paging is more close to Operating system rather than the User. It divides all the process into the form of pages regardless of the fact that a process can have some relative parts of functions which needs to be loaded in the same page.

Operating system doesn't care about the User's view of the process. It may divide the same function into different pages and those pages may or may not be loaded at the same time into the memory. It decreases the efficiency of the system.

It is better to have segmentation which divides the process into the segments. Each segment contain same type of functions such as main function can be included in one segment and the library functions can be included in the other segment,

### Translation of Logical address into physical address by segment table

CPU generates a logical address which contains two parts:

1. Segment Number
2. Offset

The Segment number is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.

In the case of valid address, the base address of the segment is added to the offset to get the physical address of actual word in the main memory.

### Advantages of Segmentation

1. No internal fragmentation
2. Average Segment Size is larger than the actual page size.
3. Less overhead
4. It is easier to relocate segments than entire address space.
5. The segment table is of lesser size as compare to the page table in paging.

### Disadvantages

1. It can have external fragmentation.
2. it is difficult to allocate contiguous memory to variable sized partition.

### 3. Costly memory management algorithms.

#### 26.a. Explain about Virtual address space.

A virtual address in memory is a pointer or marker for a memory space that an operating system allows a process to use. The virtual address points to a location in primary storage that a process can use independently of other processes.

In devices with memory management, a virtual address is different from a physical memory address. In such devices, the memory management unit (MMU) is responsible for memory management including the translation of virtual addresses into physical addresses.

With virtual addresses, the memory management system is able to allocate huge amounts of memory to individual processes. The system lets every process assume it has all available memory to itself, when, in fact, the operating system is juggling memory between processes as needed.

Virtual address is also used in the context of virtual systems. As with virtual memory addresses, newer systems replace physical memory drive destinations with virtual memory systems, where hardware is partitioned into different and more sophisticated types of storage.

#### b. Discuss about Paging in detail.

Paging is a method of writing [data](#) to, and reading it from, [secondary storage](#) for use in [primary storage](#), also known as main memory. Paging plays a role in memory management for a computer's [OS](#) (operating system).

In a memory management system that takes advantage of paging, the OS reads data from secondary storage in blocks called [pages](#), all of which have identical size. The physical region of memory containing a single page is called a frame. When paging is used, a frame does not have to comprise a single physically contiguous region in secondary storage. This approach offers an advantage over earlier memory management methods, because it facilitates more efficient and faster use of storage.

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous.

- Logical Address or Virtual Address (represented in bits): An address generated by the CPU
- Logical Address Space or Virtual Address Space( represented in words or bytes): The set of all logical addresses generated by a program
- Physical Address (represented in bits): An address actually available on memory unit
- Physical Address Space (represented in words or bytes): The set of all physical addresses corresponding to the logical addresses

#### Example:

- If Logical Address = 31 bit, then Logical Address Space =  $2^{31}$  words = 2 G words (1 G =  $2^{30}$ )
- If Logical Address Space = 128 M words =  $2^7 * 2^{20}$  words, then Logical Address =  $\log_2 2^{27} = 27$  bits
- If Physical Address = 22 bit, then Physical Address Space =  $2^{22}$  words = 4 M words (1 M =  $2^{20}$ )
- If Physical Address Space = 16 M words =  $2^4 * 2^{20}$  words, then Physical Address =  $\log_2 2^{24} = 24$  bits

The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.

- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.

- The Logical address Space is also splitted into fixed-size blocks, called **pages**.
- Page Size = Frame Size

Let us consider an example:

- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words
- Page size = frame size = 1 K words (assumption)

Address generated by CPU is divided into

- **Page number(p):** Number of bits required to represent the pages in Logical Address Space or Page number
- **Page offset(d):** Number of bits required to represent particular word in a page or page size of Logical Address Space or word number of a page or page offset.

Physical Address is divided into

- **Frame number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number.
- **Frame offset(d):** Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

The hardware implementation of page table can be done by using dedicated registers. But the usage of register for the page table is satisfactory only if page table is small. If page table contain large number of entries then we can use TLB(translation Look-aside buffer), a special, small, fast look up hardware cache.

