Khushal Shetty

SPPU New Syllabus

A Book Of

Operating System Concepts

For M.C.A. (Management) : Semester - I

[Course Code IT-14 : Credit - 03]

CBCS Pattern

As Per New Syllabus, Effective from June 2020

Dr. Sunita Dhanesh Patil
B.Sc. (Computer Science), MCM, MCA, M.Phil. (IT), Ph.D. (Computer Management)
Dean Academics
Yashaswi Institute of Technology
Yashaswi Academy for Skills
Chinchwad, Pune

Price 250.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N5313

Syllabus ...

1. Overview **[Weightage 15, Session 7]**

- 1.1 Overview of Operating Systems
- 1.2 Functionalities and Characteristics of OS
- 1.3 Hardware Concepts related to OS
- 1.4 CPU States
- 1.5 I/O Channels
- 1.6 Memory Management
  - 1.6.1 Memory Management Techniques
  - 1.6.2 Contiguous and Non-Contiguous Allocation
  - 1.6.3 Logical and Physical Memory Conversion of Logical to Physical Address
- 1.7 Paging
  - 1.7.1 Demand Paging
  - 1.7.2 Page Replacement Concept
- 1.8 Segmentation - Segment with Paging
- 1.9 Virtual Memory Concept
- 1.10 Thrashing

Extra Reading: Type of OS, Batch OS, Time Sharing OS, Network OS, Multiprogramming OS, Multiprocessing OS, Evolution of Operating System, Computer System Organization Operating System Structure and Operations - System Calls, System Programs, OS Generation and System Boot.

2. Process Management and Synchronization **[Weightage 17, Session 8]**

- 2.1 PCB
- 2.2 Job and Processor Scheduling
- 2.3 Scheduling Concept
- 2.4 Process Hierarchies
- 2.5 Problems of Concurrent Processes
- 2.6 Critical Sections
- 2.7 Mutual Exclusion
- 2.8 Synchronization
- 2.9 Deadlock
- 2.10 Device and File Management
  - 2.10.1 Overview
  - 2.10.2 Techniques
  - 2.10.3 File Systems

Extra Reading: Threads – Overview, Multithreading Models, Threading Issues, Process Synchronization – The Critical-section Problem, Synchronization Hardware, Mutex Locks, Semaphores, Classic Problems of Synchronization, Critical Regions, Monitors; Deadlock – System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock., Banker's Algorithms.

3. Multiprocessor and Multicore Operating Systems [Weightage 17, Session 8]

- 3.1 Introduction
  - 3.1.1 Advantages and Disadvantages
  - 3.1.2 Multicore System Vs. Multiprocessor System
- 3.2 Types of Multiprocessors
  - 3.2.1 Symmetric Multiprocessors
  - 3.2.2 Asymmetric Multiprocessors
- 3.3 Basic Multicore Concepts: Memory Sharing Styles
  - 3.3.1 Uniform Memory Access (UMA)
  - 3.3.2 Non-Uniform Memory Access (NUMA)
  - 3.3.3 No Remote Memory Access (NORMA)
- 3.4 Cache Coherence, Inter-Process (and Intercore) Communication
  - 3.4.1 Shared Memory
  - 3.4.2 Message Passing
- 3.5 Mobile Operating Systems
  - 3.5.1 Concept Need and Features
  - 3.5.2 Types of Mobile OS
  - 3.5.3 Overview of Android OS
  - 3.5.4 Applications of Mobile OS
- 3.6 Distributed Operating Systems
  - 3.6.1 Concept Need and Features
  - 3.6.2 Examples of Distributed OS with brief introduction
  - 3.6.3 Applications of Distributed OS

Extra Reading: Virtual Machine, Cache Memory and Catching Concept, Multi-Processor and Distributed Operating System – Introduction – Architecture – Organization, Resource sharing – Load Balancing – Availability and Fault Tolerance – Design and Development Challenges – Inter-process Communication.

4. Real Time OS [Weightage 10, Session 4]

- 4.1 Introduction and use of RTOS
  - 4.1.2 Components of RTOS
  - 4.1.3 Types of RTOS
  - 4.1.4 Features of RTOS
  - 4.1.5 Factors for Selecting in RTOS
  - 4.1.6 Applications of RTOS
  - 4.1.7 Disadvantages of RTOS
- 4.2 Embedded OS
  - 4.2.1 Concept Need and Features of Embedded OS
  - 4.2.2 Examples of Embedded OS with brief Introduction
  - 4.2.3 Applications of Embedded OS

Extra Reading: Real Time and Embedded Operating Systems – Introduction, Hardw Elements, – Structure Interrupt Driven, Interrupt Driven, Nanokernel, Nanoker Microkernel and Microkernel and Monolithic kernel based models. Monolithic ker based models. – Scheduling – Periodic, Periodic, Aperiodic and Aperiodic and Sporad Tasks, Sporadic Tasks, – Introduction to Energy Aware CPU Scheduling.

5. Windows OS and Windows Server Architecture [Weightage 24, Session 1

- 5.1 Windows OS
  - 5.1.1 Introduction
  - 5.1.2 Windows OS Installation
  - 5.1.3 Process Management
  - 5.1.4 Control Panel Overview
  - 5.1.5 Users, Security and Privacy Settings
  - 5.1.6 Identify Accessibility Settings
  - 5.1.7 Service Management
  - 5.1.8 Syncing Devices and File Sharing
  - 5.1.9 Windows Utilities (Accessories, Disk Management, Resource Monitor, Backup and Recovery), Basic Troubleshooting (Networking, Security, Device Driver).
- 5.2 Introduction to Ubuntu
  - 5.2.1 Introduction
  - 5.2.2 Overview of Kernel
  - 5.2.3 Installation of Ubuntu
  - 5.2.4 File System
  - 5.2.5 Basic Commands of Linux
  - 5.2.6 Managing Processes in Linux
  - 5.2.7 Installing and Deleting Software Packages
  - 5.2.8 User Management
  - 5.2.9 File and Device Management
  - 5.2.10 Backup and Recovery
  - 5.2.11 Introduction to Graphical Environment (GNOME), Ubuntu Utilities (VirtualBox, Evolution, Gimp, Bleach Bit, Unity Tweak Tool etc.), SAMBA Overview

Extra Reading: Deploying and Managing Windows Server 2012 and 2016, Introduction to Active Directory Domain Services, Managing Active Directory Domain Services Objects, Automating Active Directory Domain Services Administration, Implementing IPv4, Implementing DHCP, Implementing DNS, Implementing Local Storage, Implementing File and Print Services, Implementing Group Policy.

[Weightage 16, Session 6]

6. Linux Shell Scripting

6.1 Introduction
6.2 Variables
6.3 Flow Controls
6.4 Loops
6.5 Functions
6.6 Lists
6.7 Manipulating Strings
6.8 Reading and Writing Files
6.9 Positional Parameters
6.10 Case Statement
6.11 Real Time Scripts for different System Administration Activities

Extra Reading: Shell Script Programming Concepts, Sequential Flow and Components of Shell Scripting, Decision Structures, Decision - Structure Theory, Statements and Operators, Looping Structures, Loop Theory and Statements, Functions and Arrays, Functions Parts/Libraries and Arrays, Advanced Shell Programming, File Access, Sorts and Techniques, Advanced Tech and Tools, Awk and Sed, Script Design and Management Issues.

■■■

Contents ...

■■■

1... Overview

Objectives ...

- To understand overview of Operating System.
- To study functionalities and characteristics of OS.
- To learn Hardware concepts related to OS.
- To learn about memory management.
- To study concept of Paging.
- To understand virtual memory concept.

1.1 OVERVIEW OF OPERATING SYSTEM

- An operating system is a collection of system programs that controls computer and any other peripherals connected to it.
- Operating system shields the programmer from the interface, the abstraction offers by the operating system is slower & easier to use than the underlying hardware.
- Operating system is collection of software which is close to hardware. We can view operating system as a resource – hardware and software collector.
- We can call operating system as Resource Allocator. A computer has many resources hardware and software, CPU, main memory, I/O devices etc. The operating system acts as manager of these resources.
- Operating system is the control program. A control program manages the execution of user program to prevent error and improve use of computer. The storage device used to store operating system.

1.2 FUNCTIONALITIES AND CHARACTERISTICS OF OS

Important functions of an Operating System:

- **Managing Resources:** Programs that manage the resources of a computer such as the printer, mouse, keyboard, memory and monitor.
- **Providing User Interface:** Graphical user interface (GUI) is something developers create to allow users to easily click something without having to understand how or why they clicked an icon. Each icon on a desktop represents code linking to the spot in which the icon represents. It makes it very easy for uneducated users.

- **Running Applications:** It is the ability to run an application such as Word processor by locating it and loading it into the primary memory. Most operating systems can multitask by running many applications at once.
- **Support for built-in Utility Programs:** This is the program that find and fixes errors in the operating system.
- **Control Computer Hardware:** All programs that need computer hardware must go through the operating system which can be accessed through the BIOS (Basic Input Output System) or the device drivers.

Characteristics of Operating System:

- **Security:** The operating system uses password protection to protect user data and similar other techniques. It also prevents unauthorized access to programs and user data.
- **Control over system performance:** Monitors overall system health to help improve performance, records the response time between service requests and system response to have a complete view of the system health. This can help improve performance by providing important information needed to troubleshoot problems.
- **Job accounting:** Operating system keeps track of time and resources used by various tasks and users. This information can be used to track resource usage for a particular user or group of user.
- **Error detecting aids:** Operating system constantly monitors the system to detect errors and avoid the malfunctioning of computer system.
- **Co-ordination between other software and users:** Operating systems also co-ordinate and assign interpreters, compilers, assemblers and other software to the various users of the computer systems.
- **Memory Management:** The operating system manages the Primary Memory or Main Memory. Main memory is made up of a large array of bytes or words where each byte or word is assigned a certain address. Main memory is a fast storage and it can be accessed directly by the CPU. For a program to be executed, it should be first loaded in the main memory.
- **Processor Management:** In a multi-programming environment, the OS decides the order in which processes have access to the processor, and how much processing time each process has. This function of OS is called Process Scheduling. It keeps track of the status of processes. The program which perform this task is known as traffic controller. Allocates the CPU that is processor to a process. De-allocates processor when a process is no more required.
- **Device Management:** An OS manages device communication via their respective drivers. It performs the following activities for device management. Keeps tracks of all devices connected to system. Designates a program responsible for every device known as the Input/Output controller. Decides which process gets access to a certain device and for how long. Allocates devices in an effective and efficient way.

- **File Management:** A file system is organized into directories for efficient or easy navigation and usage. These directories may contain other directories and other files. An Operating System carries out the file management activities. It keeps track of where information is stored, user access settings and status of every file and more. These facilities are collectively known as the file system.

1.3 HARDWARE CONCEPTS RELATED TO OPERATING SYSTEM

- An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of its software and hardware. It also allows you to communicate with the computer without knowing the computer's language.
- 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.

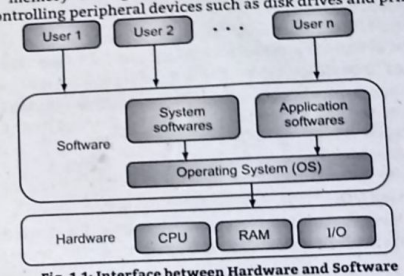


Fig. 1.1: Interface between Hardware and Software

- One of the important job of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/Off switch, network connections, audio I/O, printers etc.
- An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application.
- I/O devices can be divided into two categories:
  1. **Block devices:** A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.
  2. **Character devices:** A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). For example, serial ports, parallel ports, sound cards etc.

1.3.1 Device Controllers

- Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.
- The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical component and an electronic component where electronic component is called the Device Controller.
- There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.
- Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller.

Synchronous Vs Asynchronous I/O:

- **Synchronous I/O:** In this scheme CPU execution waits while I/O proceeds.
- **Asynchronous I/O:** I/O proceeds concurrently with CPU execution.

Communication to I/O Devices:

- The CPU must have a way to pass information to and from an I/O device. There are three approaches available to communicate with the CPU and Device.
  - Special Instruction I/O
  - Memory-mapped I/O
  - Direct memory access (DMA)

1. **Special Instruction I/O:**
- This uses CPU instructions that are specifically made for controlling I/O devices. These instructions typically allow data to be sent to an I/O device or read from an I/O, device.

2. **Memory-mapped I/O:**
- When using memory-mapped I/O, the same address space is shared by memory and I/O devices. The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.

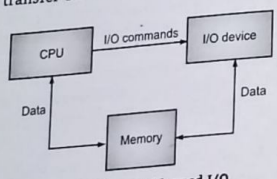


Fig. 1.2: Memory Mapped I/O

- While using memory mapped I/O, OS allocates buffer in memory and informs I/O device to use that buffer to send data to the CPU. I/O device operates asynchronously with CPU, interrupts CPU when finished.
- The advantage to this method is that every instruction which can access memory can be used to manipulate an I/O device. Memory mapped I/O is used for most high-speed I/O devices like disks, communication interfaces.

3. **Direct Memory Access (DMA):**
- Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So a typical computer uses Direct Memory Access (DMA) hardware to reduce this overhead.
- Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

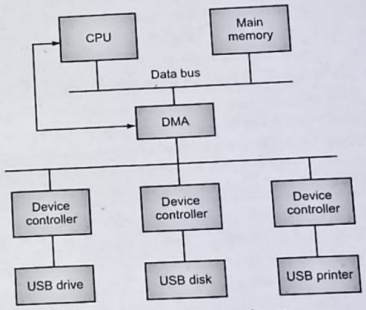


Fig. 1.3: Direct Memory Access

- Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.

1.4 CPU STATES

- The basic unit of software that the operating system deals with in scheduling the work done by the processor is either a process or a thread, depending on the operating system. It is processes, rather than applications, that the operating system controls and schedules for execution by the CPU.
- There are 3 general states your CPU can be in:
  - **Idle**, which means it has nothing to do.
  - **Running a user space program**, like a command shell, an email server, or a compiler.
  - **Running the kernel**, servicing interrupts or managing resources.
- These three meta states can be further subdivided. For example, user space programs can be categorized as those running under their initial priority level or those running with a nice priority. Niceness is a way to tweak the priority level of a process so that it runs less frequently. The niceness level ranges from -20 (most favorable scheduling) to 19 (least favorable). By default processes on Linux are started with a niceness of 0.

The 7 CPU statistics explained:

- There are several different ways to see the various CPU statistics. The most common is probably using the top command.
- To start the top command you just type top at the command line.
- The output from top is divided into two sections. The first few lines give a summary of the system resources including a breakdown of the number of tasks, the CPU statistics, and the current memory usage. Beneath these stats is a live list of the current running processes. This list can be sorted by PID, CPU usage, memory usage, and so on.
- The CPU line will look something like this:

```
%Cpu(s): 24.8 us, 0.5 sy, 0.0 ni, 73.6 id, 0.4 wa, 0.0 hi, 0.2 si, 0.0 st
```

1.5 I/O CHANNELS

- The Input/Output (I/O) channels provide for communications between the CPU and all peripheral devices. This is accomplished by electrical cables that carry both data and control information to and from the computer and peripheral devices.
- Signals are transmitted and received through a cable connecting the CPU and its online devices. This cable or line provides a path for the signal to travel and is called a Channel. Not only signals for monitoring but also data are transmitted via channels.
- All channels between the CPU and the peripheral devices are designated as I/O channels. An I/O channel may be used for data input, data output, or data input and output, depending on whether the peripheral device handles input only, output only, or both input and output.

Types of Channel:

1. **Simplex Channels:**
   - In simplex operations, communications are in one direction only, such as a radio. If a device such as a terminal were to be connected to such a circuit, it would only be capable of sending or receiving data, but not both. For this reason simplex circuits are seldom used, because a return path is generally needed to send acknowledgment, control information, or some type of error signals.
2. **Duplex Channels**
   - A duplex channel simply means that within each cable connection, there are two paths (lines) for the transmission of data. One path is for sending, and one is for receiving, similar to your telephone.
   - There are two types of duplex channels, half-duplex, and full-duplex.
     (a) **Half-duplex:** A half-duplex channel is capable of transmitting and receiving signals, but only in one direction at a time, similar to citizens' band (CB) radio transmissions. Therefore, it is necessary to check that the line is clear (idle) before starting a transmission.
     (b) **Full-duplex:** A full-duplex channel provides for simultaneous transmission in both directions, as in the use of the telephone.

1.6 MEMORY MANAGEMENT

- The memory management modules of an operating system are concerned with the management of primary memory. Primary memory is the one which the processors directly access for instructions and data.
- Primary memory is frequently called core memory.
- The memory management algorithms vary from a primitive bare-machine approach to paging and segmentation strategies.
- Selection of a memory management method for a specific system depends on many factors, especially on the hardware design of the system.
- Memory is central to the operation of a modern computer system. Both the CPU and I/O system interact with memory.
- Memory is a large array of words or bytes, each with its own address. Interaction is achieved through a sequence of reads or writes to specific memory addresses.

Functions of Memory Management:

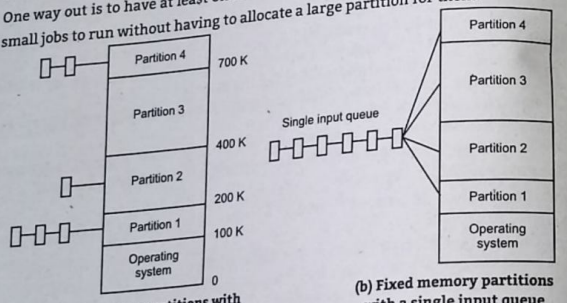
- Operating system consist of memory management module which will have a following functions:
  1. Keep the track of status of Resource (Memory).
  2. Decide the policy to allocate a free memory to a job or process.
  3. Enforce the policy to allocate memory.
  4. Allocation of free memory to a job.
  5. Deallocation of allocated memory after completion of job.

1.6.1 Memory Management Techniques

- Memory management consist of several techniques, some of them are listed below:

1. **Multiprogramming with Fixed Partitions:**

- In fixed partitions, main memory is divided into a number of partitions of system generation time.
- A process may be loaded into a partition of the equal or greater size.
- It is often useful to have more than one process in memory at once. Memory is divided into n (possibly unequal) partitions. This partitioning can be done manually by the operator when the system is started up.
- When a job arrives, it can be put into the input queue for the smallest partition large enough to hold it. Since, the partitions are fixed in this scheme; any space in a partition not used by a job is lost.
- The Fig. 1.4 shows how this fixed partitions and separate input queues looks.
- The disadvantage of sorting the incoming jobs into separate queues becomes apparently when the queue for a large partition is empty but the queue for a small partition is full, as is the case for partitions 1 and 4 in Fig. 1.4 (a).
- An alternative organisation is to maintain a single queue as in Fig. 1.4 (whenever, a partition becomes free, the job closest to the front of the queue that fits in it could be loaded into the empty partition and run.
- One way out is to have at least one small partition around. Such a partition will allow small jobs to run without having to allocate a large partition for them.



(a) Fixed memory partitions with separate input queues for each partition

(b) Fixed memory partitions with a single input queue

Fig. 1.4: Multiprogramming with fixed partitions

2. **Multiprogramming with variable partitions:**

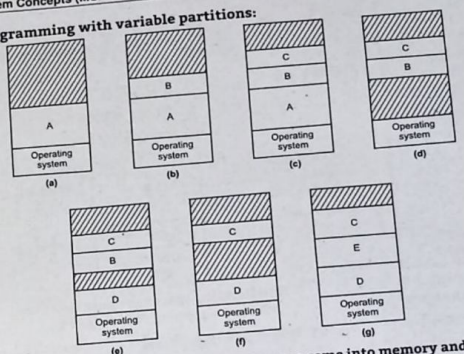
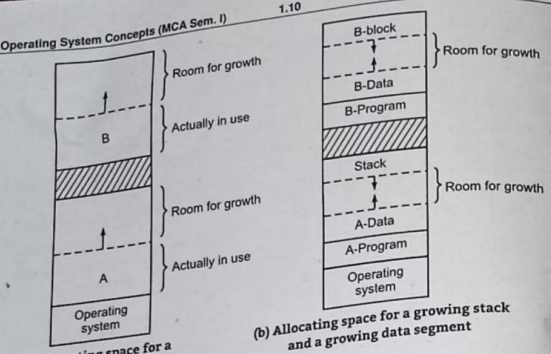


Fig. 1.5: Memory allocation changes as processes come into memory and leave it. The shaded regions are unused memory

- A swapping system could be based on fixed partitions. Whenever, a process is blocked, it could be moved to the disk and another process is brought into its partition from the disk.
- Too much memory is wasted by programs that are smaller than their partitions. A different memory management algorithm is used. It is known as Variable Partitions.
- When variable partitions are used, the number and size of the processes in memory vary dynamically. Fig. 1.5 shows how variable partitions work. Initially only process A is in memory.
- Then processes B and C are created or swapped in from disk in Fig. 1.5. A terminates or is swapped out to disk. Then D comes in and B goes out. Finally E comes in.
- If a hole (Empty space) is adjacent to the process, it can be allocated and the process allowed to grow into hole.
- On the other hand, if the process is adjacent to another process, the growing process will either have to be moved to a hole in memory large enough for it or one or more processes will have to be swapped out to create a large enough hole. If a process cannot grow in memory and the swap area on the disk is full, the process will have to wait or be killed.
- If it is expected that most processes will grow as they run, it is probably a good idea to allocate a little extra memory whenever, a process is swapped in or moved, to reduce the overhead associated with moving or swapping processes that no longer fit in their allocated memory.



(a) Allocating space for a growing data segment

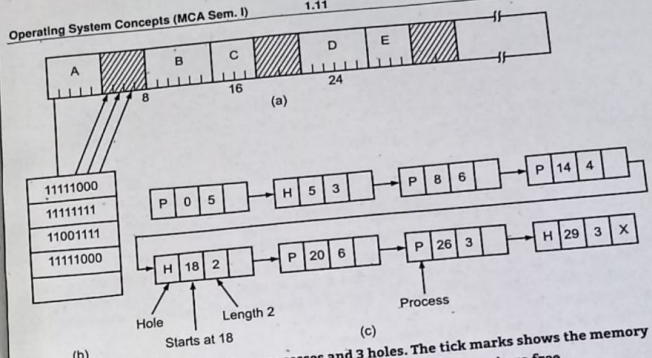
(b) Allocating space for a growing stack and a growing data segment

Fig. 1.6: Memory Configuration

- However, when swapping processes to disk, only the memory actually in use should be swapped, it is wasteful to swap the extra memory as well. The Fig. 1.6 (a) memory configuration is shown in which space for growth has been allocated to two processes.
- In Fig. 1.6 (b), each process has a stack at the top of its allocated memory growing downward and a data segment just beyond the program text, growing upward.

3. Memory Management with Bit Maps:

- In this type of management, memory is divided up into allocation units. Corresponding to each allocation unit is a bit in the bit map, which is 0 if the unit is free and 1 if it is occupied.
- Fig. 1.7 shows part of memory and the corresponding bit map.
- The size of allocation unit is an important factor. The smaller the allocation unit, the larger the bit maps.
- A bit map provides a simple way to keep track of memory words in a fixed amount of memory because the size of the bit map depends only on the size of memory and the size of allocation unit.
- Bit maps are not often used, because searching a bit map for a run of a given length is a slow operation.



(a) A part of memory with five processes and 3 holes. The tick marks shows the memory allocation units. The shaded regions (0 in the bit map) are free.

(b) The corresponding bit map.

(c) The same information as a linked list.