- The TLB is associative, high speed memory.
- Each entry in TLB consists of two parts: a tag and a value.
- When this memory is used, then an item is compared with all tags simultaneously. If the item is found, then corresponding value is returned.

Main memory access time = m

If page table are kept in main memory,

Effective access time = m(for page table) + m(for particular page in page table)

Reg.No -----

[17MMU404B]

**Karpagam Academy Of Higher Education**

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641021

**B.SC Degree Examination**

(For the candidates admitted from 2017 onwards)

**FOURTH SEMESTER**

**THIRD INTERNAL EXAM**

**OPERATING SYSTEMS: LINUX**

**Time: 2 Hours**

**Maximum: 50 Marks**

**Date & Session:**

**Class: II B.SC (Maths)**

**PART-A ( 20 X 1 = 20 Marks )**

**Answer ALL the Questions**

1. The seek optimization strategy in which the disk arm is positioned next at the request (inward or outward) that minimizes arm movement is called \_\_\_\_\_.  
a) FCFS                      b) SSTF                      c) SCAN                      d) C-SCAN
2. The size of a logical block is usually \_\_\_\_\_ bytes  
a) 128                      b) 64                      c) 200                      d) 512
3. Linux refers to the privileged mode as \_\_\_\_\_.  
a) system mode                      b) linux mode                      c) unix                      d) kernel mode
4. The system uses the \_\_\_\_\_ to measure the amount of time used by a block of code  
a) interrupt object                      b) control object                      c) profile object                      d) timer object
5. Which one of the following is a process that uses the spawn mechanism to ravage the system performance?  
a) Worm                      b) Trojan                      c) Threat                      d) Virus
6. From the following, which is not common file permission?  
a) Write                      b) Execute                      c) Stop                      d) Read
7. The operation of building a new file is called \_\_\_\_\_.  
a) open                      b) close                      c) create                      d) destroy
8. The most widely used form of authentication is to require the user to type a \_\_\_\_\_.  
a) PIN number                      b) Login name.                      c) PIN number                      d) Username
9. Outsiders can sometimes take command of people's home computers (using viruses and other means) and turn them into \_\_\_\_\_.  
a) Virus                      b) Worms                      c) Malware                      d) Zombies
10. What are characteristics of Authorization?  
a) RADIUS and RSA                      b) 3 way handshaking with syn and fin.  
c) Multilayered protection for securing resources  
d) Deals with privileges and rights
11. Which of the following is least secure method of authentication?  
a) Key card                      b) Fingerprint                      c) Retina pattern                      d) Password
12. \_\_\_\_\_ is used for both anonymous and authenticated access  
a) http                      b) ftp                      c) smtp                      d) fdp
13. The seek optimization strategy in which the disk arm moves unidirectional across the disk Surface toward the inner track is called \_\_\_\_\_.





Reg.No -----

[17MMU404B]

**Karpagam Academy Of Higher Education**

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641021

**B.SC Degree Examination**

(For the candidates admitted from 2017 onwards)

**FOURTH SEMESTER**

**THIRD INTERNAL EXAM**

**OPERATING SYSTEMS: LINUX**

**Time: 2 Hours**

**Maximum: 50 Marks**

**Date & Session:**

**Class: II B.SC (Maths)**

**PART-A ( 20 X 1 = 20 Marks )**

**Answer ALL the Questions**

1. The seek optimization strategy in which the disk arm is positioned next at the request (inward or outward) that minimizes arm movement is called \_\_\_\_\_.  
a) FCFS                      b) **SSTF**                      c) SCAN                      d) C-SCAN
2. The size of a logical block is usually \_\_\_\_\_ bytes  
a) 128                      b) **64**                      c) 200                      d) 512
3. Linux refers to the privileged mode as \_\_\_\_\_.  
a) system mode                      b) linux mode                      c) unix                      d) **kernel mode**
4. The system uses the \_\_\_\_\_ to measure the amount of time used by a block of code  
a) **interrupt object**                      b) control object                      c) profile object                      d) timer object
5. Which one of the following is a process that uses the spawn mechanism to ravage the system performance?  
a) Worm                      b) **Trojan**                      c) Threat                      d) Virus
6. From the following, which is not common file permission?  
a) Write                      b) **Execute**                      c) Stop                      d) Read
7. The operation of building a new file is called \_\_\_\_\_.  
a) open                      b) close                      c) **create**                      d) destroy
8. The most widely used form of authentication is to require the user to type a \_\_\_\_\_.  
a) PIN number                      b) Login name.                      c) PIN number                      d) **Username**
9. Outsiders can sometimes take command of people's home computers (using viruses and other means) and turn them into \_\_\_\_\_.  
a) Virus                      b) **Worms**                      c) Malware                      d) Zombies
10. What are characteristics of Authorization?  
a) RADIUS and RSA                      b) 3 way handshaking with syn and fin.  
c) Multilayered protection for securing resources  
d) **Deals with privileges and rights**
11. Which of the following is least secure method of authentication?  
a) Key card                      b) Fingerprint                      c) Retina pattern                      d) **Password**
12. \_\_\_\_\_ is used for both anonymous and authenticated access  
a) http                      b) **ftp**                      c) smtp                      d) fdp
13. The seek optimization strategy in which the disk arm moves unidirectional across the disk Surface toward the inner track is called \_\_\_\_\_.  
a) SCAN                      b) **C-SCAN**

- c) N-Step scan d) Eschenbach scheme
14. The pattern that can be used to identify a virus is known as  
a) Stealth b) Virus signature c) Armored d) **Multipartite**
15. The authentication method that measures the physical characteristics of the user that are hard to forge is called as \_\_\_\_\_  
a) **Biometrics** b) Password c) Stenography d) Access control
16. Any malware hidden in software or a Web page that people voluntarily download is called \_\_\_\_\_  
a) Worm b) Trojan Horse c) **Virus** d) Backdoor
17. What are common security threats?  
a) File Shredding b) **File sharing and permission** c) File corrupting  
d) File integrity
18. A \_\_\_\_\_ defines a path from the current directory  
a) **Relative pathname** b) absolute path name c) Acyclic path  
d) direct pathname
19. Expand UFD  
a) **user file directory** b) uniform file directory  
c) user filter directory d) uniform filter directory
20. MS-DOS environment provided by a \_\_\_\_\_ application is called a VDM.  
a) win45 b) win60 c) **win32** d) win16

**PART-B (3 X 2 = 6 Marks)**  
**(Answer ALL the Questions)**

## 21. What is a Physical Address Space?

**Physical Address** identifies a **physical** location in a memory. ... The logical **address** is mapped to the **physical address** using a hardware called Memory-Management Unit. The set of all **physical addresses** corresponding to the logical **addresses** in a Logical **address space** is called **Physical Address Space**.

## 22. What are Authentication Mechanisms?

**Authentication** is the **mechanism** of verifying the identity of a user or an application. The simplest, default **authentication** method operates for a local connection by relying on **OS** user lookup.

## 23. What is Internal Access Authorization?

**Authorization** is a security mechanism to determine **access** levels or user/client privileges related to system resources including files, services, computer

programs, data and application features. ... 2) Policy enforcement phase where **access** requests are permitted or not permitted.

**PART-C (3 X 8 = 24 Marks)**  
**(Answer ALL the Questions)**

**24. a. Describe about Files? Explain the Access Methods for Files.**

When a file is used, information is read and accessed into computer memory and there are several ways to access this information of the file. Some systems provide only one access method for files. Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

**1. Sequential Access –**

It is the simplest access method. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.

Read and write make up the bulk of the operation on a file. A read operation -*read next*- read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the write *write next* append to the end of the file and advance to the newly written material.

1.

- Data is accessed one record right after another record in an order.
- When we use read command, it move ahead pointer by one
- When we use write command, it will allocate memory and move the pointer to the end of the file
- Such a method is reasonable for tape.

**2. Direct Access –**

Another method is *direct access method* also known as *relative access method*. A file-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14 then block 59 and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

**3. Index sequential method –**

It is the other method of accessing a file which is built on the top of the direct access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index and then by the help of pointer we access the file directly.