Fig. 1.7: Part of Memory and Corresponding Bitmap

4. Memory Management with Linked Lists:

- To keep track of memory, is to maintain a linked list of allocated and free memory segments, where a segment is either a process or a hole between two processes.
- Fig. 1.8 shows four neighbours combinations for the terminating process, X.
- In this example, the segment list is kept sorted by address. When the process terminates or is swapped out, updating the list is straightforward.
- A terminating process normally has two neighbours. These may be either processes or holes, leading to the four combinations as shown in Fig. 1.8.
- In Fig. 1.8 (a) updating the list requires replacing a P by an H. In Fig. 1.8 (b) and 1.8 (c), two entries are merged and the list becomes one entry shorter. In Fig 1.8 (d), three entries are merged and two items are removed from the list.
- When the processes and holes are kept on a list sorted by address, several algorithms can be used to allocate memory for a newly created or swapped in process. The memory knows how much memory to allocate.
- The simplest algorithm is **first fit**. The memory manager scans along the list of segments until it finds a hole. The hole is then broken up into two pieces, one for the process and one for the unused memory.
- First fit is a fast algorithm because it searches as little as possible.

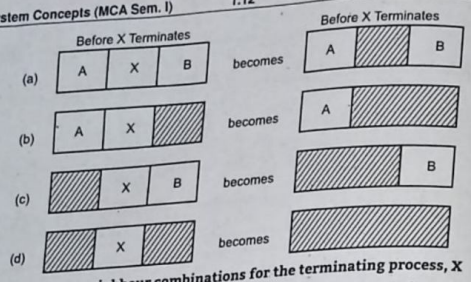


Fig. 1.8: Four neighbour combinations for the terminating process, X

- A minor variation of first fit is **next fit**. It works the same way as first fit, except that it keeps track of where it is when it finds a suitable hole.
- Another good algorithm is **best fit**. Best fit searches the entire list and takes the smallest hole that is adequate.

5. Memory Management with Buddy System:

- The buddy system is a memory management algorithm that takes the advantage of the fact that computers use binary numbers for addressing in order to speed up the merging of adjacent holes when a process terminates or is swapped out.

6. Allocation of Swap Space:

- The above algorithms are present to keep track of main memory so that when processes are swapped in, the system can find space for them. In some systems, when a process is in memory, no disk space is allocated to it.
- When it must be swapped out, space must be allocated in the disk swap area for it. On each swap, it may be placed somewhere else on disk. The algorithms for managing swap space are the same ones used for managing main memory.
- In other systems, when a process is created, swap space is allocated for it on disk. Whenever, the process is swapped out, it is always swapped to its allocated space, rather than going to a different place each time. When the process exits, the swap space is deallocated.

7. Analysis of Swapping Systems:

- The ratio of holes (empty space in main memory) to processes can be found by the following analysis.

Consider an average process in the middle of memory after the system has come to equilibrium. During its stay in memory, half of the operations on the segment just above it will be process allocations and half will be process deallocations.

Thus, half the time it has another process as upper neighbour, and half the time it has a hole as upper neighbour.

- Averaged over time, there must be half as many holes as processes. In other words, if the mean number of processes in memory is n, the mean number of holes is n/2. This result is known as fifty percent rule.
- Another useful result is the unused memory rule. Let f be the fraction of memory occupied by holes, 's' be the average size of then processes, and 'Ks' be the average hole size for some K > 0. With a total memory of m bytes, the n/2 holes occupy m-ns bytes, Algebraically,

$$(n/2) \times Ks = m\text{-}ns$$

Solving this equation for m, we get

$$m = ns\,(1 + K/2)$$

- The fraction of memory, in holes is just the number of holes, n/2, times the average hole size, Ks, divided by the total memory, m.

$$f = \frac{nKs/2}{m} = \frac{nKs/2}{ns(1 + K/2)} = \frac{K}{1+2}$$

8. Swapping:

- The partitioned memory management scheme is categorized whether it supports swapping or not. Lifting the program from the memory and placing it on the disk is called as "swapping out". To bring the program again from the disk to main memory is called as "swapping in".
- Normally, a blocked process is swapped out to make room for a ready process to improve the CPU utilization. If more than one process is blocked, the swapper chooses a process with lowest or a process waiting for a slow I/O event for swapping out.
- The operating system has to find a place on the disk for the swapped out process image. There are two alternatives one is to create a separate swap file for each process. This method is flexible but can be very important due to the increased number of files and dictionary entries.
- The other alternative is to keep a common swap file on the disk and not the location of each swapped out process image within that file. In this method, the estimate, of the swap file has to be made initially.
- If smaller area is reserved for this file, the operating system may not be able to swap out processes beyond a certain limit and thus affecting the performance.

9. Fragmentation:

- Several factors must be considered in selecting a partitioned memory algorithm. Speed and simplicity are among them. More important concern is the effect of fragmentation, the development of a large number of separate free areas i.e. total free memory is fragmented into small pieces.

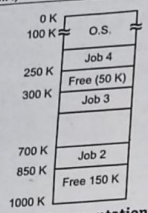


Fig. 1.9: Fragmentation

- By referring the following Fig. 1.9. Say two jobs are terminated and total of say 200 K of memory is free. In Fig. 1.9, Job 1 and Job 5 are terminated. If request were made for a partition of size 175 K it could not be possible to allocate such a partition.

(i) Although a total of 200 K bytes of free memory are available there is no single area larger than 150 K.

(ii) This type of fragmentation is called as External Fragmentation. It exists, when enough total memory space exists to satisfy a request, but it is not contiguous, storage is fragmented into large number of small pieces.

(iii) Memory fragmentation can be internal as well as external. Consider a multiple partition allocation scheme with a partition 500 bytes. Suppose that the next process request 480 bytes. In such case we left with fragments 20 bytes.

(iv) The overhead to keep track of this fragment will be larger than fragment itself.

(v) The general approach is to break the physical memory into fixed sized blocks and allocate memory in unit of block size. With this approach, the memory allocated to a process may be slightly larger than the requested memory. The difference between these two numbers is Internal Fragmentation i.e. memory which is internal to a portion but is not being used.

1.6.2 Contiguous and Non-Contiguous Memory Allocation

- In contiguous memory allocation, each programs data and instructions are allocated a single contiguous space in memory.
- In Non-contiguous memory allocation, each programs data and instructions are allocated memory space that is not continuous.
- Contiguous memory management schemes expect the program to be loaded in contiguous memory locations.
- Non-contiguous systems remove this restriction. They allow the program to be divided into different chunks and to be loaded at different portions of the memory.
- It is then the function of the operating system to manage these different chunks in such a way that they appear to be contiguous to the application programmer/user.

- In "paging", these chunks are of the same size, whereas in "segmentation", they can be of different sizes.
- The primary memory must accommodate both the user processes and operating system. For this reason, we need to allocate the parts of the memory in the most efficient way possible.
- The memory is usually divided into two partitions one partition for the resident operating system and another for the user processes. We can place the operating system in either high memory or low memory.
- The major factor affecting this decision is the location of the interrupt vector. Since, the interrupt vector is often in low memory; programmers usually place the operating system in low memory as well.
- Usually, we want several user processes to reside in memory at the same time. Therefore, we need to consider how to allocate available memory to the processes that are in the input queue waiting to be brought into memory.
- In contiguous memory allocation, each process is contained in a single contiguous section of memory.
- The contiguous allocation means the memory spaces is divided into the small and equal size and the different process uses the various partitions for running their applications process.
- And when the request has found then the process will allocate the space. And in this the contiguous spaces is provided to each and every process.

1.6.3 Logical and Physical Memory

- Logical Address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as Virtual Address. This address is used as a reference to access the physical memory location by CPU. The term Logical Address Space is used for the set of all logical addresses generated by a program's perspective. The hardware device called Memory-Management Unit is used for mapping logical address to its corresponding physical address.
- 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.
- The compile-time and load-time address-binding methods generate identical logical and physical addresses.
- However, the execution-time address binding scheme results in differing logical and physical addresses. In this case, we usually refer to the logical address as a virtual address.

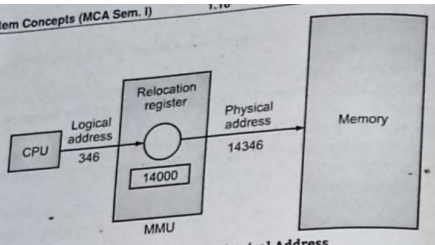


Fig. 1.10: Logical and Physical Address

- The set of all logical addresses generated by a program is a logical address space; the set of all physical addresses corresponding to these logical addresses is a physical address space. Thus, in the execution-time address-binding scheme, the logical and physical address space differ.
- The run-time mapping from virtual to physical address is done by a hardware device called the Memory Management Unit (MMU).

1.6.4 Conversion of Logical to Physical Address

- The run-time mapping logical to physical addresses is done by Memory Management Unit (MMU).
- MMU makes use of a register called as relocation or base register. The value in the base register is added to every address generated by any our process.
- The base register is also called as relocation register. The value in the relocation register is added to every address generated by our process at the time it is sent to memory.
- For example, if the base is at 17000, then an attempt by the user to address location 0 is dynamically relocated to location 17000; an access to location 4x86 is mapped to location 17400.

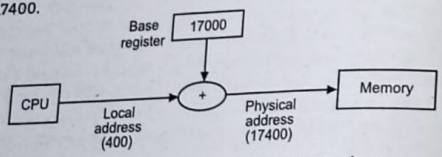


Fig. 1.11: Dynamic Relocation using Base Register

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

- Only when it is used as a memory address is it relocated relative to the base register. Our program code deals with logical addresses.
- The memory-mapping device converts logical addresses into physical addresses.

1.7 PAGING

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

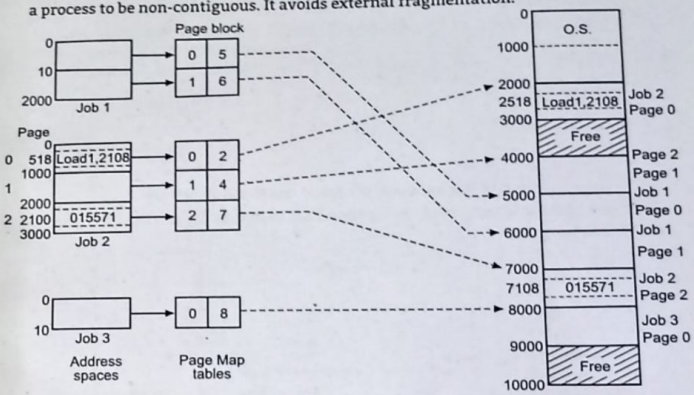


Fig. 1.12: Paging

- In paging each job's address space is divided into equal pieces, called "pages", and likewise, physical memory is divided into pieces of same. Size, called "blocks" or "Frames". Then by providing suitable hardware mapping facility any "page" can be mapped to any "block".
- The page remains logically contiguous but the corresponding blocks are not necessarily contiguous.
- Paging permits a programs memory to be non-contiguous, thus allowing a program to be allocated physical memory wherever it is available.
- Paging model of logical and physical memory is shown in Fig. 1.13.
- Every address generated by the CPU is divided into two parts: A page number (p) and a page offset (d). The page number is used as an index into a page table. The page table contains the base address of each page in physical memory.

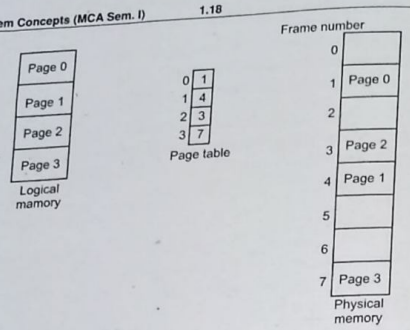


Fig. 1.13: Paging model of Logical and Physical memory

- This base address is combined with page offset to define the physical address that is sent to memory unit.

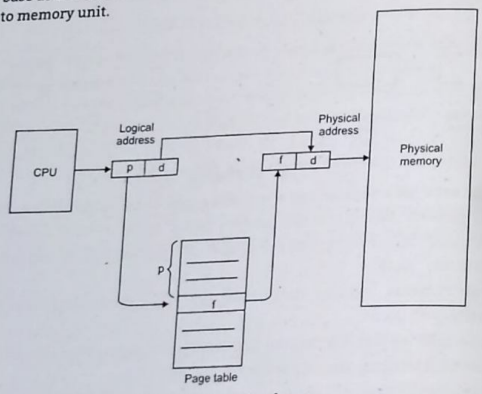


Fig. 1.14: Paging hardware

- Physical memory is broken into fixed size blocks called **frames**. Logical memory is also broken into blocks of the same size called **pages**.
- When a program is to be executed, its pages are loaded into any available frames and the page table is defined to translate from user pages to memory frames.

- For example, using a page size of 4 words and physical memory of 32 words (8 pages) we show how the user's view of memory can be mapped into physical memory. Logical address 0 is page 0 offset 0. We find that page 0 is in frame 5. Thus, logical address 0 maps to physical address $(5 \times 4 + 0) = 20$. Logical address 4 is page1, offset 0. Logical address 4 maps to physical address $(6 \times 4 + 0) = 24$.
- Paging itself is a form of dynamic relocation. Every logical address is mapped by paging hardware to some physical address.
- Each user page needs one frame. Thus, if the job requires n pages, there must be n frames available in memory.
- The page of job is loaded into one of the allocated frames and the frame number is put in the page table for this job and so on.
- Using a paging scheme, we have no external fragmentation, any free frame can be allocated to a job that needs it. Each job has its own page table. The page table is implemented as a set of dedicated registers.

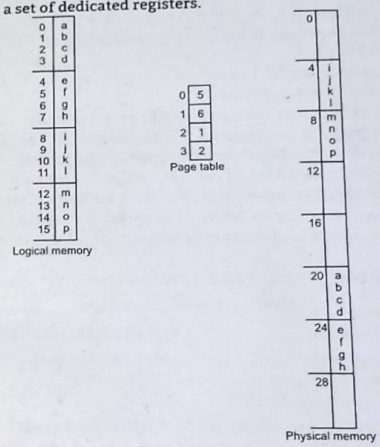


Fig. 1.15: Paging example for a 32-word memory with 4-word pages

Advantages of Paging:

1. It permits physical address space of process to be non-contiguous and support virtual memory concept.
2. It maintains clear separation between user's view of memory and actual physical memory.
3. It does not require any support for dynamic relocation because paging itself is a form of dynamic relocation. Every logical address is bound by paging hardware to physical address.

4. It does not suffer from external fragmentation. Any free frames can be allocated to process that it needs.
5. We can share common code, so memory space is properly utilized.

Disadvantages of Paging:

1. Paging suffers from internal fragmentation. Internal fragmentation especially on last page of process; average of 50% on last page.
2. Smaller frames create less fragmentation but increase number of frames and size of page table.
3. Overhead to maintain and update page table.
4. Paging hardware increases the cost of operating system.

1.7.1 Demand Paging

- Demand paging policy or technique transfers memory pages instead of process to and from secondary storage. The entire process does not have to reside in main memory to execute and the kernel loads pages for a process and demand when the process references the pages.
- A demand paging system is similar to a paging system with swapping where processes reside in secondary memory.
- When we want to execute a process then we swap into memory.
- Rather than swapping the entire process into memory we use lazy swapper and it never swaps a page into memory unless that page will be needed.
- When a program starts with a list of available option from which the user is to select. Loading the entire program into memory results in loading the executable code for all options regardless of whether an option is ultimately selected by the user or not. An alternative approach is to load pages only as they needed this technique is known as demand paging.
- Fig. 1.16 shows transfer of a paged memory to contiguous disk space.

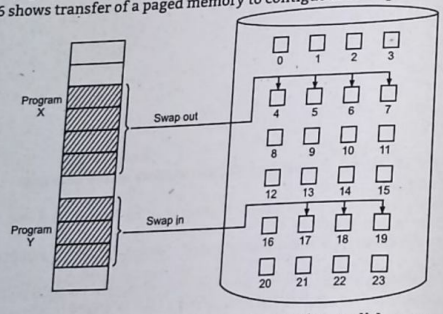


Fig. 1.16: Transfer of the paged memory to contiguous disk space

Pure Demand Paging:

- We start executing a job with no pages in memory. When operating system points towards first instrumentation first page fault occurs. Subsequent page faults occur and the necessary pages will be brought in memory.

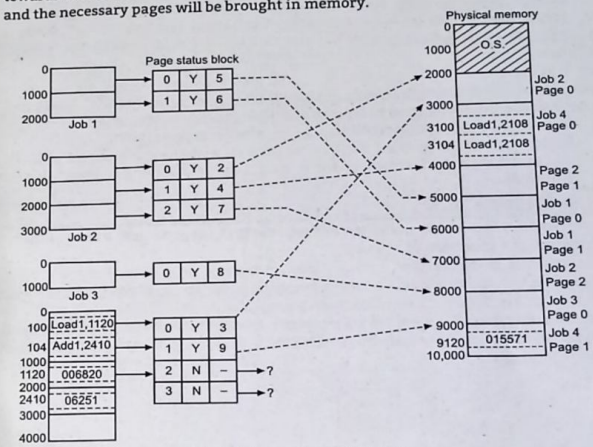


Fig. 1.17: Page Map Table (PMT)

1.7.2 Page Replacement Concept

- When a page fault occurs, the operating system has to choose a page to remove from memory to make room for the page that has to be brought in. This is known as **page replacement**.
- Page replacement algorithms should have the lowest page fault rate.
- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page p, then any immediately following references to page p will never cause a page fault. Page p will be memory after the first reference; hence, the immediately following references will not cause a page fault.
- As the number of frames increases, the numbers of page faults are minimized.
- Page replacement is basic to demand paging. It completes the separation between logical memory and physical memory. With this mechanism an enormous virtual memory can be provided for programmers on a smaller physical memory.

- Every operating system usually has its own page replacement scheme.
- "What is the best page to remove"? It is the page that will not need again or at least, not for a long time.
- Some of the algorithm for page replacements are:
  1. FIFO (First In First Out).
  2. LRU (Least Recently Used).
  3. Optimal, page replacement.
  4. MFU,
  5. LFU, etc.
- Basic idea behind the replacement analogy "Incharge of super market". Now product, where do you put it you have to replace a product already on your shelf.
  1. Replace the product that has been on your shelves longest (FIFO).
  2. Blow a dust off the product. Replace the product with the most dust on it (LRU).
  3. Replace the product which is for a long time and having a most dust on it (Optimal).

1.8 SEGMENTATION

- Segmentation is memory management scheme that supports user's view of memory. A program is a collection of segments.
- A segment is a logical unit such as main program, procedure, function, method, object, local variables, global variables, common block, stack, symbol table, arrays etc. as shown in Fig. 1.18.
- Here, the user does not view the memory as a linear array of words. Instead, the user views the memory as a collection of variable sized segments with no necessary ordering among segments.

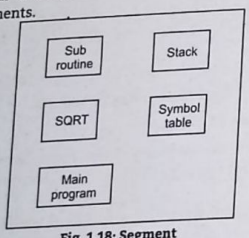


Fig. 1.18: Segment

- Each of these segments (tables, arrays, stacks, programs, subroutines, procedures functions or modules etc.) is of variable lengths.
- Elements within a segment are identified by their offset from the beginning of the segment i.e. the first statement of the program.
- Segmentation is a memory management scheme which supports user's view of Memory. A logical address space is a Collection of segments. Each segment has a name and a length.
- Address specifies both the segment name and the offset within the segment. The user specifies each address by two quantities: a segment name and an offset.

- A logical address consists of two parts: a segment number 's' and an offset into that segment 'd'. The segment number is used as an index into segment table.
- Each entry of segment table has a segment base and a segment limit.
- The offset 'd' of the logical address must be between 0 and the segment limit. If it is not, a trap is generated to the operating system. If this offset is legal, it is added to the segment base to produce the address in physical memory of the desired word.
- The segment table is thus essentially an array of base/limit register pairs.

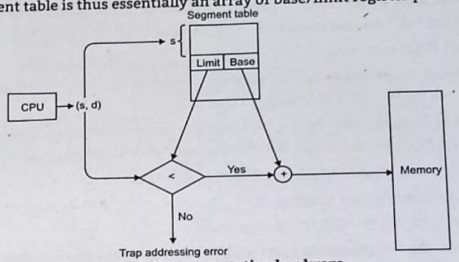


Fig. 1.19: Segmentation hardware

- As an example consider the situation shown in Fig. 1.20.
- We have five segments numbered from 0 through 4. The segments are actually stored in physical memory as shown in Fig. 1.20.

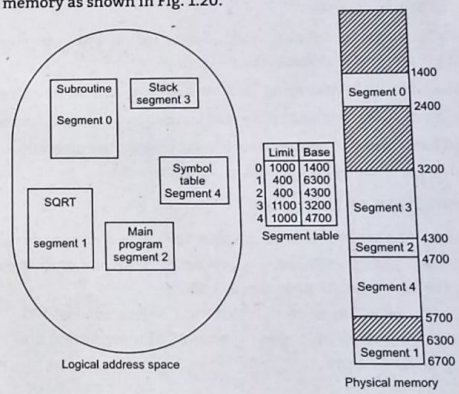


Fig. 1.20: Example of Segmentation

- The segment table has a separate entry for each segment, giving the beginning address of the segment in physical memory (the base) and the length of that segment (the limit) for example, segment 2 is 400 words long, beginning at location 4300. Thus, reference to word 53 of segment 2 is mapped on to location 4300 + 53 = 4353.
- A reference to segment 3, word 852 is mapped to 3200 (the base of segment 3) + 852 = 4052. A reference to word 1222 of segment 0 would result in a trap to the operating system, since this segment is only 1000 words long.

Advantages of Segmentation:

1. **Eliminate fragmentation:** By moving segments around, fragmented memory space can be combined into a single true area.
2. **Provides virtual memory:** By keeping only the actively used segments in main memory. The jobs total address space size may exceed the physical memory size.
3. **Segmentation allows dynamically growing segments.**
4. **Dynamic linking and loading:** At some time it links the subroutines and data area when explicitly referenced.
5. **Facilitate shared segments** (data area and procedures). For example, (square root routine).
6. **Enforced control access:** (i.e. Read, Write, Executed)

Disadvantages of Segmentation:

1. Considerable compaction overhead is incurred in order to support dynamic segment growth and eliminate fragmentation.
2. There is a difficulty in managing variable size segments on the secondary storage.
3. The maximum size of segment is limited by the size of main memory.
4. Increased hardware cost processor overhead for address mapping.
6. Increased complexity in the operating system.

1.8.1 Segmentation with Paging

- The 80386 uses segmentation with paging for memory management. In 80386, maximum number of segmentation per process is 16 KB and each segment can be as large as 4 Gigabytes. While the page size is 4 KB.
- A simple mode of Intel 80386 address translation is shown in Fig. 1.21.
- The MULTICS system solved problems of external fragmentation and lengthy search times by paging the segments.

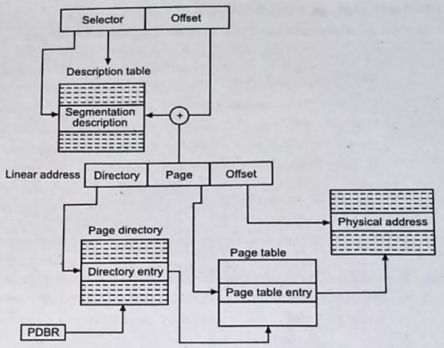


Fig. 1.21: PDBR (Page Directory Base Register)

- Solution differs from pure segmentation in that the segment-table entry contains not the base address of the segment, but rather the base address of a page table for this segment.