**Key points:**

- It is built on top of Sequential access.
- It control the pointer by using index.

## **b. Explain the Various File Operations.**

### **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**

A File Structure should be according to a required format that the 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 have 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.

---

### **25. a. Explain about OS Protection and Security.**

Protection and security requires that computer resources such as CPU, softwares, memory etc. are protected. This extends to the operating system as well as the data in the system. This can be done by ensuring integrity, confidentiality and availability in the operating system. The system must be protect against unauthorized access, viruses, worms etc.

#### **Threats to Protection and Security**

---

A threat is a program that is malicious in nature and leads to harmful effects for the system. Some of the common threats that occur in a system are:

##### **Virus**

Viruses are generally small snippets of code embedded in a system. They are very dangerous and can corrupt files, destroy data, crash systems etc. They can also spread further by replicating themselves as required.

##### **Trojan Horse**

A trojan horse can secretly access the login details of a system. Then a malicious user can use these to enter the system as a harmless being and wreak havoc.

### **Trap Door**

A trap door is a security breach that may be present in a system without the knowledge of the users. It can be exploited to harm the data or files in a system by malicious people.

### **Worm**

A worm can destroy a system by using its resources to extreme levels. It can generate multiple copies which claim all the resources and don't allow any other processes to access them. A worm can shut down a whole network in this way.

### **Denial of Service**

These type of attacks do not allow the legitimate users to access a system. It overwhelms the system with requests so it is overwhelmed and cannot work properly for other user.

### **Protection and Security Methods**

---

The different methods that may provide protect and security for different computer systems are:

#### **Authentication**

This deals with identifying each user in the system and making sure they are who they claim to be. The operating system makes sure that all the users are authenticated before they access the system. The different ways to make sure that the users are authentic are:

- **Username/ Password**

Each user has a distinct username and password combination and they need to enter it correctly before they can access the system.

- **User Key/ User Card**

The users need to punch a card into the card slot or use they individual key on a keypad to access the system.

- **User Attribute Identification**



Different user attribute identifications that can be used are fingerprint, eye retina etc. These are unique for each user and are compared with the existing samples in the database. The user can only access the system if there is a match.

### **One Time Password**

These passwords provide a lot of security for authentication purposes. A one time password can be generated exclusively for a login every time a user wants to enter the system. It cannot be used more than once. The various ways a one time password can be implemented are:

- **Random Numbers**

The system can ask for numbers that correspond to alphabets that are pre arranged. This combination can be changed each time a login is required.

- **Secret Key**

A hardware device can create a secret key related to the user id for login. This key can change each time.

### **b. Explain in detail about Policy Mechanism.**

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 an 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 chapter.

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

#### **Authentication**

Authentication refers to identifying 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 provide 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 passwords 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 example 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 suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generatlly 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 refers 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 creates such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

- **Worm** – Worm is a process which can choked 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. Worms 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, a 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    | <b>Type A</b><br>Highest Level. Uses formal design specifications and verification techniques. Grants a high degree of assurance of process security.  |
| 2    | <b>Type B</b><br>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. <ul style="list-style-type: none"><li>• <b>B1</b> – Maintains the security label of each object in the system. Label is used for making decisions to access control.</li></ul> |

|   |   |
|---|---|
|   | <ul style="list-style-type: none"> <li>• <b>B2</b> – Extends the sensitivity labels to each system resource, such as storage objects, supports covert channels and auditing of events.</li> <li>• <b>B3</b> – Allows creating lists or user groups for access-control to grant access or revoke access to a given named object.</li> </ul>  |
| 3 | <p><b>Type C</b></p> <p>Provides protection and user accountability using audit capabilities. It is of two types.</p> <ul style="list-style-type: none"> <li>• <b>C1</b> – Incorporates controls so that users can protect their private information and keep other users from accidentally reading / deleting their data. UNIX versions are mostly C1 class.</li> <li>• <b>C2</b> – Adds an individual-level access control to the capabilities of a C1 level system.</li> </ul> |
| 4 | <p><b>Type D</b></p> <p>Lowest level. Minimum protection. MS-DOS, Window 3.1 fall in this category.</p>   |

## 26. a. Explain about Device Management.

**Device Management** is another important function of the operating system. Device management is responsible for managing all the hardware devices of the computer system. It may also include the management of the storage device as well as the management of all the input and output devices of the computer system. It is the responsibility of the operating system to keep track of the status of all the devices in the computer system. The status of any computing devices, internal or external may be either free or busy. If a device requested by a process is free at a specific instant of time, the operating system allocates it to the process.