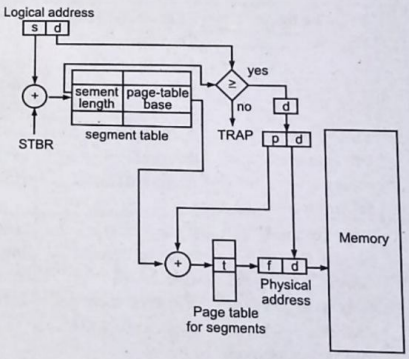


Fig. 1.22: MULTICS address translation scheme

Comparison of Memory Management Schemes:

Table 1.1: Comparison of Memory Management Schemes

Factors	Contiguous memory allocation	Paging	Segmentation
1. Memory allocation	In this, each process is contained in a single contiguous section of memory.	This scheme permits physical address space of process to be non-contiguous but make clear separation between logical memory and physical memory.	This scheme permits physical address space of process to be non-contiguous but supports user's view of memory.
2. Hardware support	Relocation and limit register.	Address mapping is done by page table.	Address mapping is done by Segment table.
3. Dynamic relocation	Yes.	Paging itself is a form of dynamic relocation.	Yes, with help of Segment table.
4. Sharing of data or code	Not possible.	Possible. Sharing of reentrant code.	Possible, easier than paging.
5. Protection	By using limit register.	By Protection bits like r/w or r and valid-invalid bit attached to each entry of page table.	By Protection bits like r/w or r attached to each entry of segment table.
6. Fragmentation	Suffers from internal and external fragmentation.	Suffers from internal fragmentation.	Suffers from external fragmentation.
7. Performance	Time does not require for mapping of logical address to physical address. (Only comparing address with limit register).	If page table is implemented in associative registers, then mapping is faster.	If segment table is implemented in fast registers then mapping is faster.

1.9 VIRTUAL MEMORY

- Virtual memory is the separation of user logical memory from physical memory.
- This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available.
- Virtual memory makes the task of programming much easier because the programmer no longer needs to worry about the amount of physical memory available.
- Virtual memory is commonly implemented by demand paging; it can also be implemented in a segmentation system.
- Virtual memory also allows files and memory to be shared by several different processes through page sharing.
- Virtual memory is a technique which allows the execution of processes that may not be completely in memory.
- The basic idea behind virtual memory is that the combined size of the program, data and stack may exceed the amount of physical memory available for it.

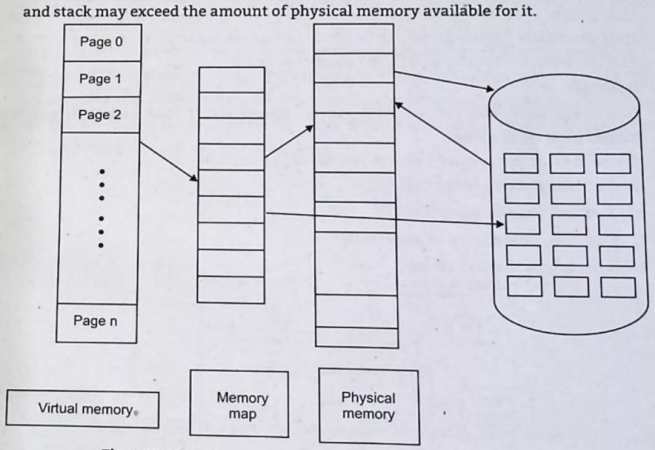


Fig. 1.23: Virtual memory that is larger than physical memory

- The operating system keeps those parts of the program currently in use in main memory and the rest on the disk.

1. Page Fault:

- The pages are mapped on to physical memory.

- The pages that mapped are shown by a cross. In the actual hardware a present / absent bit keeps track of which page are physically present in the memory.
- If any program tries to use an unmapped page, the MMU notices that the page is unmapped and causes the CPU notifies to the operating system by generating a trap interrupt. This trap is called a **page fault**.

2. **Page Tables:**
- The virtual address is split into a virtual page number (high order bits) and an offset (low order bits).
- The virtual page number is used as an index into the page table to find the entry for that virtual page.
- From the page table entry, the page frame number is found.
- The page frame number is attached to the high order end of the offset, replacing the virtual page number, to form a physical address that can be sent to the memory.
- The purpose of the page table is to map virtual pages onto page frames. Mathematically speaking, the page table is a function, with the virtual page number as argument, and the physical frame number as result.

3. **Hashing:**
- This is the technique for handling address spaces larger than 32-bits in which a hashed page table is used.
- The hash value in a hashed page table is virtual-page number.
- Each element having three fields:
  1. A virtual page number.
  2. The value of the mapped page frame.
  3. A pointer to the next element.

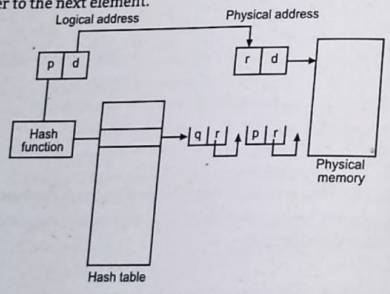


Fig.1.24: Hash Page table

1.10 THRASHING

- Removing a page from memory and then immediately needing it would require. Excessively moving the page black and forth between memory and secondary storage is called **"Thrashing"** working set model.
- Thrashing is related to the rate at which page fault occurs. If the page faults occur every time after only a few instructions, the system is said to be thrashing and naturally it decreases the performance. Because the pages are continuously moved between the main memory and disk.
- If the process not find desired page in memory, it will quickly page fault. At this point, it must replace some pages. Since all its pages are in active use, it must replace a page that will be needed again right way. Consequently, it quickly faults again and again the process continuous to fault, replacing for which it then faults and brings back in right away. This high paging activity is known as thrashing.
- Trashing results in severe performance problems. Consider the following scenario.
- The operating system monitors CPU utilization. If the CPU utilization is too low then we increase the degree of multiprogramming by introducing new process to the system. For this, a global page replacement algorithm is used and it replaces with no regard to the process to which they belong.

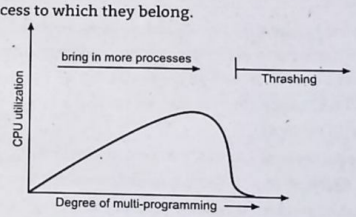
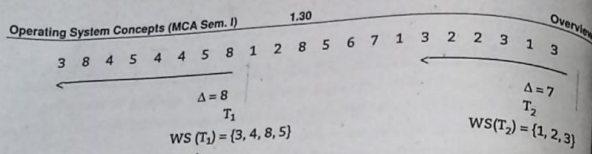


Fig. 1.25: Thrashing

Working-set Model of Thrashing:
- The working-set model is based on the assumption of locality. This model uses a parameter triangle which defines the working-set window.
- The idea is to examine the most recent triangle page references. The set of pages in the most recent triangle page references in the working-set.
- For example, given the sequence of memory references with $\Delta = 8$ memory references, then the working set at time T_1 is $WS(T_1) = \{3, 4, 8, 5\}$ and



at time T_2, the working set is

$WS(T_2) = \{1, 2, 3\}$ with $\Delta = 7$.

Summary

- Operating systems are a type of system software that allows applications to interface with computer hardware. Four major categories of operating systems are batch, time-sharing, personal computing, and dedicated.
- Memory management involves the allocation (and constant reallocation) of specific memory blocks to individual programs as user demands change. At the application level, memory management ensures the availability of adequate memory for the objects and data structures of each running program at all times.
- Contiguous and noncontiguous memory allocation is that contiguous allocation allocates one single contiguous block of memory to the process whereas, the noncontiguous allocation divides the process into several blocks and place them in the different address space of the memory.
- The address generated by the CPU is a logical address, whereas the address actually seen by the memory hardware is a physical address. Addresses bound at compile time or load time have identical logical and physical addresses.
- Virtual memory is a technique that enables us to map a large logical address space onto a smaller physical memory
- Page replacement(basic concept and replacement algorithms).
- Operating systems, demand paging (as opposed to anticipatory paging) is a method of virtual memory management. It follows that a process begins execution with none of its pages in physical memory, and many page faults will occur until most of a process's working set of pages are located in physical memory.
- Thrashing occurs when a computer's virtual memory resources become saturated, leading to a constant state of paging, to the exclusion of most application-level processing.
- In contiguous memory allocation, swapped-in processes are arranged in the originally allocated space. In non-contiguous memory allocation, swapped-in processes can be arranged in any place in the memory.

Check Your Understanding

1. CPU fetches the instruction from memory according to the value of _____.
   (a) program counter (b) status register
   (c) instruction register (d) program status word
2. Which one of the following is the address generated by CPU?
   (a) physical address (b) absolute address
   (c) logical address (d) none of the mentioned
3. Virtual memory is normally implemented by _____.
   (a) demand paging (b) buses
   (c) virtualization (d) all of the mentioned
4. Thrashing _____.
   (a) is a natural consequences of virtual memory system.
   (b) can always be avoided by swapping.
   (c) always occurs on large computers.
   (d) can be caused by poor paging algorithms.
5. Thrashing occurs when _____.
   (a) When a page fault occurs.
   (b) Processes on system frequently access pages not memory.
   (c) Processes on system are in running state.
   (d) Processes on system are in waiting state.
6. Virtual memory typically located on _____.
   (a) RAM (b) CPU
   (c) Flash card (d) Hard drive
7. Because of virtual memory, the memory can be shared among _____.
   (a) processes (b) threads
   (c) instructions (d) none of the mentioned
8. The pager concerns with the _____.
   (a) individual page of a process (b) entire process
   (c) entire thread (d) first page of a process
9. When a program tries to access a page that is mapped in address space but not loaded in physical memory, then?
   (a) segmentation fault occurs (b) fatal error occurs
   (c) page fault occurs (d) no error occurs
10. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called _____.
    (a) job queue (b) ready queue
    (c) execution queue (d) process queue

Answer Key

1. (a)	2. (c)	3. (a)	4. (d)	5. (b)	6. (a)	7.(a)	8.(a)	9.(c)	10.(b)

Practice Questions

Q.I Answer the following Questions in short.

1. What is hardware in operating system?
2. What are CPU states in operating system?
3. What is meant by memory management?
4. Enlist various memory management techniques.
5. Enlist various algorithms for page replacements.

Q.II Answer the following Questions.

1. What is operating system? Explain its characteristics.
2. Explain virtual memory management concept.
3. What is operating system? Explain its functions.
4. How to convert logical address to physical address? Explain with suitable example.
5. Compare paging, segmentation and fragmentation.

Q.III Write short note on.

1. Logical address
2. Physical address
3. Page fault
4. Demand paging
5. Page replacement
6. Virtual memory

2... Process Management and Synchronization

Objectives...

- To study Concept of Process Management and Synchronization such as PCB, Job and processor scheduling, Scheduling Concept, Process hierarchies.
- To learn about Problems of concurrent processes, Critical sections, Mutual exclusion, Synchronization and Deadlock.
- To learn about Device and File Management.

2.1 INTRODUCTION

- A process is mainly an instance of a program in execution. The execution of a process must progress in sequential order or based on some priority or algorithms. In other words, it is an entity that represents the fundamental working that has been assigned to a system.
- Process management is an integral part of any modern day Operating System (OS). The OS must allocates resources to processes, enable processes to share and exchange information, protect the resources of each process from other processes and enables synchronization among processes.

2.2 PCB (PROCESS CONTROL BLOCK)

- Process Control Block is a data structure in the operating system kernel containing the information needed to manage the scheduling of a particular process.
- The Process Control Block is also known as a Task Control Block, Entry of the process table, etc.
- It is very important for process management as the data structuring for processes is done in terms of the PCB. It also defines the current state of the operating system.

2.2.1 Structure of the Process Control Block

- The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given Fig. 2.1.

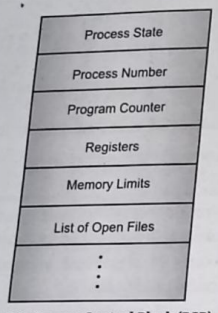


Fig. 2.1: Process Control Block (PCB)

- The following are the data items in Process Control Block:
  - **Process State**: This specifies the process state i.e. new, ready, running, waiting or terminated.
  - **Process Number**: This shows the number of the particular process.
  - **Program Counter**: This contains the address of the next instruction that needs to be executed in the process.
  - **Registers**: This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.
  - **List of Open Files**: These are the different files that are associated with the process.
  - **CPU Scheduling Information**: The process priority, pointers to scheduling queues etc. are the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.
  - **Memory Management Information**: The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.
  - **I/O Status Information**: This information includes the list of I/O devices used by the process, the list of files etc.
  - **Accounting Information**: The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

  - **Location of the Process Control Block**: The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

2.3 JOB AND PROCESSOR SCHEDULING

- Job and process are used roughly replacing each other. Much of the operating system theory and terminology was developed during a time when main action of operating system was job processing.

2.3.1 Job Scheduling

- Job scheduling is the process of allocating system resources to many different tasks by an operating system (OS). The system handles prioritized job queues that are waiting CPU time and it should determine which job to be taken from which queue and the amount of time to be allocated for the job.
- Most Operating Systems like UNIX, Windows, etc., include standard job-scheduling abilities. A number of programs including Database Management Systems (DBMS), Backup, Enterprise Resource Planning (ERP) and Business Process Management (BPM) feature specific job-scheduling capabilities as well.
- The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming.

Attributes from a Job Scheduler:

- Real-time scheduling in accordance with external, unforeseen events.
- Automated restart and recovery in case of failures.
- Notifying the operations personnel.
- Generating reports of incidents.
- Audit trails meant for regulation compliance purposes.
- In scheduling, many different schemes are used to determine which specific job to run. Some parameters that may be considered are as follows:
  - Job priority.
  - Availability of computing resource.
  - License key if the job is utilizing licensed software.
  - Execution time assigned to the user.
  - Number of parallel jobs permitted for a user.
  - Projected execution time.
  - Elapsed execution time.
  - Presence of peripheral devices.
  - Number of cases of prescribed events.
- It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

2.3.2 Processor Scheduling

- Processor scheduling is the allocation of a computer's processor power to specific tasks. The practice uses the term "scheduling" because, it assigns a specific percentage of time the processor is running to individual tasks.
- Processor scheduling is used to prevent specific tasks from controlling all of a computer's processor resources.
- Processor scheduling is also used to assign processors to different instances of Windows in virtual machines as well as to share CPU capacity with users connecting to a computer through remote desktop.

Task Priority:

- The most basic form of processor scheduling is set by assigning each individual task a priority rating in increasing order of importance of low, below normal, normal, above normal, high and real-time.
- The higher level of importance is the more processor time or usage windows will assign to the task. Tasks that have more processing time assigned to them will run and complete faster.
- The process isn't a chronological schedule like appointments on a calendar but rather a hierarchy of importance.

Core Assignment:

- Processor scheduling can be further customized through affinity settings. Affinity settings tell the computer which processors and processor cores can be used to operate any given program.
- Affinity settings are not available on a single processor with single core computers, however, multi-core processors and multi-processor computers can take advantage of this setting.
- Core assignment applies in conjunction with task prioritization. For example, with a four-core, single-processor computer you can use affinity settings to enable video editing software to use only three of the cores. If you're running a task that uses all available CPU power like exporting video, the computer will use only the three assigned cores for the process, leaving the fourth core available for handling other tasks on the computer like a Web browser. Affinity schedules the video editing task to all of the time for three of the cores and none of the time for the other one.

Foreground and Background:

- Processor scheduling is also important for your computer to handle foreground and background tasks at the same time.
- Foreground tasks are the programs that are open and you're actively using. Background tasks are the programs that run behind the scenes and handle automated tasks.
- Processor scheduling between foreground and background tasks is automated, which can prevent anti-virus software from taking over the full CPU usage when running a routine scan and rendering the system unusable until it completes.

Control Options:

- Processor scheduling options are mostly automated but can be fine-tuned in the Task Manager window. You can access both the affinity and priority settings on the Processes tab of the task manger by right-clicking on any displayed task and selecting it from the drop-down menu.
- Priority is assigned through a drop-down menu and affinity is handled through check boxes pertaining to each core. Windows Server operating systems have additional advanced processor scheduling abilities.

2.4 SCHEDULING CONCEPT

Basic terms in scheduling are:

- **Multi-programming:** A number of programs can be in memory at the same time. Allows overlap of CPU and I/O.
- **Jobs** (batch) are programs that run without user interaction.
- **Schedule:** A schedule specifies when and how many times a job is executed.
- **User** (time shared) are programs that may have user interaction.
- **Process** is the common name for both.
- **CPU - I/O burst cycle:** Characterizes process execution, which alternates, between CPU and I/O activity. CPU times are generally much shorter than I/O times.
- **Pre-emptive Scheduling:** An interrupt causes currently running process to give up the CPU and be replaced by another process.
- **Dispatcher:** The dispatcher is the module that gives control of the CPU to the process selected by the scheduler.

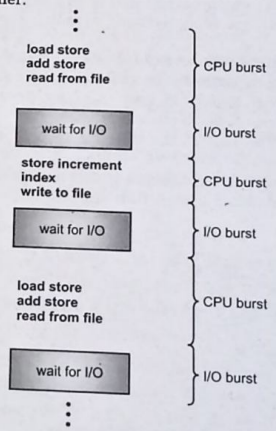


Fig. 2.2: Alternative sequence of I/O and CPU burst in Scheduling

2.4.1 Scheduling Criteria

- There are several different criteria to consider when trying to select the "best" scheduling algorithm for a particular situation and environment, including:
  - **CPU utilization:** Ideally the CPU would be busy 100% of the time, so as to waste 0 CPU cycles. On a real system CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded). Keep CPU as busy as possible.
  - **Throughput:** Number of processes completed per unit time. May range from 10/second to 1/hour depending on the specific processes.
  - **Turnaround time:** Time required for a particular process to complete, from submission time to completion. (Wall clock time)
  - **Waiting time:** This is the total time which processes spend in the ready queue waiting their turn to get on the CPU.
    - **(Load average:** The average number of processes sitting in the ready queue waiting their turn to get into the CPU. Reported in 1-minute, 5-minute, and 15-minute averages by "uptime" and "who".)
  - **Response time:** The amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment).

2.5 PROCESS HIERARCHIES

- In some computer systems when a process creates another process, then the parent process and child process continue to be associated in certain ways. The child process can itself creates more processes that form a process hierarchy.
- Modern general purpose operating systems permit a user to create and destroy processes.
- In UNIX, this is done by the fork system call, which creates a child process, and the exit system call, which terminates the current process.
- After a fork both parent and child keep running (indeed they have the same program text) and each can fork off other processes.
- A process tree results. The root of the tree is a special process created by the OS during startup.
- A process can choose to wait for children to terminate. For example, if C issued a wait() system call it would block until G finished.

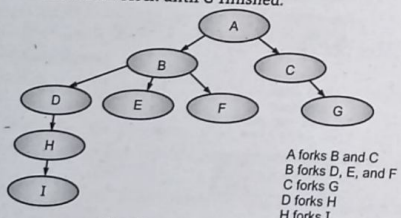


Fig. 2.3: Process Hierarchy

Process States:

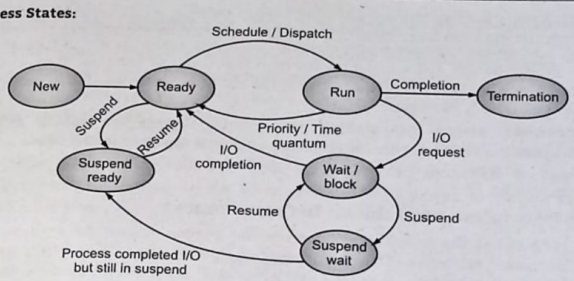


Fig. 2.4: Process State Diagram

- **New (Create):** In this step, the process is about to be created but not yet created; it is the program which is present in secondary memory that will be picked up by OS to create the process.
- **Ready:** After the creation of a process, the process enters the ready state i.e. the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue for ready processes.
- **Run:** The process is chosen by CPU for execution and the instructions within the process are executed by any one of the available CPU cores.
- **Block or Wait:** Whenever the process requests access to I/O or needs input from the user or needs access to a critical region (the lock for which is already acquired). It enters the block/wait state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state.
- **Terminated or completed:** Process is killed as well as PCB is deleted.
- **Suspend Ready:** Process that was initially in the ready state but were swapped out of main memory and placed onto external storage by scheduler are said to be in suspend ready state. The process will transition back to ready state whenever the process is again brought onto the main memory.
- **Suspend Wait or Suspend block:** Similar to Suspend Ready state but uses the process which was performing I/O operation and lack of main memory caused them to move to secondary memory. When work is finished, it may go to Suspend Ready state.

2.6 PROBLEMS OF CONCURRENT PROCESSES

- Concurrency encompasses a host of design issues, including communication among processes, sharing and competing for resources (such as memory, files, and I/O access), synchronization of the activities of multiple processes, and allocation of processor time to processes.
- Concurrency means that multiple processes or threads are making progress concurrently. While only one thread is executed at a time by the CPU, these threads can be switched in and out as required. So all the threads are executing concurrently.

2.6.1 Principles and Problems in Concurrency

- Concurrency is the interleaving of processes in time to give the appearance of simultaneous execution. Thus it differs from parallelism, which offers genuine simultaneous execution. However the issues and difficulties raised by the two overlap to a large extent:
  - Sharing global resources safely is difficult.
  - Optimal allocation of resources is difficult.
  - Locating programming errors can be difficult, because the contexts in which errors occur cannot always be reproduced easily.
- Parallelism also introduces the issue that different processors may run at different speeds, but again this problem is mirrored in concurrency because different processes progress at different rates.

Example:

- The fundamental problem in concurrency is processes interfering with each other while accessing a shared global resource. This can be illustrated with a surprisingly simple example:

```
chin = getchar();
chout = chin;
putchar(chout);
```

- Imagine two processes P1 and P2 both executing this code at the "same" time, with the following interleaving due to multi-programming:
  1. P1 enters this code, but is interrupted after reading the character x into *chin*.
  2. P2 enters this code, and runs it to completion, reading and displaying the character y.
  3. P1 is resumed, but *chin* now contains the character y, so P1 displays the wrong character.
- The essence of the problem is the shared global variable *chin*. P1 sets *chin*, but this write is subsequently lost during the execution of P2. The general solution is to allow only one process at a time to enter the code that accesses *chin*: such code is often called a **Critical Section**. When one process is inside a critical section of code, other processes must be prevented from entering that section.

2.7 CRITICAL SECTIONS

- Critical Section is the part of a program which tries to access shared resources. The critical section cannot be executed by more than one process at the same time. Operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.
- The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.

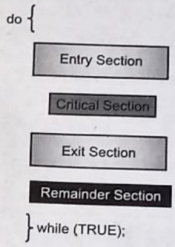


Fig. 2.5: Critical Section

- In the Fig. 2.5, the entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

Solution to the Critical Section Problem:

- The critical section problem needs a solution to synchronize the different processes. The solution to the critical section problem must satisfy the following conditions:
  1. **Mutual Exclusion**: Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.
  2. **Progress**: Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.
  3. **Bounded Waiting**: Bounded waiting means that each process must have a limited waiting time. It should not wait endlessly to access the critical section.

2.8 MUTUAL EXCLUSION

- *Mutual exclusion is in many ways the fundamental issue in concurrency. It is the requirement that when a process P is accessing a shared resource R, no other process should be able to access R until P has finished with R.*
- *There are essentially three approaches:*
  1. *Leave the responsibility with the processes themselves: This is the basis of most software approaches.*
  2. *Allow access to shared resources only through special purpose machine instructions.*
  3. *Provide support through the operating system, or through the programing language.*
- *Mutual exclusion (mutex) is a program object that prevents simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource.*
- *Only one thread owns the mutex at a time, thus a mutex with a unique name is created when a program starts. When a thread holds a resource, it has to lock the mutex from other threads to prevent concurrent access of the resource. Upon releasing the resource, the thread unlocks the mutex.*

2.9 SYNCHRONIZATION

- *Process Synchronization means sharing system resources by processes in such way that, concurrent access to shared data is handled thereby minimizing the chance of inconsistent data. Maintaining data consistency demands mechanisms to ensure synchronized execution of co-operating processes.*
- *Process Synchronization was introduced to handle problems that arise while multiple process executions.*

Types of Synchronization: *There are two types of synchronization: 1. Process Synchronization 2. Data Synchronization.*

1. ***Process Synchronization****: The simultaneous execution of multiple threads or processes to reach a handshake such that they commit a certain sequence of actions. Lock, mutex, and semaphores are examples of process synchronization.*
2. ***Data Synchronization****: Involves the maintenance of data to keep multiple copies of data coherent with each other, or to maintain data integrity. For example, database replication is used to keep multiple copies of data synchronized with database servers that store data in different locations.*

Synchronization forms the basis of the execution of multiple threads asynchronously in a multithreaded application. It provides the means to achieve the sharing of resources such as file handling, network connections and memory by coordinating threads and processes to avoid data corruption.

- The term is used in the context of multi-threaded applications where the resources to be shared across multiple threads have to be controlled, which otherwise can lead to an unpredictable and undesirable outcome. The .NET framework provides synchronization primitives using the multi-threaded applications controlled without any race conditions.
- Synchronization is designed to be cooperative, demanding that every thread follows the synchronization mechanism before accessing protected resources for consistent results.

2.10 DEADLOCK

- In general, high overall resource utilization and the possibility of parallel operation of many Input/Output devices driven by concurrent processes, contribute significantly to high performance potential of multitasking and multiprogramming systems.
- At the same time, concurrency and high resource utilization also provide the necessary conditions and the fertile ground for deadlocks.
- When several processes compete for a finite number of resources, a situation comes where a process requests a resource and the resource is not available at that time, in that case the process enters a wait state.
- It may happen that waiting processes will never again change state, because the resources that they have requested are held by other waiting processes. This situation is called a Deadlock.
- **Definition of Deadlock:** A set of processes is in a deadlock state when every process in the set is waiting for an event that can only be caused by another process in the set.

2.10.1 Principles of Deadlock

- Deadlock can be defined as "the permanent blocking of a set of processes that either compete for system resources or communicate with each other".
- In other words, when a process request resources, if the resources are not available at that time, the process enters a wait state.
- Waiting process may never again change state, because resources they have requested are held by other waiting processes. This situation is called deadlock.

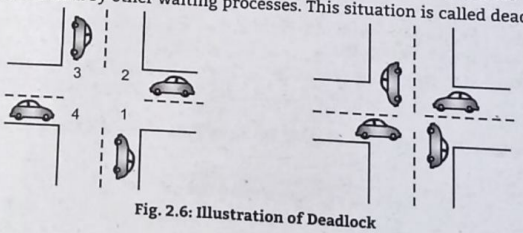


Fig. 2.6: Illustration of Deadlock

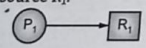
- All deadlocks involve conflicting needs for resources by two or more processes.
- A common example is the traffic deadlock.
- Fig. 2.6 shows a situation in which four cars have arrived at four-way stop intersections at approximately the same time.
- The four quadrants of the intersection are the resources over which control is needed.
- In particular, if all four cars wish to go straight through the intersection, the resource requirements are as follows:
  1. Car travelling North needs quadrants 1 and 2.
  2. Car travelling West needs quadrants 2 and 3.
  3. Car travelling South needs quadrants 3 and 4.
  4. Car travelling East needs quadrants 4 and 1.
- If all four cars arrive at about the same time, each will refrain from entering the intersection causing a deadlock.

2.10.2 System Model

- A system consists of a finite number of resources to be distributed among a number of competing processes. The resources are partitioned into several types, each of which consists of some number of identical instances.
- Main Memory space, CPU cycles, files and Input/Output devices such as printers, tape drives are examples of resource types. If a system has two CPUs then the resources type CPU has two instances.
- A process must request a resource before using it, and must release the resource after using it. A process may request as many resources as it requires carrying out its designated task.
- Obviously, the number of resources requested may not exceed the total number of resources available in the system. For example, a process cannot request three printers if the system has only two.
- Under the normal mode of operation a process may utilize a resource in only the following sequence.
  1. **Request:** If the request cannot be granted immediately, then the requesting process must wait until it can acquire the resource.
  2. **Use:** The process can operate on the resource.
  3. **Release:** The process releases the resource.

Fig. 2.7 shows the above sequence.

(a) Process P_1 is requesting resource R_1.



(b) If Resource R_1 is free, then request made by P_1 is granted. Process P_1 use resource R_1.

(c) Process P_1 releases resource R_1 and may be used by P_2.

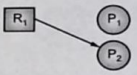


Fig. 2.7: System Model

- For example, there are two tape drives, in some system. There are two processes, each holding one more tape. Now each of the process needs one more tape drive. Now both the processes are waiting for each other to release a tape drive. This is a deadlock situation.
- A set of processes in a deadlock state when every process in the set is waiting for event that can be caused only by another process in the set. The events with which we are mainly concerned here are resource acquisition and release.

2.10.3 Deadlock Characterization

- In a deadlock, processes never finish executing and system resources are tied up, preventing either job from starting.
- Before we discuss the various methods for dealing with the deadlock problems, we shall describe features that characterize deadlocks.

Necessary Conditions of Deadlock:

- A deadlock situation can arrive if the following four conditions hold simultaneously in a system.
  - **Mutual Exclusion:** At least one resource is held in a non-sharable mode; that is only one process at a time can use the resource. Examples: monitor, printer etc. If another process requests that resource, the requesting process must be delayed until the resource has been released. Each resource is either currently assigned to exactly one process or is available.
  - **Hold and Wait:** There must available a process that is holding at least one resource and is waiting to acquire additional resources that are currently being held by another process. Process currently holding resources that were granted earlier can request for new resources.
  - **No Pre-emption:** Resources cannot be pre-empted; i.e. resource can only be released voluntarily by the process holding it, after the process has completed its task.
  - **Circular Wait:** There must exist a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, ..., P_{n-1} is waiting for a resource that is held by P_n and P_n is waiting for a resource that is held by P_0. Thus, there must be a circular chain of two or more processes, each of which is waiting for a resource held by the next member of the chain.

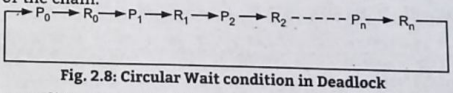


Fig. 2.8: Circular Wait condition in Deadlock

Above all four conditions must hold for a deadlock to occur.

2.10.4 Deadlock Handling Methods

- There are three ways to handle the deadlock.
  1. We can use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.
  2. We can allow the system to enter a deadlock state, detect it and recover.
  3. We can ignore the problem completely and pretend that deadlock never occurs in the system.
- **Deadlock Prevention:** Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions cannot hold.
- **Deadlock Avoidance:** It requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime. With this additional information, we can decide for each request can be satisfied or must be delayed.
- **Deadlock Detection and Recovery:** If system does not employ either a deadlock - prevention or a deadlock avoidance algorithm, then deadlock situation may occur. In this case, the system can provide an algorithm for deadlock detection and recovery.
- **Deadlock Never Occur:** In many systems, deadlock occur infrequently so no mechanism for deadlock detection and recovery is provided.
- But we may arrive at a situation where the system is in deadlock state yet has no way of recognizing what has happened. In this case, the undetected deadlock will result in the deterioration of the system performance, because resources are being held by processes that cannot run, and because more and more processes as they make requests for resources, enter a deadlock state.
- Eventually, the system will stop functioning and will need to be restarted manually. Although this method does not seem to be a viable approach to the deadlock problem, it is nevertheless used in some operating systems.

2.11 DEVICE AND FILE MANAGEMENT

- In this section, we will learn about device and file management.

2.11.1 Overview

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 devices as well as the management of all the input and output devices of the computer system such as a keyboard, magnetic tape, disk, printer, microphone, USB ports, scanner, camcorder etc.
- Device management enables organizations to protect and secure their resources and data, and from different devices.

- Using a device management provider, organization can make sure that only authorized people and devices get access to proprietary information. Similarly, device users can feel at ease accessing work data from their phone, because they know their device meet their organization's security requirements.

File Management:

- File management is one of the basic and important features of operating system. File management describes the fundamental methods for naming, storing and handling files. By using appropriate file and folder naming strategies, along with good metadata practice and catalog software. All the files with different extensions are managed by operating system.
- A file is collection of specific information stored in the memory of computer system. File management is defined as the process of manipulating files in computer system. This management includes the process of creating, modifying and deleting the files.
- The data that we work with on computers is kept in a hierarchical file system in which directories have files and sub-directories beneath them. Subdirectory is a directory present inside another directory in the file storage system. The directory base storage system ensures better organization of files in the memory of the computer system.
- The operating system's organization of our data can be enhanced by the use of cataloguing programs, which make organizing and finding image files easier than simply relying on the computer's directory structure. Another feature of catalog programs is that they can streamline backup procedures for better file protection.

Tasks Performed by File Management:

- The following are some of the tasks performed by file management of operating system of any computer system:
  1. It helps to create new files in computer system and place them at the specific locations.
  2. It helps in easily and quickly locating these files in computer system.
  3. It makes the process of sharing of the files among different users very easy and user friendly.
  4. It helps to stores the files in separate folders known as directories. These directories help users to search file quickly or to manage the files according to their types or uses.
  5. It helps the user to modify the data of files or to modify the name of the file in the directories.
- The file management function in operating system (OS) is based on the following concepts:
  1. **File Attributes:** It specifies the characteristics of the files such as type, date of last modification; size, location on the disk etc. file attributes help the user to understand the value and location of files. File attributes is one most important feature. It is uses to describe all the information regarding particular file.

2. **File Operations:** It specifies the task that can be performed on a file such as opening and closing of file.
3. **File Access permission:** It specifies the access permissions related to a file such as read and write.
4. **File Systems:** It specifies the logical method of file storage in a computer system. Some of the commonly used file systems include FAT (File Allocation Table) and NTFS (New Technology File System).

2.11.2 Techniques

Device Management Techniques:

- An operating system or the OS manages communication with the devices through their respective drivers. The operating system component provides a uniform interface to access devices of varied physical attributes.
- Main functions of device management in operating system are:
  - Keep tracks of all devices and the programs which is responsible to perform. This is called I/O controller.
  - Monitoring the status of each device such as storage drivers, printers and other peripheral devices.
  - Enforcing preset policies and taking a decision which process gets the device when and for how long. A wide range of techniques is available for implementing these polices. There are three basic techniques for implementing the policies of device management.
    - ✓ **Dedicated:** A device is assigned to a single process.
    - ✓ **Shared:** A device is shared by many processors.
    - ✓ **Virtual:** In this technique, one physical device is simulated on another physical device
  - Allocates and deallocates the device in an efficient way:
    - ✓ **Allocation:** Physically ascending a device to process with controlling unit and channel.
    - ✓ **De-allocation:** De-allocating the devices at two levels: at the process level when I/O command has been executed and the device is temporarily released, and at the job level, when the job is finished and the device is permanently released.
  - Optimizes the performance of individual devices.

File Management Techniques:

- File management is organizing and keeping track of files and folders. It helps you stay organized, so information is easy to locate and use.
- There are many benefits to maintain an organized filing system, and in the business world, this type of organizational skill can help you increase your productivity and efficiency. Since file management systems take many forms, it's important to come up with filing techniques that work for you and serve their intended purpose.
- The stack of files on your desk and determining what you need to file. Sorting through documents will give you an idea of what types of organization techniques to

institute, based on what kind of information you have to file away. Create sub-piles according to categories that you determine, such as "client files," "finance files" or "marketing files." While sorting through your documents, be sure to get rid of the files you no longer need, or put them in your company's archives.

- Using the file management tools:
  - You can save files in folders with appropriate names, so these files can be found or identified easily.
  - Create new folders quickly for recognition of information.
  - Delete unnecessary files and folders, search for files and folders.
  - Create shortcuts of files for quick access and compress files and folders to save space.

2.11.2.1 Filing Methods

- If you need to organize electronic files, perform the same activity using the electronic documents. Create electronic sub-folders and drop your documents into the folder depending on the kinds of files they are. Move old and outdated documents into your computer trash.

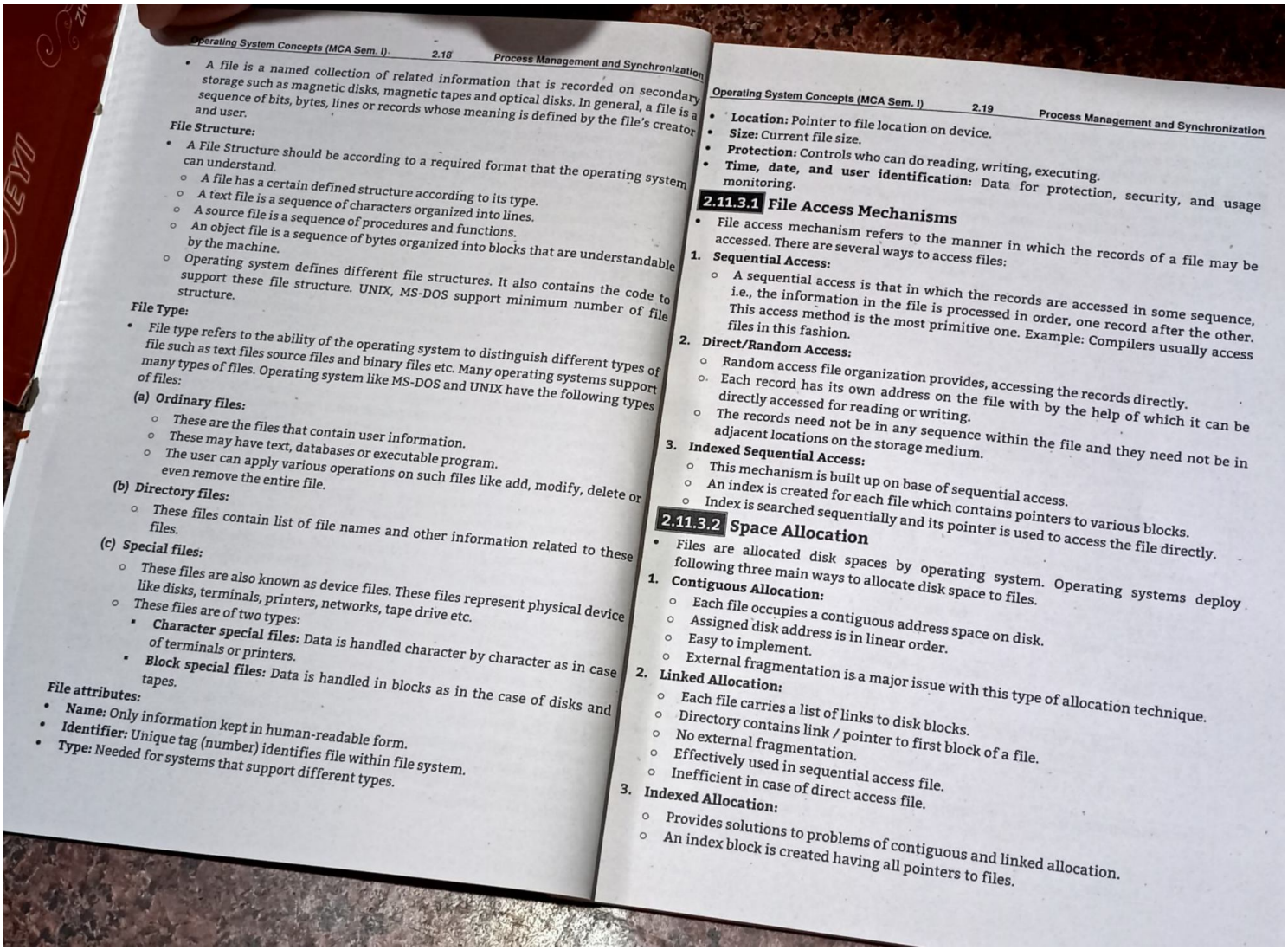
1. **Alphabetical Order:** Alphabetical filing systems are one of the basic and most preferred file organization techniques. Alphabetical filing has to do with organizing your documents according to the letters of the alphabet, and can be done in hard copy or electronically.
   In case name of more than one person starts with same letter then the second letter of the name is taken into consideration. It is a flexible method. It is used in both small and large organization.
2. **Chronological Order:** In this method, files and folders of documents are arranged in an order of their date, day, and time. In an office, several letters and documents may be received and dispatched. They all are arranged according to time and date when they were received and dispatched.
   This type of organization system may be especially handy when putting away files from meetings. For hard and electronic files, create folders that are divided into weeks, months, quarters or years – depending on your need.

2.11.2.2 Maintaining the Filing Process

- When new documents land on your desk, try filing them away as promptly as you can. This will save you time down the road from having a massive filing project. If you aren't able to file documents away immediately, set them in a filing tray in the same order you have established (alphabetically or chronologically). This will make your filing efforts, later on, be faster.

2.11.3 File Systems

- A file system is a process that manages how and where data on storage disk, typically a Hard Disk Drive (HDD), is stored, accessed and managed. It is a logical disk component that manages a disk's internal operations as it relates to a computer and is abstract to a human user.



- 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 file's 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.
  - 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:

(a) 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.

(b) Directory files:

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

(c) 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 attributes:

- **Name:** Only information kept in human-readable form.
- **Identifier:** Unique tag (number) identifies file within file system.
- **Type:** Needed for systems that support different types.

- **Location:** Pointer to file location on device.
- **Size:** Current file size.
- **Protection:** Controls who can do reading, writing, executing.
- **Time, date, and user identification:** Data for protection, security, and usage monitoring.

2.11.3.1 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:

1. **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.
2. **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.
3. **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.

2.11.3.2 Space Allocation

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

1. **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.
2. **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.
3. **Indexed Allocation:**
   - Provides solutions to problems of contiguous and linked allocation.
   - An 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.

2.11.3.3 File Systems in 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.

Table 2.1: File attributes, Types and Operations

Attributes	Types	Operations
Name	Doc	Create
Type	Exe	Open
Size	Jpg	Read
Creation Data	Xls	Write
Author	C	Append
Last Modified	Java	Truncate
protection	class	Delete
		Close

Table 2.2: Common File Types

File type	Usual extension	Function
Executable	exe, com, bin	Ready to run machine language program
Object	obj, o	Compiled, machine language, not linked
Source Code	C, java, pas, asm, perl, cc	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word Processor	wp, rtf, doc,docx	Various word processor formats
Archive	arc, zip, tar, rar	Related files grouped into one compressed file, for archiving or storage
Multimedia	mpeg, mov, mp3,mp4,avi	Binary files containing audio/video information
library	lib,dll,a	Libraries of routines for programmers
Print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
Markup	xml, html,tex	Textual data, documents

2.13.3.4 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

Operations performed on directory are:

1. **Search for a file in directory:** A directory can be searched for a particular file or for another directory. It can also be searched to list all the files with the same name.
2. **Create a file:** A new file can be created and inserted to the directory or new directory can be created keeping in mind that its name must be unique under that particular directory.
3. **Delete a directory:** The entire directory can also be deleted if it is not needed. An empty directory can also be deleted.
4. **List a directory:** List of all the files in the directory can be retrieved and also the contents of the directory entry, for each file in a list. To read the list of all the files in the directory, it must be opened and after reading the directory must be closed to free up the internal table space.
5. **Rename a file/directory:** The file or directory can be renamed in case, the content inside or the use of the file gets changed. Renaming the file or directory also changes its position inside the directory.
6. **Traverse the file system:** Allows or denies moving through a restricted folder to reach files and folders beneath the restricted folder in the folder hierarchy.

Advantages of maintaining directories are:

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

Types of directory structure:

- The directory structure has the entries of all the files related to that directory.
- The most common types of directories are:

1. **Single-Level Directory:**
   - Single level directory structure has only one directory which is called the **root** directory. The users are not allowed to create sub-directories under the root directory. All the files created by the several users are present in the root directory only.

   Disadvantages:

   (i) Naming problem: Users cannot have same name for two files.
   (ii) Grouping problem: Users cannot group files according to their need.
2. **Two-Level Directory:**
   - In this type, separate directories for each user are maintained.
   - In **Two-level directory structure,** the users create directory directly inside the root directory. But once a user creates such directory, further he cannot create any subdirectory inside that directory.
   - This structure allows to use the same name for the files but under different user directories.
   - **Path name:** A path name is the path from the root through all subdirectories to a specific file. Due to two levels, there is a path name for every file to locate that file.
   - Searching is efficient in this method.

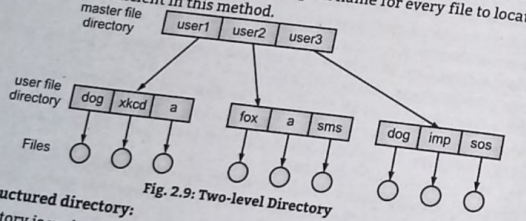


Fig. 2.9: Two-level Directory

3. **Tree-structured directory:**
   - Directory is maintained in the form of a tree.
   - In a tree directory structure, except root directory, every directory or file has only **one parent directory**. So, there is a total separation between the users which provide complete naming freedom. Here, if a user wishes to access another user's file, it has to go through two or more directories.
   - Searching is efficient and also there is grouping capability.
   - We have absolute or relative path name for a file.

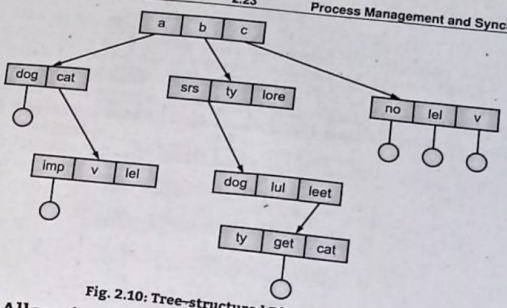


Fig. 2.10: Tree-structured Directory

2.13.3.5 File Allocation Methods

- The allocation method defines how the files are stored in the disk blocks. The direct access nature of the disks gives us the flexibility to implement the files. In many cases, different files or many files are stored on the same disk.
- There are mainly three methods of file allocation in the disk. Mainly a system uses one method for all files within the system.

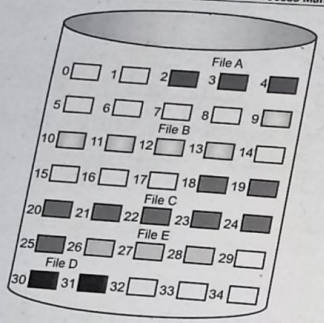
1. **Contiguous Allocation:**
   - A single contiguous set of blocks is allocated to a file at the time of file creation. Thus, this is a pre-allocation strategy, using variable size portions.
   - The file allocation table needs just a single entry for each file, showing the starting block and the length 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.

Advantages:

1. In the contiguous allocation, sequential and direct access both is supported.
2. This is very fast and the number of seeks is minimal in the contiguous allocation method.

Disadvantages:

1. 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.
2. Also, with pre-allocation, it is necessary to declare the size of the file at the time of creation.



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

Fig. 2.11: Continuous Allocation with File Allocation table

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.

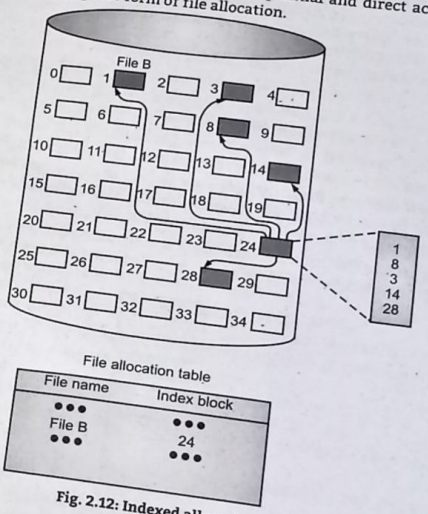
Advantages:

1. This scheme is very flexible in terms of the file size.
2. 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.
3. 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 the file.