An operating system manages the devices in a computer system with the help of device controllers and device drivers. Each device in the computer system is equipped with the help of device controller. For example, the various devices controllers in a computer system may be disk controller, printer controller, tape-drive controller and memory controller. All these devices controllers are connected with each other through a system bus. The device controllers are actually the hardware components that contains some buffers registers to store the data temporarily. The transfer of data between a running process and the various devices of the computer system is accomplished only through these devices controllers.

## Some important points of device management

1. An operating system communicates with the devices controllers with the help of device drivers while allocating the device to the various processes running on the computer system.
  2. Device drivers are the software programs that are used by an operating system to control the functioning of various devices in a uniform manner.
  3. The device drivers may also be regarded as the system software programs acting as an intermediary between the processes and the devices controllers.
  4. The device controller used in the device management operation usually includes three different registers command, status, and data.
- 
5. The other major responsibility of the device management function is to implement the Application Programming Interface (API).

### b. Explain in detail about Directory Structures.

A [filename](#) is a string used to uniquely identify a file stored on the file system of a computer. Before the advent of [32-bit](#) operating systems, file names were typically limited to short names (6 to 14 characters in size). Modern operating systems now typically allow much longer filenames (more than 250 characters per [pathname](#) element).

### Windows 10<sup>[edit]</sup>

The following folders may appear in the root of a [boot partition](#).

| Folder               | Description   |
|----------------------|---|
| \PerfLogs (Hidden)   | May hold Windows performance logs, but on a default configuration, it is empty.   |
| \Program Files       | <b>32-bit architecture:</b> All apps (both 16-bit and 32-bit) are installed in this folder.<br><b>64-bit architecture:</b> 64-bit apps are installed in this folder.  |
| \Program Files (x86) | Appears on 64-bit editions of Windows. 32-bit and 16-bit apps are by default installed in this folder, even though 16-bit apps do not run on 64-bit Windows. <sup>[3]</sup>   |
| \ProgramData         | Contains program data that are expected to be accessed by computer programs regardless of the user account in the context of which they run. For example, an app may store specific information needed to operate <a href="#">DVD recorders</a> or <a href="#">image scanners</a> connected to a computer, because all users use them. Windows itself uses this folder. For example, <a href="#">Windows Defender</a> stores its <a href="#">virus definitions</a> in \ProgramData\Microsoft\Windows Defender. Programs do not have permission to store files in this folder, but have permission to create subfolders and store files in them. The |

|                                   |   |
|-----------------------------------|---|
|                                   | organization of the files is at the discretion of the developer.  |
| \Users                            | User profile folders. This folder contains one subfolder for each user that has logged onto the system at least once. In addition, it has two other folders: "Public" and "Default" (Hidden). It also has two folder like-items called "Default User" (an <a href="#">NTFS junction point</a> to "Default" folder) and "All Users" (a <a href="#">NTFS symbolic link</a> to "C:\ProgramData").  |
| \Public                           | This folder serves as a buffer for users of a computer to share files. By default this folder is accessible to all users that can log on to the computer. Also, by default, this folder is shared over the network, although anonymous access (i.e. without a valid password-protected user account) to it is denied. This folder contains user data, not program data, meaning that users are expected to be sole decider of what is in this folder and how it is organized. It is unethical for an app to store its proprietary data here. (There are other folders dedicated to program data.)               |
| [username]\AppData                | This folder stores per-user application data and settings. The folder contains three subfolders: Roaming, Local, and LocalLow. Roaming is for networked based logins for roaming profiles. Data saved in Roaming will synchronize to the computer when the user logs into that. Local and LocalLow does not sync up with networked computers. <sup>[4]</sup>  |
| \Windows                          | Windows itself is installed into this folder.   |
| \System<br>\System32<br>\SysWOW64 | These folders store <a href="#">dynamic-link library</a> (DLL) files that implement the core features of Windows and <a href="#">Windows API</a> . Any time a program asks Windows to load a DLL file and do not specify a path, these folders are searched after app's own folder is searched. <sup>[5]</sup> "System" stores 16-bit DLLs and is normally empty on 64-bit editions of Windows. "System32" stores either 32-bit or 64-bit DLL files, depending on whether the Windows edition is 32-bit or 64-bit. "SysWOW64" only appears on 64-bit editions of Windows and stores 32-bit DLLs. <sup>[6]</sup> |
| \WinSxS                           | This folder is officially called "Windows component store" and constitutes the majority of Windows. A copy of all Windows components, as well as all Windows updates and <a href="#">service packs</a> is stored in this folder. Starting   |