Disadvantages:

1. Internal fragmentation exists in last disk block of the file.
2. There is an overhead of maintaining the pointer in every disk block.
3. If the pointer of any disk block is lost, the file will be truncated.
4. 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.



File allocation table

File name	Index block
File B	24

Fig. 2.12: Indexed allocation Method

Advantages:

1. Supports random access of the file.
2. Free from the problem of external fragmentation.
3. Provides fast access to the file blocks.

Disadvantages:

1. The pointer head is relatively greater than the linked allocation of the file.
2. Indexed allocation suffers from the wasted space.

3. For the large size file, it is very difficult for single index block to hold all the pointers.
4. For very small files say files that use only 2-3 blocks, the indexed allocation would keep on the entire block for the pointers which is insufficient in terms of memory utilization.

2.11.3.6 Free Space Management of Disk

- 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:** 0001101011100110001

In this vector, every bit corresponds to a particular block and 0 implies 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.

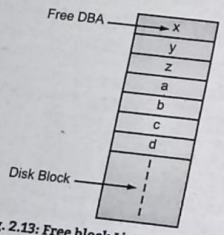


Fig. 2.13: Free block List Method

Summary

- Process Management is a comprehensive system for managing and transforming organizational operations.
- Scheduling is the task of selecting a waiting process from the ready queue and allocating the CPU.
- Scheduling algorithm is pre-emptive which is based on FCFS policy and time quantum. This algorithm is suitable for the time sharing systems.

- A deadlock state occurs when two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes.
- Device management is responsible for managing all the hardware devices of the computer system.
- File management describes the fundamental methods for naming, storing and handling files.
- A file is a collection of related information that is recorded on secondary storage.
- 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.
- Directory is also considered as a file that contains information about the other files and directories.
- The file allocation method defines how the files are stored in the disk blocks.

Check Your Understanding

1. Which module gives control of the CPU to the process selected by the short-term scheduler?
   (a) dispatcher (b) interrupt
   (c) scheduler (d) none of the mentioned
2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called _____.
   (a) job queue (b) ready queue
   (c) execution queue (d) process queue
3. Concurrent access to shared data may result in _____.
   (a) data consistency (b) data insecurity
   (c) data inconsistency (d) none of the mentioned
4. The segment of code in which the process may change common variables, update tables, write into files is known as _____.
   (a) program (b) critical section
   (c) non – critical section (d) synchronizing
5. Which of the following condition is required for deadlock to be possible?
   (a) Mutual exclusion
   (b) A process may hold allocated resources while awaiting assignment of other resources.
   (c) No resource can be forcibly removed from a process holding it.
   (d) All of the mentioned.
6. Which one of the following is the deadlock avoidance algorithm?
   (a) banker's algorithm (b) round-robin algorithm
   (c) elevator algorithm (d) Karn's algorithm
7. Which of the following is not a file allocation strategy?
   (a) Contiguous (b) Linked
   (c) Indexed (d) Bit Vector

8. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?
   (a) first-come, first-served scheduling (b) shortest job scheduling
   (c) Priority scheduling (d) none of the mentioned

Answer Key

1. (a)	2. (b)	3. (c)	4. (b)	5. (d)	6. (a)	7. (d)	8. (a)

Practice Questions

Q.I Answer the following questions in short.

1. What is indexed file allocation?
2. Which are types of directory structure?
3. What are CPU states in operating system?
4. What is Process Management in operating system?
5. What is Process Control in OS?

Q.II Answer the following questions.

1. What is Process? Explain Process Scheduling.
2. What are the different states of process?
3. What is concurrent process OS?
4. What are the principles of Concurrency in operating system?
5. What is device and file management?

Q.III Write a short note on:

1. Mutual exclusion
2. File systems
3. PCB
4. Device management
5. critical section
6. Job scheduling

3... Multiprocessor and Multicore Operating Systems

Objectives...

- To study about Multiprocessor and Multicore Operating Systems.
- To understand types of Microprocessors.
- To learn the basic Multicore Concepts.
- To know about Mobile OS and Distributed OS.

3.1 INTRODUCTION OF MULTIPROCESSOR OS

- Multiprocessor Operating System refers to the use of two or more Central Processing Units (CPU) within a single computer system.
- These multiple CPUs are in a close communication sharing the computer bus, memory and other peripheral devices.
- These types of systems are used when very high speed is required to process a large volume of data.

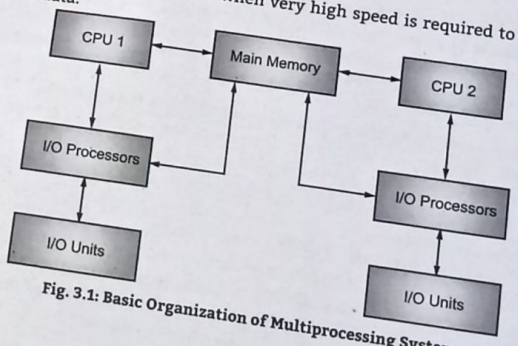


Fig. 3.1: Basic Organization of Multiprocessing System

Multiprocessor operating system (OS) is an operating system which is regularly used. It also handles system calls, do memory management, provides file system, and also manage Input/Output devices.

The basic organization of multiprocessing system is shown in Fig. 3.1.

Multiprocessing system is based on the symmetric multiprocessing model, in which each processor runs an identical copy of operating system and these copies communicate with each other.

In this system, processor is assigned a specific task. A master processor controls the system. This scheme defines a master-slave relationship.

These systems can save money in compare to single processor systems because the processors can share peripherals, power supplies and other devices.

The main advantage of multiprocessor system is to get more work done in a shorter period of time. Moreover, multiprocessor systems prove more reliable in the situations of failure of one processor.

These types of systems are used when very high speed is required to process a large volume of data.

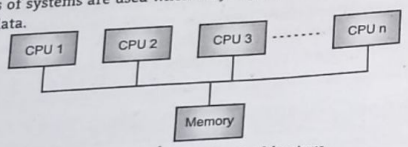


Fig. 3.2: Multiprocessing Architecture

ıtages and Disadvantages of Multiprocessor Operating System:

ıtages:

ere are multiple advantages to multiprocessor systems. Some of these are:

More Reliable Systems: In a multiprocessor system, even if one processor fails, the system will not halt. This ability to continue working despite hardware failure is known as graceful degradation. For example: If there are 5 processors in a multiprocessor system and one of them fails, then also 4 processors are still working. So the system only becomes slower and does not ground to a halt.

Enhanced Throughput: If multiple processors are working in tandem, then the throughput of the system increases i.e. number of processes getting executed per unit of time increase. If there are N processors then the throughput increases by an amount just under N.

More Economic Systems: Multiprocessor systems are cheaper than single processor systems in the long run because they share the data storage, peripheral devices, power supplies etc. If there are multiple processes that share data, it is better to schedule them on multiprocessor systems with shared data than have different computer systems with multiple copies of the data.

Disadvantages:

- There are some disadvantages as well to multiprocessor systems. Some of these are:
  1. **Increased Expense:** Even though multiprocessor systems are cheaper in the long run than using multiple computer systems, still they are quite expensive. It is much cheaper to buy a simple single processor system than a multiprocessor system.
  2. **Complicated Operating System Required:** There are multiple processors in a multiprocessor system that share peripherals, memory etc. So, it is much more complicated to schedule processes and impart resources to processes. Than in single processor systems. Hence, a more complex and complicated operating system is required in multiprocessor systems.
  3. **Large Main Memory Required:** All the processors in the multiprocessor system share the memory. So a much larger pool of memory is required as compared to single processor systems.

3.1.1 Multicore OS

- Multicore microprocessor is an interconnected set of independent processors called *cores* integrated on a single silicon chip. These processing cores communicate and cooperate with one another to execute one or more programs faster than a single core processor. Each 'core' is an independent processor and in multicore systems these cores work in parallel to speed up the processing.
- In a multicore system, a single processor can run several instructions at the same time and this, in turn, increases the overall speed of the program execution in the system. It lessens the heat generated by the CPU.
- Multicore-based processors are used in mobile devices, desktops, workstations and servers. Multicore technology is very effective in challenging tasks and applications, such as encoding, 3-D gaming and video editing.

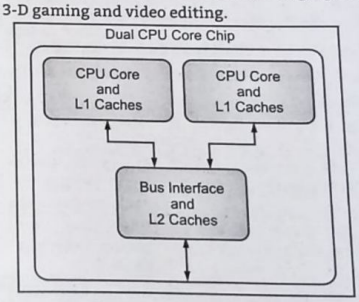


Fig. 3.3: Multicore Processor

- The performance of a multicore system totally or majorly depends on the software algorithms that are used for the implementation of the cores in the multicore system. Multicore processing can increase performance by running multiple applications concurrently.
- A simple definition of a multicore microprocessor is "It is an interconnected set of independent processing cores integrated on a single silicon chip. These processing cores communicate and cooperate with one another to execute one or more programs".
- The key points in the above definition are:
  - Set of independent processors called processing cores are integrated in a single silicon chip.
  - The cores are interconnected and communicate with one another.
  - The processing cores cooperate to execute programs.

Types of Multicore Processors:

Following are types of Multicore processors which depend on number of cores.

1. ***Homogeneous Cores:*** Homogeneous cores are functionally identical; they can perform exactly the same tasks and have exactly the same capabilities available.
2. ***Heterogeneous Cores:*** These are usually a mix of general purpose CPU architectures and DSP (Digital Signal Processing) architectures. A system uses more than one kind of processor or cores. Heterogeneous multicore processors have a mix of core types that often run different operating systems and include graphics processing units.

Advantages:

1. Since multiple CPU cores are placed on the same die of processor, so in this case, cache coherency will be higher.
2. It allows higher performance at lower energy because the cores are very energy efficient.

Disadvantages:

1. If you are using a dual-core system, then its speed should be double than the single-core but in reality, you will get 70-80% more speed only.
2. Not every Operating System supports multicore.

3.1.2 Multicore System Vs Multiprocessor System

Table 3.1: Difference between Multicore and Multiprocessor Operating System

Multicore	Multiprocessor
1. Single or multiple integrated circuit die/s.	1. Single or multiple systems.
2. Cheaper (Single CPU that does not require multiple CPU support system).	2. Expensive (Multiple separate CPU's that require system that support multiprocessors).
3. Will have less traffic (cores integrated into a single chip and will require less time).	3. Will have more traffic (distances between the two will require a longer time).
4. Does not need to be configured.	4. Needs a little complex configuration.
5. Faster running a single program.	5. Faster running multiple programs.

3.2 TYPES OF MULTIPROCESSORS

- There are mainly two types of multiprocessors i.e. Symmetric and Asymmetric multiprocessors. Details about them are as follows:
  1. **Symmetric Multiprocessors:** In these types of systems, each processor contains a similar copy of the operating system and they all communicate with each other. All the processors are in a peer to peer relationship i.e. no master - slave relationship exists between them.
     An example of the symmetric multiprocessing system is the Encore version of UNIX for the Multimax Computer.
  2. **Asymmetric Multiprocessors:** In asymmetric systems, each processor is given a predefined task. There is a master processor that gives instruction to all the other processors. Asymmetric multiprocessor system contains a master-slave relationship.
     Asymmetric multiprocessor was the only type of multiprocessor available before symmetric multiprocessors were created. Now also, this is the cheaper option.

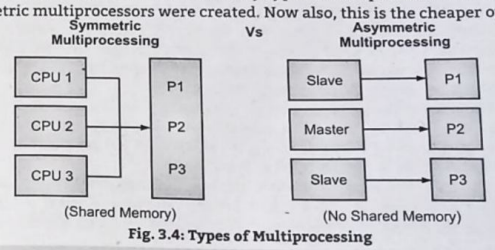


Fig. 3.4: Types of Multiprocessing

3.3 BASIC MULTICORE CONCEPTS: MEMORY SHARING STYLES

- In shared memory systems, all CPUs (or cores) can access a common memory space through a shared bus or crossbar switch.
- Prominent examples of such systems are modern multi-core CPU-based workstations in which all cores share the same main memory.

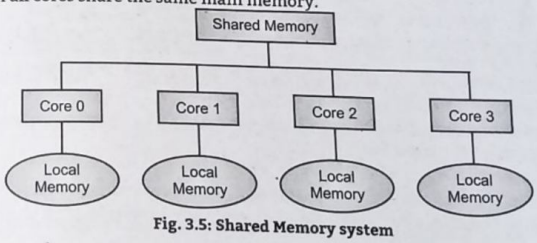


Fig. 3.5: Shared Memory system

- The main problem with shared memory is that it really does not scale very well as the number of cores (or nodes) increases.

Basic Multiprocessor System Architecture:

- Based on the vicinity and accessibility of the main memory to the processors, there are three types of multiprocessor system architectures: UMA (Uniform Memory Access), NUMA (Non-uniform Memory Access) and NORMA (no remote memory access).
- There are a number of these different styles of architecture used in a wide range of different scenarios. The main two different architectural styles employed by system designers tend to be those that follow a Uniform Memory Access Pattern or a Non-uniform memory access pattern, or UMA and NUMA respectively.

3.3.1 Uniform Memory Access (UMA)

- Uniform Memory Access (UMA) is a shared memory architecture used in parallel computers. All the processors in the UMA model share the physical memory uniformly.
- In UMA architecture, access time to a memory location is independent of which processor makes the request or which memory chip contains the transferred data. Uniform memory access computer architectures are often contrasted with non-uniform memory access (NUMA) architectures.
- In the UMA architecture, each processor may use a private cache. Peripherals are also shared in some fashion. The UMA model is suitable for general purpose and time sharing applications by multiple users. It can be used to speed up the execution of a single large program in time-critical applications.
- UMA and NUMA are shared memory models. Multiprocessors are divided among these types of categories. In UMA, Uniform Memory Access, a single memory controller is used and it is applicable for general purpose applications and time sharing applications.

3.3.2 Non-Uniform Memory Access (NUMA)

- NUMA (Non-Uniform Memory Access) is a method of configuring a cluster of microprocessors in a multiprocessing system so that they can share memory locally. It improves performance and the ability of the system to be expanded.
- NUMA is used in a symmetric multiprocessing (SMP) system. An SMP system is a "tightly-coupled," "share everything" system in which multiple processors working under a single operating system access each other's memory over a common bus or "interconnect" path.
- A limitation of SMP is that as microprocessors are added, the shared bus or data path get overloaded and becomes a performance bottleneck. NUMA adds an intermediate level of memory shared among a few microprocessors so that all data accesses don't have to travel on the main bus.
- NUMA can be thought of as a "cluster in a box." The cluster typically consists of four microprocessors (for example, four Pentium microprocessors) interconnected on a local bus (for example, a Peripheral Component Interconnect bus) to a shared memory

(called an "L3 cache ") on a single motherboard (it could also probably be referred to as a card). This unit can be added to similar units to form a symmetric multiprocessing system in which a common SMP bus interconnects all of the clusters. Such a system typically contains from 16 to 256 microprocessors. To an application program running in an SMP system, all the individual processor memories look like a single memory.

- When a processor looks for data at a certain memory address, it first looks in the L1 cache on the microprocessor itself, then on a somewhat larger L1 and L2 cache chip nearby, and then on a third level of cache that the NUMA configuration provides before seeking the data in the "remote memory" located near the other microprocessors. Each of these clusters is viewed by NUMA as a "node" in the interconnection network. NUMA maintains a hierarchical view of the data on all the nodes.
- Data is moved on the bus between the clusters of a NUMA SMP system using scalable coherent interface (SCI) technology. SCI coordinates what is called "cache coherence" or consistency across the nodes of the multiple clusters.
- SMP and NUMA systems are typically used for applications such as data mining and decision support system in which processing can be packed out to a number of processors that collectively work on a common database.

3.3.3 No Remote Memory Access (NORMA)

- No Remote Memory Access is computer memory architecture for multiprocessor systems.
- In NORMA based architectures, memory is physically partitioned across the processors. No common memory exists. Data stored at remote sites are requested via messages. Once a processor receives a request for data from a remote processor, it fetches that data from its memory and sends them to the requesting processor. Commercial examples of NORMA based architectures include the Intel iPSC and Paragon systems.
- The advantage of the NORMA model is the ability to construct extremely large configurations, which is achieved by shifting the problem to the user configuration.
- One major difference between NORMA multiprocessors and NUMA is that there is no hardware support for direct access to remote memory modules. As a result, the formers are more loosely coupled.

3.4 CACHE COHERENCE, INTER-PROCESS (AND INTERCORE) COMMUNICATION

- In this section, we will discuss the cache coherence protocols and Intercore communications to handle with the multi-cache inconsistency problems.

3.4.1 Cache Coherence

- Cache memory is the fastest system memory, required to keep up with the CPU as it fetches and executes instructions. The data most frequently used by the CPU is stored in cache memory.
- Caching (pronounced "cashing") is the process of storing data in a cache.

- Coherence defines the behavior of reads and writes to a single address location.
- Local caches pose a fundamental issue: each processor sees memory though its own cache, thus two processors can see different values for the same memory location.
- This issue is Cache Coherency, and if not solved, it prevents using caches in the processors, with heavy consequence's on performances.
- Many cache coherency protocols have been proposed, all of them designed to prevent different versions of the same cache block from being present in two or more caches (false sharing).
- All solutions are at the hardware level: the controller at each cache is capable of monitoring all memory requests on the bus coming from other CPUs, and, if necessary, the coherency protocol is activated.
- Cache coherency is tackled with different approaches in UMA and NUMA multiprocessors (and many-core processors).
  - UMA multiprocessors have a processors-to-memory pathway that favors bus interconnection, so that cache coherency is based on bus snooping
  - NUMA multiprocessors rely on complex interconnection networks for processor-to-memory communication, and the only viable solution is based on cache directories
  - Mixed mode approaches are emerging for MANY-CORE multiprocessors, with chips hosting a fairly large numbers of cores with on die shared caches.

Coherence Mechanisms:

- Two classes of protocols to ensure cache coherence:
  - **Directory based:** A central location (directory) contains the sharing status of all blocks in all caches (more scalable).
  - **Snooping:** Each cache listens on a shared bus for events regarding cache blocks (less scalable).

Coherence Protocols:

- The main problem is dealing with writes by a processor. There are two general protocols for dealing with writes to a cache:
  - **Write-through:** All data written to the cache is also written to memory at the same time.
  - **Write-back:** When data is written to a cache, a dirty bit is set for the affected block. The modified block is written to memory only when the block is replaced.
- Write-through caches are simpler, and they automatically deal with the cache coherence problem, but they increase bus traffic significantly. Write-back caches are more common where higher performance is desired. The MSI cache coherence protocol is one of the simpler write-back protocols.

3.4.2 Inter-Process (and Intercore) Communication (IPC)

- A process is a program whose execution has started but is not yet complete (i.e. a program in execution). A process can be in any of the following three basic states:

  - **Running:** The processor is executing the instructions of the corresponding process.
  - **Ready:** The process is ready to be executed, but the processor is not available for the execution of this process.
  - **Blocked:** The process is waiting for an event to occur. Examples of events are an I/O operation waiting to be completed, memory to be made available, a message to be received, etc.
- A process can be of two types:
  - Independent process.
  - Co-operating process.
- An independent process is not affected by the execution of other processes while a co-operating process can be affected by other executing processes.
- Though one can think that those processes, which are running independently, will execute very efficiently. In reality, there are many situations when co-operative nature can be utilised for increasing computational speed, convenience and modularity.
- **Inter-Process Communication (IPC)** is a mechanism which allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them.

3.4.2.1 Approaches for Inter-Process Communication

- Here, are few important methods for Inter-process Communication:

1. **Pipes:**

- Pipe is widely used for communication between two related processes. This is a half-duplex method, so the first process communicates with the second process. However, in order to achieve a full-duplex, another pipe is needed.

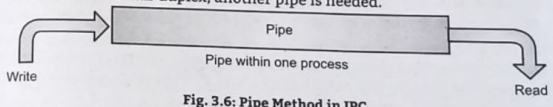


Fig. 3.6: Pipe Method in IPC

- Communication is achieved by one process writing into the pipe and other reading from the pipe. To achieve the pipe system call, create two files, one to write into the file and another to read from the file.

2. **Message Passing:**

- It is a mechanism for a process to communicate and synchronize. Using message passing, the process communicates with each other without resorting to shared variables.
- IPC mechanism provides two operations:
  - Send (message)- message size fixed or variable
  - Received (message)

3. **Message Queues:**
   - A message queue is a linked list of messages stored within the kernel. It is identified by a message queue identifier. This method offers communication between single or multiple processes with full-duplex capacity.
4. **Direct Communication:**
   - In this type of inter-process communication process, should name each other explicitly. In this method, a link is established between one pair of communicating processes, and between each pair, only one link exists.
5. **Indirect Communication:**
   - Indirect communication establishes like only when processes share a common mailbox each pair of processes sharing several communication links. A link can communicate with many processes. The link may be bi-directional or unidirectional.
6. **Shared Memory:**
   - Shared memory is a memory shared between two or more processes that are established using shared memory between all the processes. This type of memory requires to protect from each other by synchronizing access across all the processes.
7. **FIFO:**
   - A **FIFO** (First In First Out) is similar to a pipe. The principal difference is that a FIFO has a name within the file system and is opened in the same way as a regular file. This allows a FIFO to be used for communication between unrelated processes.
   - It is a full-duplex method, which means that the first process can communicate with the second process and the opposite can also happen.

Terms Used in IPC:

- The following are a few important terms used in IPC:
  - **Semaphores:** A semaphore is a signaling mechanism technique. This OS method either allows or disallows access to the resource, which depends on how it is set up.
  - **Signals:** It is a method to communicate between multiple processes by way of signaling. The source process will send a signal which is recognized by number, and the destination process will handle it.

Mechanisms for Inter-Process Communication:

- There are several mechanisms for inter-process communication. Two basic communication models for providing IPC are Shared Memory systems and Message Passing systems.
- In the former model, a part of memory is shared among the co-operating processes. The process that needs to exchange data or information can do so by writing to and reading from this shared memory.
- However, in the latter model, the co-operating processes communicate by sending and receiving messages from each other.

- The communication using message passing is very time consuming as compared to shared memory. This is because the message passing system is implemented with the help of operating system calls and thus, it requires a major involvement of the kernel. On the other hand, in shared memory systems, system calls are used only to set up the shared memory area. Once the shared area is set up, no further kernel intervention is required.

3.4.2.1.1 Shared Memory

- Shared memory is memory that may be simultaneously accessed by multiple programs with committed to provide communication among them or avoid redundant copies. Shared memory is an efficient means of passing data between programs.
- The Fig. 3.7 below shows a basic structure of communication between processes via the shared memory method.

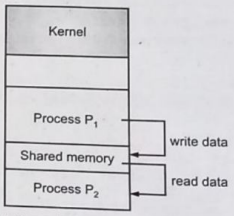


Fig. 3.7: Shared Memory Communication Model

- An operating system can implement both method of communication. First, we will discuss the shared memory methods of communication and then message passing.
- Communication between processes using shared memory requires processes to share some variable and it completely depends on the how programmer will implement it.
- One way of communication using shared memory can be imagined like this: Suppose Process P_1 and Process P_2 are executing simultaneously and they share some resources or use some information from another process.
- Process P_1 generate information about certain computations or resources being used and keeps it as a record in shared memory.
- When Process P_2 needs to use the shared information, it will check in the record stored in shared memory and take note of the information generated by Process P_1 and act accordingly. Processes can use shared memory for extracting information as a record from another process as well as for delivering any specific information to other processes.

- Shared memory is a memory shared between two or more processes. To repeat, each process has its own address space, if any process wants to communicate with some information from its own address space to other processes, then it is only possible with IPC (Inter-Process Communication) techniques. As we are already aware, communication can be between related or unrelated processes.

3.4.2.1.2 Message Passing

- In this model, data is shared by sending and receiving messages between co-operating processes, using system calls. Message Passing is particularly useful in a distributed environment where the communicating processes may reside on different, network connected, systems. Message passing architectures are usually easier to implement but are also usually slower than shared memory architectures.
- Message passing refers to a means of communication between:
  - Different Threads within a process.
  - Different Processes running on the same node.
  - Different processes running on different nodes.
- When messages are passed between two different processes, we speak of *inter-process communication*, or IPC.
- A message might contain:
  1. Header of message that identifies the sending and receiving processes.
  2. A block of data.
  3. Process control information.
- Message passing can be used as a more "process-oriented" approach to synchronization than the "data-oriented" approaches used in providing mutual exclusion for shared resources.

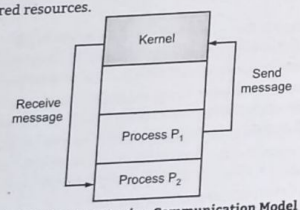


Fig. 3.8: Message Passing Communication Model

- In Message Passing systems, two system calls, send() and receive() are used. The sender process (say, P_1) sends the message to the operating system by invoking the send() system call. The operating system stores this message in the buffer area until the receive() system call is invoked by the receiver process (say, P_2).
- After that the operating system delivers this message to P_2. In case there is no message available for P_2 when it invokes the receive() system call, the operating system blocks it until some message arrives for it.

- On the other hand, if a number of messages arrive for P_2, the operating system puts them in a queue and delivers them in FIFO order upon the invocation of receive() call (one for each process) by P_2.

Ways to communicate using Message Passing:

- In the following we consider several methods for logically implementing a communication link and the send/receive operations.
- Here are different ways to communicate using Message Passing:

(a) Synchronous vs. Asynchronous:

- Messages can be sent or received either synchronously or asynchronously, also called blocking or non-blocking, respectively.
- Various design options for implementing send() and receive() calls are as follows:
  - **Blocking send:** Sending is blocked, until a message is received by receiving process or mailbox.
  - **Non-blocking send:** sending process sends the message and resumes operation.
  - **Blocking receive:** Receiver blocks until a message is available.
  - **Non-blocking receive:** The receiver retrieves either a valid message or a null.

(b) Naming, consisting of direct and indirect communication.

(i) Direct Communication: In direct communication, each process that wants to send or receive a message must explicitly name the recipient or sender of the communication. In this case, the send and receive primitives are defined as follows:

- **send (P, message):** To send a message to process P.
- **receive (Q, message):** To receive a message from process Q.

- This scheme shows the symmetry in addressing, i.e. both the sender and the receiver have to name one another in order to communicate.
- In contrast to this, asymmetry in addressing can be used, i.e., only the sender has to name the recipient; the recipient is not required to name the sender. So the send and receive primitives can be defined as follows:
  - **send (P, message):** To send a message to the process P.
  - **receive (id, message):** To receive a message from any process; id is set to the name of the process with whom the communication has taken place.

(ii) Indirect Communication: With indirect communication, the messages are sent to, and received from a mailbox or port. A mailbox can be abstractly viewed as an object into which messages may be placed and from which messages may be removed by processes. In order to distinguish one from the other, each mailbox owns a unique identification.

- A process may communicate with some other process by a number of different mailboxes. The send and receive primitives are defined as follows:
  - **send (A, message):** To send a message to the mailbox A.
  - **receive (A, message):** To receive a message from the mailbox A.

  Mailbox may be owned either by a process or by the system. If the mailbox is owned by a process, then we distinguish between the owner who can only receive from this mailbox and user who can only send message to the mailbox. When a process that owns a mailbox terminates, its mailbox disappears.
- Any process that sends a message to this mailbox must be notified in the form of an exception that the mailbox no longer exists. If the mailbox is owned by the operating system, then it has an existence of its own, i.e., it is independent and not attached to any particular process.
- The operating system provides a mechanism that allows a process to:
  (a) Create a new mailbox.
  (b) Send and receive message through the mailbox.
  (c) Destroy a mailbox.

  Since all processes with access rights to a mailbox may terminate, a mailbox may no longer be accessible by any process after some time. In this case, the operating system should reclaim whatever space was used for the mailbox.

(c) **Buffering,** consisting of capacity and messages proprieties.

(i) **Capacity Link**: A link has some capacity that determines the number of messages that can temporarily reside in it. This propriety can be viewed as a queue of messages attached to the link. Basically there are three ways through which such a queue can be implemented:

1. **Zero capacity**: This link has a message queue length of zero, i.e., no message can wait in it. The sender must wait until the recipient receives the message. The two processes must be synchronized for a message transfer to take place. The zero-capacity link is referred to as a message-passing system without buffering.
2. **Bounded capacity**: This link has a limited message queue length of n, i.e., at most n messages can reside in it. If a new message is sent, and the queue is not full, it is placed in the queue either by copying the message or by keeping a pointer to the message and the sender should continue execution without waiting. Otherwise, the sender must be delayed until space is available in the queue.
3. **Unbounded capacity:** This queue has potentially infinite length, i.e., any number of messages can wait in it. That is why the sender is never delayed. Bounded and unbounded capacity link provide message-passing system with automatic buffering.

- Zero capacity is called *no buffering* and other systems called *automatic buffering.*

(ii) **Messages:** Messages sent by a process may be one of three varieties:
(a) Fixed-sized Messages
(b) variable sized Messages
(c) Typed Messages.

If only fixed-sized messages can be sent, the physical implementation is straightforward. However, this makes the task of programming more difficult.

On the other hand, variable-size messages require more complex physical implementation, but the programming becomes simpler.

Typed messages, i.e., associating a type with each mailbox, are applicable only to indirect communication. The messages that can be sent to, and received from a mailbox are restricted to the designated type.

3.5 MOBILE OPERATING SYSTEMS

- A Mobile Operating System (mobile OS) is an Operating System built exclusively for a mobile device, such as a smartphone, Personal Digital Assistant (PDA), tablet PCs (Personal computers).
- This the master control program in a **smartphone** or tablet.
- Popular mobile operating systems are Apple's iOS (iPhone, iPad, iPod touch), Google's Android, Microsoft's Windows 10 and BlackBerry's QNX.

3.5.1 Concept, Need and Features

- A Mobile Operating System (OS) is software that allows smartphones, tablet PCs (personal computers) and other devices to run applications and programs.
- A Mobile OS typically starts up when a device power on, presenting a screen with icons or tiles that present information and provide application access.
- A Mobile OS is built exclusively for a mobile device, such as a smartphone, Personal Digital Assistant (PDA), tablet or other embedded mobile OS.
- Popular mobile operating systems are Android, Symbian, iOS, BlackBerry OS and Windows Mobile.
- A Mobile OS is responsible for identifying and defining mobile device features and functions, including keypads, application synchronization, email, thumbwheel and text messaging.
- A Mobile OS is similar to a standard OS (like Windows, Linux, and Mac) but is relatively simple and light and primarily manages the wireless variations of local and broadband connections, mobile multimedia and various input methods.
- The mobile operating system manages the hardware and makes it possible for smartphones, tablets, and wearables to run apps and other programs in a user-friendly way.

- Mobile phones are the most popular device for communication today. Every mobile requires some type of mobile operating system as a platform to run the other services and being easy for the users to use the services like voice calling, messaging service, camera functionality, Internet facilities and so on.
- The previous mobile operating systems were simple and were unable to provide an effective interface; therefore the capabilities of the phones they supported were limited.
- However, modern smart phones are laced with most advanced features of a full-fledged computer which includes high-speed Central Processing Units (CPU) and Graphics Processing Unit (GPU), large storage space, multitasking, high-resolution screens and cameras with clarity, multipurpose communication hardware and so on.

Messaging in Mobile OS:

- Following example shows how text messaging works on a Mobile OS:
  - A mobile application allows a user to read and write a message for delivery to a mobile device through radio signal waves. After the device receives the message signals, the device notifies the mobile OS, which stores the message and notifies the messaging application.
  - The user reads the message and responds with a reply message.
  - The OS uses the hardware antenna to transmit the message.

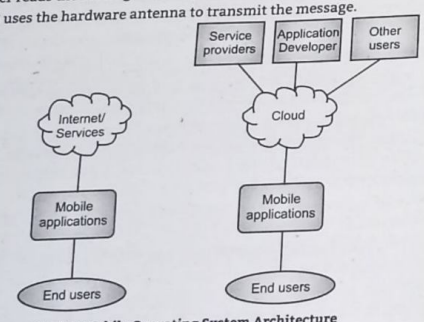


Fig. 3.9: Mobile Operating System Architecture

- With the exception of Android (developed by Google), Mobile Operating Systems are developed by different mobile phone manufacturers, including Nokia (Symbian, MeeGo, Maemo); Apple (Apple iOS); Research In Motion (RIM) (BlackBerry OS); Microsoft (Windows Mobile, Windows Phone) and Samsung (Palm WebOS and bada). Android, LiMo, Maemo, Openmoko and Qt Extended (Qtopia) are based on the Linux open source OS.

Features:

- Mobile OS has following features:
  - **Multitasking:** Multitasking is mostly a new feature of mobile operating systems. Apple's updated iOS allows for multiple apps at one time and Android's latest Jellybean update lets users multi-task, too. With devices more and more being used as extensions of offices and workspaces, it makes sense to make a mobile operating system that allows us to run as many things as we should want from our laptops.
  - **Scheduling:** All Smartphones are embedded with an operating system which enables the operation of software applications. In addition to the principle features like phone calls and messaging, you can send e-mails, manage your personal and office documents, and visit websites for searching information, play online games, and read news. It also allows sharing and downloading of documents and applications.
  - **Memory Allocation:** Mobile operating system facilitates the sharing of processing and memory resources among multiple applications. Additionally, it allows users to switch between the active applications.
  - **File System Interface:** Some operating system has a file system that is compatible with Microsoft Windows operating system. e.g. Symbian.
  - **Keypad Interface:** You can get a keyboard like the one that you use with your computer. You can use a physical keyboard to type on or a touch screen like an iPhone which is operated by software.
  - **I/O Interface:** This OS Provides a standard I/O interface. Users have the ability to add applications and upgrade features and services on their mobile phones.
  - **Protection and Security:** The operating system must enforce security policies regarding information on interconnected systems.
  - **Multimedia features:** Smartphones are also featured with built-in digital camera and a sound recorder. Apart from taking still pictures, you can also record video clips. These features allow you to share the Multi-Media Messages (MMS) with other Smartphones via email, Bluetooth or infrared with or without the help of additional software.

3.5.2 Types of Mobile OS

- Some of the popular mobile operating systems are:

iOS:

- Apple's very own operating system, iOS runs the company's iPod, iPad, iPhone, and Apple Watch devices. It is only available on Apple products and responds to commands from the user's fingertips. iOS features iTunes for music and the App Store for everything else.
- One of the benefits of the Apple operating system is its built-in video chatting and Apple Music capabilities.

- The iOS user interface is based on the concept of direct manipulation, using multi-touch gestures. Interface control elements consist of sliders, switches, and buttons. Interaction with the OS includes gestures such as swipe, tap, pinch, and reverse pinch, all of which have specific definitions within the context of the iOS operating system and its multi-touch interface.
- In iOS, there are four abstraction layers: the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch layer.

Android:

- Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets.
- Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input.

Windows:

- Windows Mobile is an operating system used in various mobile phones and Smartphones. It encompasses the entire software stack from the kernel to the application interface. This operating system is compatible with the Microsoft Office suite of programs.
- Windows Mobile is best described as a subset of platforms based on a Windows CE. Currently, Pocket PC (now called Windows Mobile Classic), Smartphone (Windows Mobile Standard), and Pocket PC Phone Edition (Windows Mobile Professional) are the three main platforms under the Windows Mobile umbrella.

BlackBerry OS:

- BlackBerry OS is a proprietary mobile operating system developed by BlackBerry Ltd for its BlackBerry line of smartphone handheld devices. The operating system provides multitasking and supports specialized input devices that have been adopted by BlackBerry Ltd. for use in its handhelds, particularly the track wheel, trackball, and most recently, the trackpad and touchscreen.
- The BlackBerry operating system is well-liked for its enhanced security and safety measures.

Symbian OS (Nokia):

- Symbian is a mobile operating system (OS) targeted at mobile phones that offers a high-level of integration with communication and Personal Information Management (PIM) functionality.
- Symbian OS combines middleware with wireless communications through an integrated mailbox and the integration of Java and PIM functionality (agenda and contacts). Nokia has made the Symbian platform available under an alternative, open and direct model, to work with some OEMs and the small community of platform development collaborators. Nokia does not maintain Symbian as an open source development project.

- Symbian operating system supports multitasking and multithreading. Many processes can run concurrently, they can communicate with each other and utilize multiple threads that run internal to each process. It facilitates good support for graphics and data management.

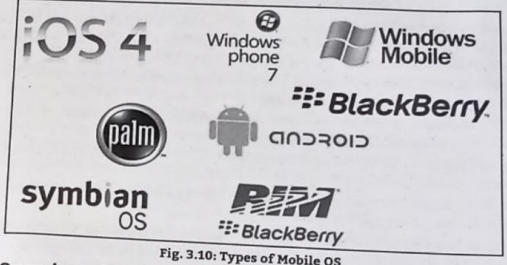


Fig. 3.10: Types of Mobile OS

3.5.3 Overview of Android

- Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.
- Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.
- The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.
- On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 Jelly Bean. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

3.5.3.1 Features of Android

- Android is a powerful operating system competing with Apple 4GS and supports great features. Some of them as follow:
  - **Beautiful UI:** Android OS basic screen provides a beautiful and intuitive user interface.
  - **Connectivity:** Supports large group of networks like: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

- **Battery Swap and Storage:** SQLite, a lightweight relational database, is used for data storage purposes. Android phones hold individual hardware capacities. Google's OS makes it possible to shift and improve the battery which does not hold a charge for a long time. For expandable storage, Android phones have SD card slots.
- **Media support:** Include support for large number of media formats for Images, Audio as well as for Video, like: H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
- **Messaging:** Supports for sending or receiving Short Message Services (SMS) and Multimedia Message Services (MMS).
- **Web browser:** Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
- **Multi-touch:** Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
- **Multi-tasking:** User can jump from one task to another and same time various applications can run simultaneously.
- **Resizable widgets:** Widgets are resizable, so users can expand them to show more content or shrink them to save space. Apps are flexible, but sometimes we need data at a sight instead of having to open an app and wait for it to load. Android widgets allow you to see the weather apps, music and also remind you of the forthcoming meetings or deadlines by a notification on the screen.
- **Multi-Language:** Supports single direction and bi-directional text.
- **GCM:** Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
- **Wi-Fi Direct:** A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
- **Android Beam:** A popular NFC (Near Field Communication)-based technology that lets users instantly share, just by touching two NFC-enabled phones together. It is the short-range wireless connectivity. It is supported by several Android devices. It permits electronic devices to communicate quickly across short distances. The main aim of NFC is to perform the payment option quickly than carrying cash and cards.

3.5.3.2 Applications of Android

- Android applications are usually developed in the Java language using the Android Software Development Kit.
- Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play, SlideME, Opera Mobile Store, Mobango, F-droid and the Amazon Appstore.

- Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast.
- Now we are using Android to achieve the following purposes:
  - **Market Sharing:** In the current era, there are lots of android users. Using android development, we can develop our own application and we can publish that application on Google Play.
  - **API:** Android provides the Application Programming Interface (API) to develop the android application. Using this API, we can develop full featured application and after completion this app, we can publish this app on Google play and can earn some money.
  - **Open Platform:** Android provides the open platform facility. Android application can be developed using any kind of operating system.
  - **Compatibility:** Android provides cross-platform means android application can run on any type screen, size and resolutions including mobile phones, tablets etc. Google provides the facility to run the application only on the compatible devices such as mobile phones or tablets.
  - **Geo-location and social networking:** Using android application, users are able to share their location via social networking such as Facebook or Twitter.

3.5.4 Applications of Mobile OS

- A mobile operating system, also called a *mobile OS*, is an operating system that is specifically designed to run on mobile devices such as,
  - Mobile phones
  - Smartphones
  - PDAs
  - Tablet computers and other handheld devices.
- The operating system is responsible for determining the functions and features available on your device, such as thumb wheel, keyboards, WAP, synchronization with applications, email, text messaging and more. The Mobile OS will also determine which third-party applications (mobile apps) can be used on your device.
- Linux or Windows operating system controls your desktop or laptop computer, a mobile operating system is the software platform on top of which other programs can run on mobile devices. The operating system is responsible for determining the functions and features available on your device, such as thumb wheel, keyboards, WAP, synchronization with applications, email, text messaging and more. The mobile OS will also determine which third-party applications (mobile apps) can be used on your device.

3.6 DISTRIBUTED OPERATING SYSTEMS

- Distributed Operating System is one of the important types of operating system.

Definitions:

- A distributed system contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software.

OR

- A distributed operating system is an operating system which manages a collection of independent computers and makes them appear to the users of the system as a single computer.

OR

- A distributed operating system is one that looks to its users like a centralized OS but runs on multiple, independent CPUs.
- 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 as loosely coupled systems or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

Examples:

- Users sharing a processor pool. System dynamically decides where processes are executed.
- Distributed Banking.

3.6.1 Concept, Need and Features

- The concept of distributed computing is the most efficient way to achieve the optimization. A distributed system consists of multiple computers that communicate through a computer network. It deals with hardware and software systems that contain more than one processing / storage and run in concurrently.
- Thus distributed computing has the chance to lead into new age in terms of computer paradigm, resources sharing pattern and online collaboration. Grid computing and cloud computing are form of distributed computing.

Features/goals of Distributed Operating System

- There are various important goals that must be met to build a distributed system worth the effort. A distributed system should easily connect users to resources, it should hide the fact that resources are distributed across a network, must be open, and must be scalable.

1. **Connecting Users and Resources:** The main goal of a distributed system is to make it easy for users to access remote resources, and to share them with other users in a controlled manner. Typical examples of Resources are printers, storage facilities, data, files, web pages, and networks.
2. **Transparency:** An important goal of a distributed system is to hide the fact that its process and resources are physically distributed across multiple computers. A distributed system that is capable of presenting itself to users and applications such that it is only a single computer system is called transparent. Transparency can be of various types like access, location, concurrency, replication, etc.
3. **Openness:** Another important goal of distributed systems is openness. Openness is concerned with extensions and improvements of distributed systems.
   - Detailed interfaces of components need to be published.
   - New components have to be integrated with existing components.
   - Differences in data representation of interface types on different processors (of different vendors) have to be resolved.
4. **Scalability:** It is inevitable that a distributed system will grow with time since expanding the network by adding new machines or interconnecting two networks together is common place. Therefore, a good distributed file system should be designed to easily cope with the growth of nodes and users in the system. That is, such growth should not cause serious disruption of service or significant loss of performance to users. In short, a scalable design should withstand high service load, accommodate growth of the user community and enable simple integration of added resources.
5. **Reliability:** The main goal of building distributed systems was to make them more reliable than single processor systems. The idea is that if some machine goes down, some other machine gets used to it.
6. **Performance:** The performance of a file system is usually measured as the average amount of time needed to satisfy client requests. In centralized file systems, this time includes the time for accessing the secondary storage device on which the file is stored and the CPU processing time. In a distributed file system, however, this time also includes network communication overhead when the accessed file is remote. Although acceptable performance is hard to quantify, it is desirable that the performance of a distributed file system should be comparable to that of a centralized file system. Users should never feel the need to make explicit file placement decisions to improve performance.

3.6.2 Examples of Distributed OS with brief Introduction

1. **Networks:** The earliest example of a distributed system happened in the 1970s when Ethernet was invented and LAN (Local Area Networks) were created. For the first time computers would be able to send messages to other systems with a local IP address. Peer-to-peer networks evolved and e-mail and then the Internet as we know it continue to be the biggest, ever growing example of distributed systems. As the internet changed from IPv4 to IPv6, distributed systems have evolved from "LAN" based to "Internet" based.

2. **Telecommunication networks**: Telephone and cellular networks are also examples of distributed networks. Telephone networks have been around for over a century and it started as an early example of a peer to peer network. Cellular networks are distributed networks with base stations physically distributed in areas called cells. As telephone networks have evolved to VOIP (Voice Over IP), it continues to grow in complexity as a distributed network.
3. **Distributed Real-time Systems**: Many industries use real-time systems that are distributed locally and globally. Airlines use flight control systems, Uber and Lyft use dispatch systems, manufacturing plants use automation control systems, logistics and e-commerce companies use real-time tracking systems.
4. **Parallel Processing**: There used to be a distinction between parallel computing and distributed systems. Parallel computing was focused on how to run software on multiple threads or processors that accessed the same data and memory. Distributed systems meant separate machines with their own processors and memory. With the rise of modern operating systems, processors and cloud services these days, distributed computing also encompasses parallel processing.
5. **Distributed Artificial Intelligence**: Distributed Artificial Intelligence is a way to use large scale computing power and parallel processing to learn and process very large data sets using multi-agents.
6. **Distributed Database Systems:**
   - A distributed database is a database that is located over multiple servers and/or physical locations. The data can either be replicated or duplicated across systems.
   - Most popular applications use a distributed database and need to be aware of the homogeneous or heterogeneous nature of the distributed database system.
   - A homogeneous distributed database means that each system has the same database management system and data model. They are easier to manage and scale performance by adding new nodes and locations.
   - Heterogeneous distributed databases allow for multiple data models, different database management systems. Gateways are used to translate the data between nodes and usually happen as a result of merging applications and systems.

Advantages and Disadvantages of Distributed Operating System

Advantages:

- Distributed operating systems provide the following advantages:
  1. With resource sharing facility, a user at one site may be able to use the resources available at another.
  2. Speedup the exchange of data with one another via electronic mail.
  3. Failure of one site in a distributed system doesn't affect the others, the remaining sites can potentially continue operating.

4. Better service to the customers.
5. Reduction of the load on the host computer.
6. Reduction of delays in data processing.

Disadvantages:

- Disadvantages of distributed operating system are:
  1. Lack of simplicity.
  2. Lack of accurate global state information.
  3. Atomic transactions.
  4. Communication protocol overhead.
  5. Security problem i.e. easy access also applies to secret data.
  6. Process and data migration.

Distributed Operating System Architecture:

In a distributed system, the following things happen:

- All hardware and software components are located remotely; they coordinate and communicate with each other by passing the messages.
- Resource sharing is the most important aspect of a distributed system; resources are managed by the servers and clients use these resources.
- A distributed operating system runs on a number of independent sites, those are connected through a communication network, but users feel it like a single virtual machine and runs its own operating system.
- A Distributed Operating System (DOS) is an operating system that is built, from the ground up, to provide distributed services. As such, a DOS integrates key distributed services into its architecture.

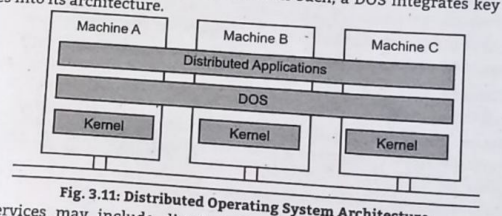


Fig. 3.11: Distributed Operating System Architecture

- These services may include distributed shared memory, assignment of tasks to processors, masking of failures, distributed storage, inter-process communication, transparent sharing of resources, distributed resource management, etc.
- A key property of a distributed operating system is that it strives for a very high level of transparency, ideally providing a single system image.
- That is, with an ideal DOS users would not be aware that they are, in fact, working on a distributed system. Distributed operating systems generally assume a homogeneous multicomputer. They are also generally more suited to LAN environments than to wide-area network environments.