|  |  |
|--|--|
|  | <p>with <a href="#">Windows 7</a> and <a href="#">Windows Server 2008 R2</a>, Windows automatically scavenges this folder to keep its size in check. For security reasons and to avoid the <a href="#">DLL Hell</a> issue, Windows enforces very stringent requirements on how the files in this folder are organized.<sup>[7]</sup></p> |
|--|--|



**KARPAGAM ACADEMY OF HIGHER EDUCATION**

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Pollachi Main Road, Eachanari Post, Coimbatore – 641 021

(For the candidates admitted from 2017 onwards)

**B.Sc., DEGREE EXAMINATION, APRIL 2019**

Fourth Semester

**MATHEMATICS**

**OPERATING SYSTEM: LINUX**

Time: 3 hours

Maximum : 60 marks

**PART - A (20 x 1 = 20 Marks) (30 Minutes)**  
**(Question Nos. 1 to 20 Online Examinations)**

**PART B (5 x 2 = 10 Marks) (2 ½ Hours)**

**Answer ALL the Questions**

21. Define Swapping.
22. What is Critical Section?
23. List the scheduling algorithms
24. Define: Directory file.
25. What is a Program threats?

**PART C (5 x 6 = 30 Marks)**  
**Answer ALL the Questions**

26. a. Discuss about i. Batch System      ii. Real time System.  
      b. Explain in detail about Distributed System.  
      Or
27. a. Explain about Deadlock and its process  
      Or  
      b. Explain about process scheduling in detail
28. a. Explain about Virtual address space.  
      Or  
      b. Discuss about Paging in detail
29. a. Explain about Device management.  
      Or  
      b. Explain in detail about directory structure

30. a. Discuss about Internal access authorization.  
      Or  
      b. Describe the Security in Operating System.



Reg. No. ....

[17CAU401]

## KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University)

(Established Under Section 3 of UGC Act, 1956)

Pollachi Main Road, Eachanari Post, Coimbatore – 641 021

(For the candidates admitted from 2017 onwards)

**BCA DEGREE EXAMINATION, APRIL 2019**

Fourth Semester

### COMPUTER APPLICATIONS

### SOFTWARE ENGINEERING

Time: 3 hours

Maximum : 60 marks

**PART – A (20 x 1 = 20 Marks) (30 Minutes)**  
**(Question Nos. 1 to 20 Online Examinations)**

**PART B (5 x 2 = 10 Marks) (2 ½ Hours)**

**Answer ALL the Questions**

21. What is CMMI?
22. List the components of SRS.
23. Write short notes on: RMMM Plan.
24. What is meant by Design Engineering?
25. What are testing strategies?

**PART C (5 x 6 = 30 Marks)**  
**Answer ALL the Questions**

26. a. Describe Software Evolution.  
Or  
b. Explain the changing nature of software with suitable example.
27. a. Describe Flow Oriented Modeling.  
Or  
b. Elaborate estimation in Project planning process.
28. a. Discuss Risk identification and Risk projection.  
Or  
b. List the metrics for process and projects.

29. a. Elaborate Data design at the Architectural level.

Or

b. What is the role of Data Engineering in Software Development? Explain.

30. a. Write a brief note on: Software Testing Fundamentals.

Or

b. Explain White – Box testing and their type.

-----