- In the earlier days of distributed systems research, distributed operating systems where the main topic of interest. Most research focused on ways of integrating distributed services into the operating system or on ways of distributing traditional operating system services.
- Currently however, the emphasis has shifted more toward middleware systems. The main reason for this is that middleware is more flexible and is more suitable for heterogeneous and wide-area multi computers.

3.6.3 Applications of Distributed OS

- The various applications of distributed operating system are as follows:

1. **Telecommunication networks:**
   - Phone networks and cellular networks.
   - PC networks like the web.
   - Wireless sensor networks.
   - Routing algorithms.
2. **Network applications:**
   - Web and P2P networks.
   - Remarkably multi-player web-based games and virtual communities.
   - Distributed database management systems and distributed databases.
   - Network file systems.
   - Distributed info processing systems like banking systems and airline reservation systems.
3. **Real time process control:**
   - Aircraft control systems.
   - Business control systems.
4. **Parallel computation:**
   - Systematic computing, including cluster computing and grid computing and varied volunteer computing projects; see the list of distributed computing projects.
   - Distributed rendering in PC graphics.

Summary

- A Multiprocessing Operating System (OS) is one in which two or more central processing units (CPUs) control the functions of the computer. Each CPU contains a copy of the OS, and these copies communicate with one another to coordinate operations.
- A processor that has more than one core is called Multicore Processor while one with single core is called Unicore Processor or Uniprocessor. Multicore systems support Multi-threading and Parallel Computing.

- Uniform Memory Access (UMA) is a shared memory architecture used in parallel computers. All the processors in the UMA model share the physical memory uniformly.
- Non-Uniform Memory Access (NUMA) is a computer memory design used in multiprocessing, where the memory access time depends on the memory location relative to the processor.
- The UMA (shared memory) model uses one or two memory controllers. As against, NUMA can have multiple memory controllers to access the memory.
- In NORMA architecture, the address space globally is not unique and the memory is not globally accessible by the processors.
- In computer architecture, cache coherence is the uniformity of shared resource data that ends up stored in multiple local caches.
- A process is a program under execution or we can say an executing set of machine instructions. It can be either a system process executing the system's code or a user process executing the user's code.
- The co-operating processes require some mechanism to communicate with each other. One such mechanism is Inter-Process Communication (IPC) - a facility provided by the operating system. Two basic communication models for providing IPC are shared memory systems and message passing systems.
- In shared memory systems, a part of memory is shared among the cooperating processes. The processes that need to exchange data or information can do so by writing to and reading from this shared memory.
- In message passing systems, cooperating processes communicate by sending and receiving messages from each other. The system calls, send() and receive() are used to send and receive messages, respectively.
- A Mobile Operating System (OS) is software that allows smartphones, tablet PCs and other devices to run applications and programs.
- Distributed Operating System is a model where distributed applications are running on multiple computers linked by communications.

Check Your Understanding

1. The process is _____.
   (a) an instance of a program in execution
   (b) a program only
   (c) A processor state
   (d) none of the above
2. IPC stands for _____.
   (a) Internal program controller (b) Internal process control
   (c) Inter-process communication (d) None of the above
3. Which of the following two operations are provided by the IPC facility?
   (a) write & delete message (b) delete & receive message
   (c) send & delete message (d) receive & send message

4. Telecommunication Networks is application of _____ operating system.
   (a) Batch processing (b) Time sharing
   (c) Distributed (d) Real time
5. Which mobile OS is only available on Apple products?
   (a) iOS (b) Android
   (c) Symbian (d) Windows

Answer Key

1. (a)	2. (c)	3. (d)	4. (c)	5. (a)

Practice Questions

Q.I Answer the following Questions in short.

1. What is message passing in IPC?
2. What is shared memory in IPC?
3. Define Distributed Operating System.
4. What does mean by Mobile operating system.
5. Which are types of multiprocessors?

Q.II Answer the following Questions.

1. Write difference between Multicore and Multiprocessing system.
2. Explain the different types of operating systems.
3. Explain in detail different types of Mobile operating system.
4. Describe examples of Distributed Operating System.
5. What is Multiprocessing? Explain with its advantages and disadvantages.
6. Explain in detail the Cache coherence.

Q.III Write a short note on:

1. Multicore Operating system.
2. Uniform Memory Access (UMA).
3. Non-Uniform Memory Access (NUMA).
4. No Remote Memory Access (NORMA).
5. Android OS.

■■■

4... Real Time OS

Objectives...

- To learn about Real Time Operating System.
- To study about Embedded Operating System.

4.1 INTRODUCTION OF REAL TIME SYSTEM

- Today's users demand quick responses in order to make immediate decisions. These decisions and answers must be accurate and based on facts. In the business environment, a real-time operating system enables you use incoming data and information to support fast and knowledgeable decision-making.
- Before understanding Real Time Operating Systems few concepts need to be understood. Let us start with basic terminologies of real time operating system.

Terms related to Real time system:

- **System:** A system is something that solves a problem. It can be an intellectual unit that can manage various functions in a coherent way.
- **Task:** A set of related jobs that jointly provide some system functionality.
- **Multitasking:** Performing multiple tasks at same time is known as multitasking.
- **Job:** A job is a small piece of work that can be assigned to a processor and may or may not require resources.
- **Release time of a job:** It is the time at which job becomes ready for execution.
- **Execution time of a job:** It is the time taken by job to finish its execution.
- **Response time of a job:** It is the length of time from release time of a job to the instant when it finishes.
- **Deadline of a job:** It is the time by which a job should finish its execution. Deadline is of two types: Absolute deadline and Relative deadline.
  - Maximum allowable response time of a job is called its **Relative deadline**.
  - **Absolute deadline** of a job is equal to its relative deadline plus its release time.
- **Active Resources:** Processors are also known as *active resources*. They are essential for execution of a job. A job must have one or more processors in order to execute and proceed towards completion. Example: computer, transmission links.

- **Passive Resources:** Resources are also known as *passive resources*. A job may or may not require a resource during its execution. Example: Memory, mutex.
- Two resources are identical if they can be used interchangeably else they are heterogeneous.

Real time Operating System (RTOS)

- "RTOS" or "real time operating system," which is responsible for everything from scheduling tasks to enabling high-level languages like C and Python.
- Real time system means that the system is subjected to real time, i.e., response should be guaranteed within a specified timing constraint or system should meet the specified deadline. For example: Flight control system, Real time monitors etc.
- A Real Time Operating System is the type of operating system that is designed to serve real time applications or embedded applications. It is necessarily able to process input data without any delay. The measure of processing time requirements is in tenths of seconds or shorter.

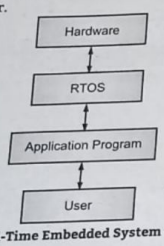


Fig. 4.1: Real-Time Embedded System with RTOS

Reference model of real time system:

- Our reference model is characterized by three elements:
  1. **A workload model:** It specifies the application supported by system.
  2. **A resource model:** It specifies the resources available to the application.
  3. **Algorithms:** It specifies how the application system will use resources.

Types of Operating systems:

- Operating systems are there from the very first computer generation and they keep evolving with time. In this chapter, we will discuss some of the important types of operating systems which are most commonly used.

1. Batch Operating System:

- 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.

- **Advantages of Batch Operating systems:**
  1. Processors of the batch systems are aware of the time duration of the job even when it is present in the queue.
  2. There is very less idle time of the batch system.
  3. Batch systems can be shared by multiple users.
  4. It enables us to manage the efficiently large load of work.
- **Disadvantages of Batch Operating Systems:**
  1. Lack of interaction between the user and the job.
  2. CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
  3. Difficult to provide the desired priority.

2. Time-sharing Operating System:

- 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 Time-sharing operating systems:**
  1. Provides the advantage of quick response.
  2. Avoids duplication of software.
  3. Reduces CPU idle time.
- **Disadvantages of Time-sharing operating systems:**
  1. Problem of reliability.
  2. Question of security and integrity of user programs and data.
  3. Problem of data communication.

3. **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 as **loosely coupled systems** or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.
   - **Advantages of Distributed Systems:**
     1. A user at one site may be able to use the resources available at another with resource sharing facility.
     2. Speedup the exchange of data with one another via electronic mail.
     3. If one site fails in a distributed system, the remaining sites can potentially continue operating.
     4. Better service to the customers.
     5. Reduction of the load on the host computer.
     6. Reduction of delays in data processing.
   - **Disadvantages of Distributed Systems:**
     1. If the main network fails, this will stop the complete communication.
     2. They are very expensive.
     3. The underlying software is highly complex.
4. **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.
   - **Advantages of Network Operating Systems:**
     1. Centralized servers are highly stable.
     2. Security is server managed.
     3. Upgrades to new technologies and hardware can be easily integrated into the system.
     4. Remote access to servers is possible from different locations and types of systems.
   - **Disadvantages of Network Operating Systems:**
     1. High cost of buying and running a server.
     2. Dependency on a central location for most operations.
     3. Regular maintenance and updates are required.

5. **Real Time System:**
   - 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 systems:
     1. **Hard real time systems:** These types of systems guarantee the execution of critical systems on time. These systems are very restrictive in terms of time constraint. These operations always have high priority over other tasks.
     2. **Soft real time systems:** These are less restrictive in terms of time constraints. Where a critical real-time tasks gets priority over the other tasks and when high priority tasks are finished, these operations would start executing once again.

4.1.1 Use of Real Time OS

Why use a real-time operating system?

- RTOS benefits are more apparent in more complex software solutions. Some of the reasons for using an RTOS include:
  - **Extensibility**: This is when easier maintenance is required. For instance, abstracting timing reliance and task-based designs, cause fewer interdependencies between the modules.
  - **Priority-based scheduling**: This is when the separation of non-critical processes from critical processing is required.
  - **Abstracting timing data**: RTOS provides timing and API functions allowing for smaller and cleaner application code.
  - **Modularity**: Naturally, task-based API encourages the development of modular as the role is typically defined clearly.
  - **Team development**: RTOS allows separate teams or designers to operate independently on the project.
  - **Easier testing**: RTOS is the best option when you want to make testing easier. This is because it allows for testing of modular tasks.
  - **Enhanced efficiency**: The operating system can be event driven fully. This means that no processing time will be wasted on polling for an event that does not happen.

- **Code reuse**: Modularity allows for the development of standard tasks library, since similar applications are used on similar platforms.
- **Idle processing**: Idle or background processing can be performed in idle tasks. As such, things like background CRC checking and CPU load measurement do not affect the primary processing.

4.1.2 Components of Real Time OS

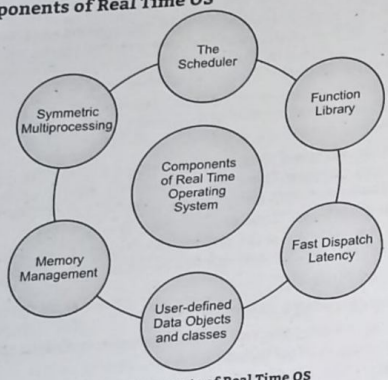


Fig. 4.2: Components of Real Time OS

Important components of RTOS system are:

1. The Scheduler
2. Symmetric Multiprocessing
3. Function Library
4. Memory Management
5. Fast dispatch latency.
6. User-defined data objects and classes.

- Let us details of each Component of RTOS:

1. **The Scheduler**: This component of RTOS tells that in which order, the tasks can be executed which is generally based on the priority.
2. **Symmetric Multiprocessing (SMP)**: It is a number of multiple different tasks that can be handled by the RTOS so that parallel processing can be done.
3. **Function Library**: It is an important element of RTOS that acts as an interface that helps you to connect kernel and application code. This application allows you to send the requests to the kernel using a function library so that the application can give the desired results.

4. **Memory Management**: This element is needed in the system to allocate memory to every program, which is the most important element of the RTOS.
5. **Fast dispatch latency**: It is an interval between the termination of the task that can be identified by the OS and the actual time taken by the thread, which is in the ready queue that has started processing.
6. **User-defined data objects and classes**: RTOS system makes use of programming languages like C or C++, which should be organized according to their operation.

4.1.3 Types of Real Time Operating System

- There are two types of real-time operating systems. These types of real time systems based on timing constraints:

1. **Hard Real Time System(HRTS):**
   - In Hard RTOS, the deadline is handled very strictly which means that given task must start executing on specified scheduled time, and must be completed within the assigned time duration.
   - In other words, this type of system can never miss its deadline. These systems guarantee that critical tasks complete on time. Missing the deadline may have disastrous consequences.
   - The usefulness of result produced by a hard real time system decreases abruptly and may become negative if tardiness increases. Tardiness means how late a real time system completes its task with respect to its deadline.
   - 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.
   - **Examples:** Flight controller system, Medical critical care system, etc.
2. **Soft Real Time System (SRTS):**
   - Soft Real Time RTOS accepts some delays by the Operating system. In this type of RTOS, there is a deadline assigned for a specific job, but a delay for a small amount of time is acceptable. So, deadlines are handled softly by this type of RTOS.
   - This type of system can miss its deadline occasionally with some acceptably low probability. Missing the deadline has no disastrous consequences. The usefulness of result produced by a soft real time system decreases gradually with increase in tardiness. **Example:** Telephone switches.
   - 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.
   - **Examples:** Multimedia, Virtual reality, Advanced Scientific Projects like Undersea exploration and planetary rovers, Online Transaction system and Livestock price quotation System.

3. **Firm Real Time:**
   - These types of RTOS also need to follow the deadlines. However, missing a deadline may not have big impact but could cause undesired affects, like a huge reduction in quality of a product.
   - **Example:** Various types of Multimedia applications.

4.1.4 Features of RTOS

- Here are important features of RTOS:
  - Occupy very less memory.
  - Consume fewer resources.
  - Response times are highly predictable.
  - Unpredictable environment.
  - The kernel saves the state of the interrupted task ad then determines which task it should run next.
  - The kernel restores the state of the task and passes control of the CPU for that task.

4.1.5 RTOS Architecture

- The architecture of an RTOS is dependent on the complexity of its deployment. Good RTOSs are scalable to meet different sets of requirements for different applications. For simple applications, an RTOS usually comprises only a kernel.
- For more complex embedded systems, an RTOS can be a combination of various modules, including the kernel, networking protocol stacks, and other components as illustrated in Fig. 4.3.

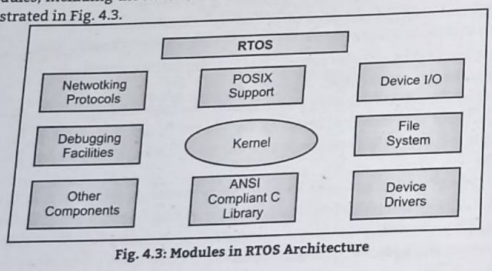


Fig. 4.3: Modules in RTOS Architecture

Kernel:

- An operating system generally consists of two parts: Kernel space (kernel mode) and User space (user mode). Kernel is the smallest and central component of an operating system. Its services include managing memory and devices and also to provide an interface for software applications to use the resources. Additional services such as managing protection of programs and multitasking may be included depending on architecture of operating system.

- There are three broad categories of kernel models available, namely:
  1. **Monolithic kernel:** It runs all basic system services (i.e. process and memory management, interrupt handling and I/O communication, file system, etc) in kernel space. As such, monolithic kernels provide rich and powerful abstractions of the underlying hardware. Amount of context switches and messaging involved are greatly reduced which makes it run faster than microkernel. **Examples** are Linux and Windows.
  2. **Microkernel:** It runs only basic process communication (messaging) and I/O control. The other system services (file system. networking, etc) reside in user space in the form of daemons/servers. Thus, micro kernels provide a smaller set of simple hardware abstractions. It is more stable than monolithic as the kernel is unaffected even if the servers failed (i.e. File System). **Examples** are AmigaOS and QNX.
  3. **Exokernel:** The concept is orthogonal to that of micro- vs. monolithic kernels by giving an application efficient control over hardware. It runs only services protecting the resources (i.e. tracking the ownership, guarding the usage, revoking access to resources, etc.) by providing low-level interface for library operating systems (libOSes) and leaving the management to the application.

4.1.6 Factors needed for Selecting RTOS

- Here, are essential factors that you need to consider for selecting RTOS:
  - **Performance:** Performance is the most important factor required to be considered while selecting for a RTOS.
  - **Middleware:** If there is no middleware support in Real time operating system, then the issue of time-taken integration of processes occurs.
  - **Error-free:** RTOS systems are error-free. Therefore, there is no chance of getting an error while performing the task.
  - **Embedded system usage:** Programs of RTOS are of small size. So we widely use RTOS for embedded systems.
  - **Maximum Consumption:** We can achieve maximum Consumption with the help of RTOS. It can be also using by the servers that are hosted to give maximum output of hosting companies.
  - **Task shifting:** Shifting time of the tasks is very less.
  - **Unique features:** A good RTOS should be capable, and it has some extra features like how it operates to execute a command, efficient protection of the memory of the system, etc.
  - **24/7 performance:** RTOS is ideal for those applications which require to run 24/7 because it do less task shifting and give maximum output.

4.1.7 Applications of RTOS

- Real-time System has applications in various fields of the technology. Here we will discuss the important applications of real-time system.

1. **Industrial applications:** Real-time system has a vast and prominent role in modern industries. Systems are made real time based so that maximum and accurate output can be obtained. In order to such things real-time systems are used in maximum industrial organizations. These systems somehow lead to the better performance and high productivity in less time. Some of the examples of industrial applications are: Automated Car Assembly Plant, Chemical Plant, Anti-lock Brake Systems etc.
2. **Medical Science applications:** In the field of medical science, real-time system has a huge impact on the human health and treatment. Due to the introduction of real-time system in medical science, many lives are saved and treatment of complex diseases has been turned down to easier ways. People specially related to medical, now feel more relaxed due to these systems. Some of the examples of medical science applications are: Robot, MRI Scan, Radiation therapy, Heart Pacemaker etc.
3. **Peripheral Equipment applications:** Real-time system has made the printing of large banners and such things very easier. Once these systems came into use, the technology world became stronger. Peripheral equipment is used for various purposes. These systems are embedded with microchips and perform accurately in order to get the desired response. Some of the examples of Peripheral equipment applications are: Laser printer, fax machine, digital camera etc.
4. **Telecommunication applications:** Real-time system maps the world in such a way that it can be connected within a short time. Real-time systems have enabled the whole world to connect via a medium across internet. These systems make the people connect with each other in no time and feel the real environment of togetherness. Some examples of telecommunication applications of real-time systems are: Video Conferencing, Cellular system etc.
5. **Defence applications:** In the new era of atomic world, defence is able to produce the missiles which have the dangerous powers and have the great destroying ability. All these systems are real-time system and it provides the system to attack and also a system to defend. Some of the applications of defence using real time systems are: Missile guidance system, anti-missile system, Satellite missile system, RADAR etc.
6. **Aerospace applications:** The most powerful use of real time system is in aerospace applications. Basically hard real time systems are used in aerospace applications. Here, the delay of even some nanosecond is not allowed and if it happens, system fails. Some of the applications of real-time systems in aerospace are: Satellite tracking system, Avionics, Flight simulation etc.

4.1.8 Disadvantages of RTOS

- Here, are disadvantages of using RTOS system:
  - RTOS system can run minimal tasks together, and it concentrates only on those applications which contain an error so that it can avoid them.
  - This is the system that concentrates on a few tasks. Therefore, it is really hard for these systems to do multi-tasking.
  - Specific drivers are required for the RTOS so that it can offer fast response time to interrupt signals, which helps to maintain its speed.
  - Plenty of resources are used by RTOS, which makes this system expensive.
  - The tasks which have a low priority need to wait for a long time as the RTOS maintains the accuracy of the program, which are under execution.
  - Minimum switching of tasks is done in Real time operating systems.
  - It uses complex algorithms which is difficult to understand.
  - RTOS uses lot of resources, which sometimes not suitable for the system.

4.2 EMBEDDED OS

- An embedded operating system (OS) is a specialized operating system designed to perform a specific task for a device that is not a computer. An embedded operating system's main job is to run the code that allows the device to do its job.

4.2.1 Concept, Need and Features of Embedded OS

Concept:

- **Embedded Operating Systems:** We find embedded System everywhere around us in our daily life. Embedded Systems are a specially designed computer system that essentially contains software and hardware for performing specific tasks. Mobile Phones, Laptops, Cameras, Washing Machines, ATMs, and Hair Straightener etc. are examples of Embedded System. Now let's see what is Embedded Operating Systems.

What is Embedded Operating System?

- An embedded operating system's main job is to run the code that allows the device to do its job. The embedded OS also makes the device's hardware accessible to the software that is running on top of the OS.
- As the name suggests Embedded Operating System is an Embedded System's Operating System. It is usually designed for some particular operations to control an electronic device. For instance, all mobile phones essentially consist of an operating system that always boots up when the mobile phone is in running condition. It controls all the features and basic interface of the mobile phone. There are some other programs that can be loaded onto the mobile phones. Mostly, JAVA Apps run on the top. Embedded operating systems run on embedded processors.

Features of Embedded Operating Systems:

- The main features of Embedded Operating Systems are as follows:
  1. **Direct use of interrupts:** The embedded operating system provides the use of interrupts to give them more control over the peripheral. The interrupt also has the priority. And according to that priority, the task is serviced by the CPU.
  2. **Reactive operation:** A system is called reactive if it acts on certain input by the user in the form of external events such as switch press or by some sensor. For example, a motion sensor in security system.
  3. **Real-time operation:** Real-time embedded systems have a time constrained to execute the task, which is called a deadline. The soft-real time system may vary the deadline. But the hard real-time system must complete the task in a given time frame.
  4. **Streamlined protection mechanisms:** embedded systems are typically designed for a limited, well-defined functionality. Untested programs are rarely added to the software. After the software has been configured and tested, it can be assumed to be reliable. Thus, apart from security measures, embedded systems have limited protection mechanisms.
  5. ***I/O device flexibility:*** There is no generalize hardware that is suitable or adjustable for all operating system versions.
  6. **Configurability:** Embedded systems are designed as per the application requirement. And according to the hardware we need to customize the embedded operating system. So the operating system should be designed in such a way that an embedded developer can configure the operating system as per the need.
  7. **Fast and Lightweight:** As the embedded systems have small CPU with limited processing power. It should be customized perfectly so that it can execute fast.
  8. **Small size:** It has very limited resources like RAM, ROM and CPU power. So keep the embedded operating system small as possible to fit into given memory space.
- There are two different kinds of operating system, either general purpose operating system that is modified in such a way that it runs on top of a device or the operating system can be custom written. The approaches for the design of operating system include that either we take embedded Operating System that is existing and adapt it to our embedded application or we can design and use a new operating system that is particularly for our Embedded System.
- We can adapt the existing Operating System to our embedded application by streamline operation, real-time capability and be adding other necessary functions. The advantage of this approach that it has a familiar interface and its disadvantage is that it is not optimized for real-time.
- Flash Memory Chip is added on a motherboard in case of the embedded operating system of your personal computer to boot from the Personal Computer.

4.2.2 Basic Structure of an Embedded System

- The following illustration shows the basic structure of an embedded system:

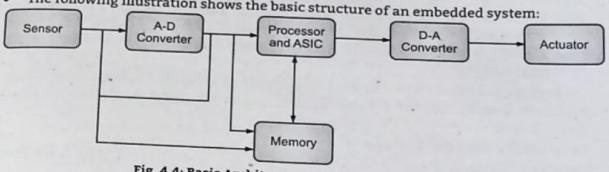


Fig. 4.4: Basic Architecture of an Embedded System

1. **Sensor:** It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.
2. **A-D Converter:** An analog-to-digital (A-D) converter converts the analog signal sent by the sensor into a digital signal.
3. **Processor & ASICs:** Processors process the data to measure the output and store it to the memory.
4. **D-A Converter:** A digital-to-analog(D-A) converter converts the digital data fed by the processor to analog data
5. **Actuator:** An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

4.2.3 Types of Embedded Operating Systems

- Embedded systems can be classified into different types based on performance, functional requirements and performance of the microcontroller.

(I) Embedded systems based on Performance and Functional requirements:

- Embedded systems are classified into four categories based on their performance and functional requirements:
  - Stand alone embedded systems
  - Real time embedded systems
  - Networked embedded systems
  - Mobile embedded systems

1. Stand alone Embedded Systems:

- Stand alone Embedded Systems are those that can work by themselves i.e. they are self-sufficient and do not depend on a host system. Stand-alone embedded systems are made in a way such that an input is received, processed and thereafter the desired output is produces. Input can be received via sensors, keyword or push button.
- **Examples:** mp3 players, digital cameras, video game consoles, microwave ovens and temperature measurement systems.

2. **Real Time Operating System:**
   - A real-time operating system is the one which serves real time applications. It processes data as it comes in. The time requirements for processing of operating system are usually measured in shorter increments or in 10^{th} of seconds. They may be time sharing or driven by events. Real time Operating systems are used in small embedded systems.
   - These types of embedded systems follow the time deadlines for completion of a task. Real time embedded systems are classified into two types such as soft and hard real time systems.
3. **Networked Embedded Systems:**
   - Networked Embedded Systems depend on connected network to perform its assigned tasks.
   - These systems consist of components like sensors, controllers etc. which are interconnected. Many of these systems are built on general purpose processors.
   - Example for the LAN networked embedded system is a home security system wherein all sensors are connected and run on the protocol TCP/IP.
4. **Mobile Embedded Systems:**
   - Mobile Embedded Systems are those that are small-sized . Mobile embedded systems are used in portable embedded devices like cell phones, mobiles, digital cameras, mp3 players and personal digital assistants, etc.
   - They are used in mobile phones and digital cameras because of the small size. They often have memory constraints and lacks good user interface.

(II) Microcontroller Performance Based Embedded System:

- Embedded Systems are classified in three types based on its microcontroller performance.

1. **Small Scale Embedded System:** Small Scale Embedded System is normally designed and created using an 8-bit microcontroller. This microcontroller can be battery activated. For developing embedded software for small scale embedded systems, the main programming tools are an editor, assembler, cross assembler and integrated development environment (IDE).
2. **Medium Scale Embedded System:** Medium Scale Embedded System uses a single 16-bit or 32-bit microcontroller or multiple microcontrollers linked together. These systems have a lot of hardware as well as software complexities, hence are not preferred by many. For developing embedded software for medium scale embedded systems, the main programming tools are C, C++, JAVA, Visual C++, RTOS, debugger, source code engineering tool, simulator and IDE.
3. **Sophisticated Embedded System:** Sophisticated Embedded System often functions on multiple algorithms that result in complexities in both hardware and software. They often need a processor that is configurable and logic array that can be programmed. They are used for cutting-edge applications that need hardware and software co-design and components which have to assemble in the final system.

4.2.4 Examples of Embedded Operating Systems with brief Introduction

- An embedded system is a device with a computer designed for a specific purpose. To achieve that, the device needs an operating system that can respond fast and is prepared to keep working in any event. That is why we cannot rely on a general-use OS, but an embedded operating system.
- An embedded operating system is an OS designed and optimized to:
  1. Improve the efficiency of managing the hardware resources.
  2. Reduce response times specifically for the task the device was created for.
- Some examples of embedded systems are industrial robots, smart devices, IoT machines, drones, medical systems, video game consoles, and many others.
- The most common examples of embedded operating system around us include:
  1. Windows Mobile/CE (handheld Personal Data Assistants),
  2. Symbian (cell phones) and
  3. Linux.

Real Time Embedded Operating Systems Examples:

1. **VxWorks:**
   - It is developed by Wind River.
   - The latest version of this operating system is VxWorks 6.0.
   - It is widely used software operating system.
   - At the moment, there are 300 million devices that utilize this operating system.
   - The core or processor of VxWorks credits reliability, high performance, low latency, determinism, and scalability.
   - It has increased potential for the management of errors.
   - It has compatibility with extensive POSIX 1003.1, .1b, .1c,
   - It uses preemptive priority and has scheduled for both real time and non-real time processes.
   - Memory is protected on the basis of MMU.
   - It has reduced context switch time. It restores only register windows.
   - For increasing response time, it saves the windows that are registered in a register cache and they are used for recurring tasks.
2. **Microkernels:**
   - Microkernels is an operating system that consists of the basic necessities like scheduling, task switching, and device handling.
   - Micro Kernels implement different memory techniques and protect all system components.
   - Some of the microkernels protect memory by separating all components of software from one another.

4.2.5 Applications of Embedded Operating Systems

(I) Here we have different applications of Embedded Operating System in our everyday life. These applications include:

- **Symbian:** It is used in mobile phones mainly in Nokia.
- **BlackBerry Operating System:** It is particularly used in BlackBerry Phones,
- **Embedded Linux:** It is used in Android phones and other devices like printers.
- **iOS:** It is used in MAC operating systems and other Apple devices.
- **Windows Mobile Operating System:** It is used in window phones.

(II) Some real-life applications of Embedded Systems are as follows:

- **Consumer electronics:** Televisions and digital cameras; computer printers; video game consoles and home entertainment systems like PS4.
- **Household appliances:** Refrigerators; washing machines, microwave ovens, air conditioners.
- **Medical equipment:** Scanners like those for MRI, CT; ECG machines; devices to monitor blood pressure and heartbeat.
- **Automobiles:** Fuel injection systems, anti-lock braking systems, music and entertainment systems, controls for air-conditioner.
- **Industrial applications:** Assembly lines, systems for feedback, systems for data collection
- **Aerospace:** Systems for navigation and guidance, GPS.
- **Communications:** Routers, satellite phones.

4.2.6 Embedded Operating System Vs Desktop Operating System

Table 4.1: Difference between Embedded OS and Desktop OS

Sr. No.	Embedded Operating System	Desktop Operating System
1.	The first embedded OS is Apollo guidance computer in 1965.	The first desktop OS is NLC (On-Line system) developed in 1960.
2.	It is designed to run only a single task.	It is designed to run many tasks simultaneously.
3.	Boot time is faster compared to desktop OS.	Boot time is slower in desktop OS.
4.	Performance of the web browser takes less time to load the websites.	Performance of the web browser takes a long time to load the websites.
5.	It takes less time to run the applications.	It takes a longer time to run the applications.
6.	It uses only flash drives for storage.	It uses hard drives and flash drives for storage.
7.	Embedded OS cost is less.	Cost is expensive.
8.	It requires less storage compared to desktop OS.	It requires more storage.
9.	It has fewer application features.	It has more application features.

Summary

- A Real Time Operating System, commonly known as an RTOS, is a software component that rapidly switches between tasks, giving the impression that multiple programs are being executed at t he same time on a single processing core.
- Important components of RTOS system are:
  1. The Scheduler.
  2. Symmetric Multiprocessing.
  3. Function Library.
  4. Memory Management.
  5. Fast dispatch latency.
  6. User-defined data objects and classes.
- Three types of RTOS are:
  1. Hard time
  2. Soft time
  3. Firm time
- Real-time systems are used in Airlines reservation system, Air traffic control system, etc.
- The biggest drawback of RTOS is that the system only concentrates on a few tasks.
- An Embedded operating system is an operating system for embedded computer systems. It is usually designed for some particular operations to control an electronic device.
- The most common examples of Embedded operating system around us include Windows Mobile/CE (handheld Personal Data Assistants), Symbian (cell phones) and Linux.

Check Your Understanding

1. In Real Time Operating System ____.
   (a) All processes have the same priority
   (b) A task must be serviced by its deadline period.
   (c) Process scheduling can be done only once
   (d) Kernel is not required
2. Which one of the following is a real time operating system?
   (a) Rtlinux (b) VxWorks
   (c) Windows CE (d) All of the mentioned
3. Which is not application of embedded OS?
   (a) Symbian (b) Blackberry
   (c) Unix (d) Linux

4. Aircraft system is an example of____.
   (a) Hard real OS (b) Soft Real OS
   (c) Firm Real OS (d) Time sharing OS
5. Which is component of RTOS?
   (a) Symmetric Multiprocessing (b) Scheduler
   (c) Memory Management (d) All of the above

Answer Key

1. (b)	2. (d)	3. (c)	4. (a)	5. (d)

Practice Questions

Q.I Answer the following questions in short.

1. What is RTOS?
2. What is embedded OS?
3. Which are types of RTOS?
4. Write examples of Embedded OS?
5. Write important features of RTOS.

Q.II Answer the following questions.

1. Explain applications of RTOS in detail.
2. Write in detail need and features of Embedded OS.
3. Describe components of RTOS.
4. Write the types of Embedded OS.
5. Which are types of operating systems? Explain in detail.

Q.III Write a short note on:

1. Hard time RTOS.
2. Soft time RTOS.
3. Firm RTOS.
4. Deadline of job.
5. VxWorks.

■■■

5... Windows OS and Windows Server Architecture

Objectives ...

- To study about Windows OS.
- To learn Ubuntu OS.

5.1 WINDOWS OS

- An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of its software and hardware.
- It also allows you to communicate with the computer without knowing how to speak the computer's language. Without an operating system, a computer is useless.
- Microsoft created the Windows operating system in the mid-1980s. There have been many different versions of Windows, but the most recent ones are Windows 10 (released in 2015), Windows 8 (2012), Windows 7 (2009), and Windows Vista (2007). Windows comes pre-loaded on most new PCs, which helps to make it the most popular operating system in the world.
- Windows 10 is a latest version of the Microsoft Windows operating system that was released on July 29, 2015. It is built on the Windows NT kernel and follows Windows 8.

Some of the functions of Windows Operating System are:

- Access applications (programs) on the computer (word processing, games, spread sheets, calculators and so on).
- Load any new program on the computer.
- Manage hardware such as printers, scanners, mouse, digital cameras etc.,
- File management activities (For example creating, modifying, saving, deleting files and folders).
- Change computer settings such as color scheme, screen savers of monitor, etc.

5.1.1 Introduction

- MS Windows is a multitasking operating system which uses a Graphical User Interface (GUI) to link the user to the computer.
- A GUI is an interface that helps users to interact with the computer by use of windows, icons, and menus.
- The interaction between the user and the computer heavily relies on the use of a mouse and keyboard which make it possible to enter data into the computer and also manipulate it the way we want.

5.1.2 Windows OS Installation

Installing or Upgrading Windows:

- To start the windows install or upgrade process, you need to configure our computer to boot from a CD or DVD before booting to the hard drive. Changing the boot process forces the computer to look for the Windows installation disc before trying to boot from the **hard drive.**

1. Open the BIOS or CMOS setup.
2. Change the computer's boot order. Set the CD, DVD or disc drive as the first boot device if you are trying to boot from a disc. Or, set the first boot device to your USB drive if you're trying to boot from a USB thumb drive. If the drive is not shown, keep the disc is inserted and reboot the computer. With the disc in the drive, BIOS should recognize and include it in the list.
3. Save the settings change and exit BIOS. Once you have updated the boot order, you can begin the Windows installation process.
4. Place the Windows disc in the CD/DVD drive or USB thumb drive into the back of the computer.
5. Turn on or restart the computer. As the computer starts up, it should detect the installation disc or drive and show a message similar to Press any key to boot from CD. Press any key on the keyboard to have the computer boot from the Windows disc or drive.
6. After the Windows install begins, there are several prompts that you need to answer. Select either Yes or the appropriate option to install Windows.
7. When asked which partition to install Windows onto, select the main partition, which is usually the C: drive or one labeled "Unallocated partition". If upgrading Windows, select the existing installation of Windows on the hard drive.
8. You may be asked if you want to erase all contents on the hard drive, then install Windows. We recommend you choose this option, as it also formats the hard drive to allow the Windows operating system to be installed.
9. The computer may need to restart several times during the Windows install process. The restarts are normal and if prompted to restart, select the Yes option.

10. When the install process is nearly complete, the Windows configuration option screens are shown. On these screens, you may be asked to select the time zone you live in, your preferred language, and the name of the account you use to access Windows. Select the appropriate options and enter the appropriate information on each configuration screen.

- The Windows install process is completed when the computer prompts you to log in or when it loads into Windows.

5.1.3 Process Management

- Every process contains one or more threads, and the Windows thread is the basic executable unit.
- Threads are scheduled on the basis of the usual factors: availability of resources such as CPUs and physical memory, priority, fairness, and so on. Windows has long supported multiprocessor systems, so threads can be allocated to separate processors within a computer.
- **Following are Windows Process Components:**
  - One or more threads.
  - A virtual address space that is distinct from other processes' address spaces. Note that shared memory-mapped files share physical memory, but the sharing processes will probably use different -virtual addresses to access the mapped file.
  - One or more code segments, including code in DLLs.
  - One or more data segments containing global variables.
  - Environment strings with environment variable information, such as the -current search path.
  - The process heap.
  - Resources such as open handles and other heaps.
- Each thread in a process shares code, global variables, environment strings, and resources. Each thread is independently scheduled, and a thread has the following elements:
  - A stack for procedure calls, interrupts, exception handlers, and automatic storage.
  - Thread Local Storage (TLS)- An array like collection of pointers giving each thread the ability to allocate storage to create its own unique data environment.
  - An argument on the stack, from the creating thread, which is usually unique for each thread.
  - A context structure, maintained by the kernel, with machine registers values.
- Fig 5.1 shows a process with several threads. This figure is schematic and does not indicate actual memory addresses, nor is it drawn to scale.

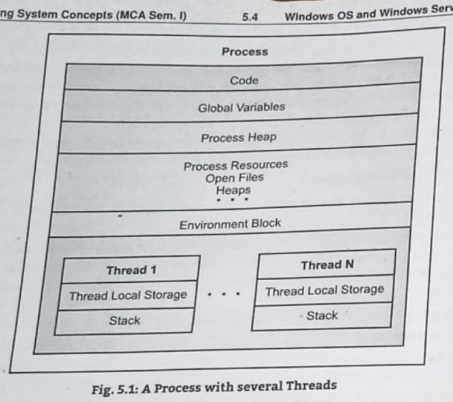


Fig. 5.1: A Process with several Threads

Process Creation:

- The fundamental Windows process management function is Create Process, which creates a process with a single thread. Specify the name of an executable program file as part of the Create Process call.
- It is common to speak of parent and child processes, but Windows does not actually maintain these relationships. It is simply convenient to refer to the process that creates a child process as the parent.

5.1.4 Control Panel Overview

- The **Control Panel** is a component of Microsoft Windows that provides the ability to view and change system settings.
- It consists of a set of applets that include adding or removing hardware and software, controlling user accounts, changing accessibility options, and accessing networking settings.
- Additional applets are provided by third parties, such as audio and video drivers, VPN tools, input devices, and networking tools.
- Control Panel allows you to view and change settings (controls) for Windows via applets.

Start Control Panel in Windows Server 2012:

1. **Open Control Panel with Shortcut:** Windows + X shortcut is used commonly in Windows server 2012 to open common Windows programs or tools easily. Press Windows + X on desktop, and click Control Panel to open it.
2. **Open Control Panel in Start Screen:** Press Windows + C and click Start to enter Start screen. Or move mouse to the left bottom side of desktop screen, click "Start" button. On Start screen, you could see Control Panel, click it and open.
3. **Access Control Panel from Desktop:** In Windows server 2012 desktop, move mouse to the right upper side of screen, and then click "Settings" button. And then click Control Panel.
4. **Start Control Panel over Run Dialog Box:** Press Windows + R and type following command: shell:ControlPanelFolder or control and then press Enter.
5. **Run Control Panel with command prompt:** Open Command Prompt first. And use the command control to start the Control Panel via command prompt.
6. **Access Control Panel via Explorer:** Press Windows + E and then type "Control Panel" in address bar and press Enter.

5.1.5 Users, Security and Privacy Settings

5.1.5.1 User Security Settings

- Windows 10, the account creation process during the initial setup of a new device or installation. Lot of additional configuration options that you can use to make your account more secure and to enhance the experience.

To manage account sign-in options on Windows 10:

- On Windows 10, the "Sign-in options" page allows you to manage the different methods available for authentication to your account. Using these settings, you can change your current password, set up Windows Hello Face, PIN, Security Key, or Picture Password, and you can even turn on Dynamic lock to lock the device as you step away.

(A) Change Account Password:

- To change the account password on Windows 10, use these steps:
  1. Open **Settings**.
  2. Click on **Accounts**.
  3. Click on **Sign-in options**.
  4. Under the "Manage how your sign in to your device" section, select. the **Password** option.
  5. Click the **Change** button.

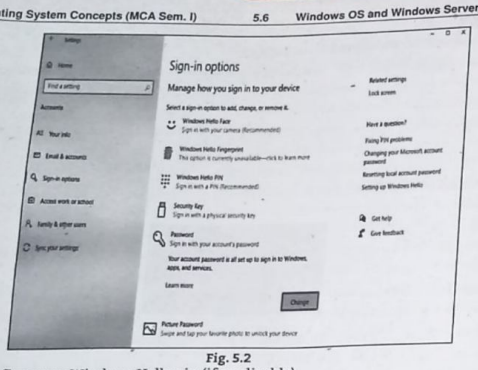


Fig. 5.2

6. Confirm your Windows Hello pin (if applicable).
7. Confirm the current password.
8. Confirm the new password.

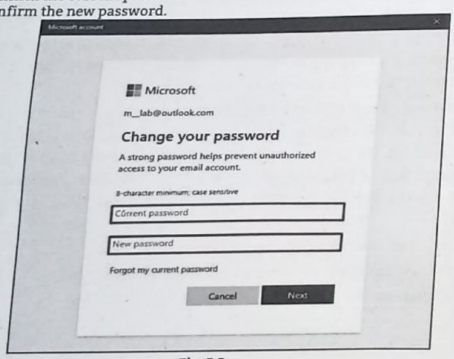


Fig. 5.3

9. Click the ***Next*** button.
10. Continue with the on-screen directions (if applicable).

- After you complete the steps, the password will update on your Windows 10 installation, as well as across devices and services if you use a Microsoft account.

(B) Set up Windows Hello PIN:

- If you want to use a faster and more secure way to sign in to Windows 10, you can use a PIN instead of a traditional password.
- To add a PIN to your Windows 10 account, use these steps:

1. Open **Settings**.
2. Click on **Accounts**.
3. Click on **Sign-in options**.
4. Under the "Manage how your sign in to your device" section, select the **Windows Hello PIN** option.
5. Click the **Add** button.

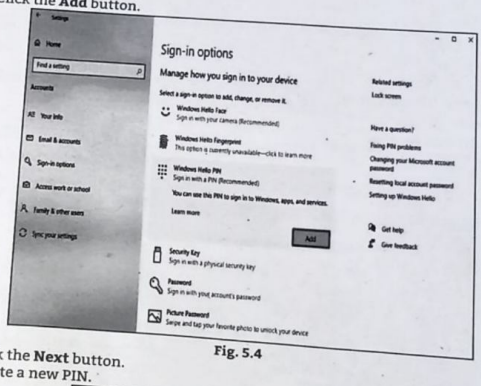


Fig. 5.4

6. Click the **Next** button.
7. Create a new PIN.

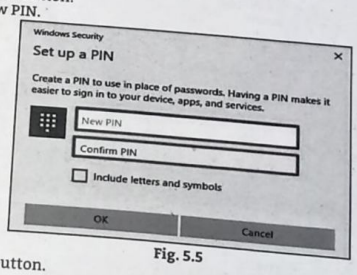


Fig. 5.5

8. Click the **OK** button.

- Once you complete the steps, you can start using the PIN instead of a password to quickly sign in to your account.
- Usually, a PIN is more secure than a password, because it's only tied to one device, and it's never transmitted over the network, and it only works locally. You can't use it to access your device remotely, and you need to set it per device.

(C) Set up Picture Password:

- Windows 10 even lets you use a picture as a password. This authentication method allows you to use gestures on an image to sign in. To configure a picture password, use these steps:

1. Open **Settings**.
2. Click on **Accounts**.
3. Click on **Sign-in options**.
4. Under the "Manage how you sign in to your device" section, select the **Picture Password** option.
5. Click the **Add** button.

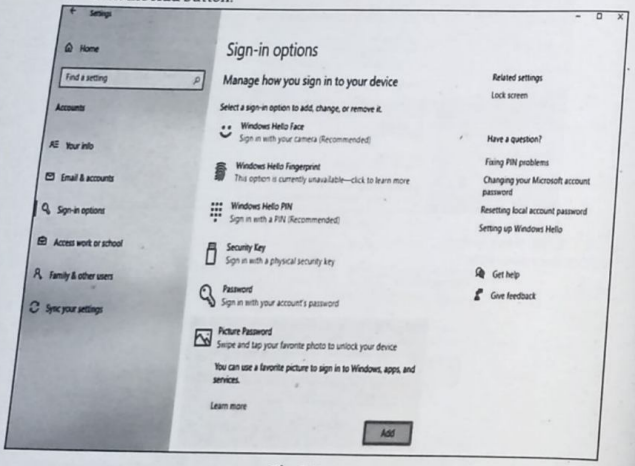


Fig. 5.6

6. Confirm the current password.
7. Click the **Choose new picture** button from the left pane.

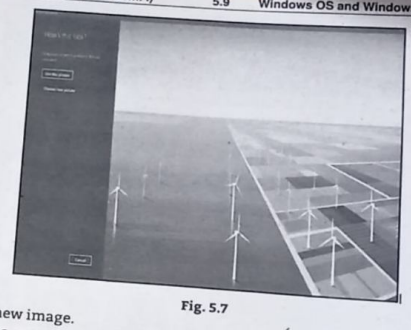


Fig. 5.7

8. Select a new image.
9. Click the **Open** button.
10. Click the **Use this picture** button.

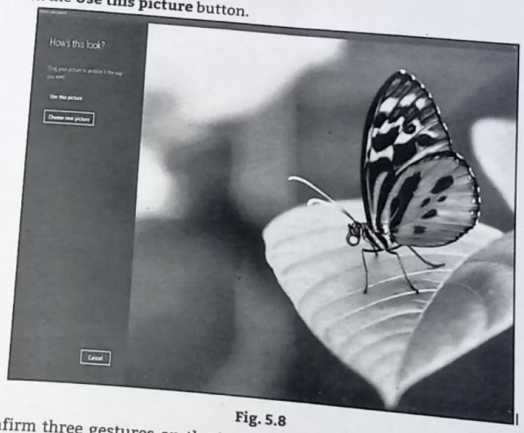


Fig. 5.8

11. Confirm three gestures on the image, including circles, straight lines, taps, or a combination of the three, which you'll use as a password.

Fig. 5.9

12. Repeat the gestures to confirm.
13. Click the **Finish** button.

- After you complete the steps, sign out, and when you try to sign back in, you can start using gesture patterns to sign in.

5.1.5.2 User Privacy Setting

- In this point we study, how we can secure our privacy with Windows 10:

(A) Turn off ad tracking:

- At the top of many people's privacy concerns is what data is being gathered about them as they browse the web.
- Windows 10 does this with the use of an advertising ID. The ID doesn't just gather information about you when you browse the web, but also when you use Windows 10 apps.
- You can turn that advertising ID off if you want. Launch the Windows 10 Settings app (by clicking on the Start button at the lower left corner of your screen and then clicking the Settings icon, which looks like a gear) and go to Privacy > General. There you'll see a list of choices under the title "Change privacy options"; the first controls the advertising ID. Move the slider from On to Off. You'll still get ads delivered to you, but they'll be generic ones rather than targeted ones, and your interests won't be tracked.

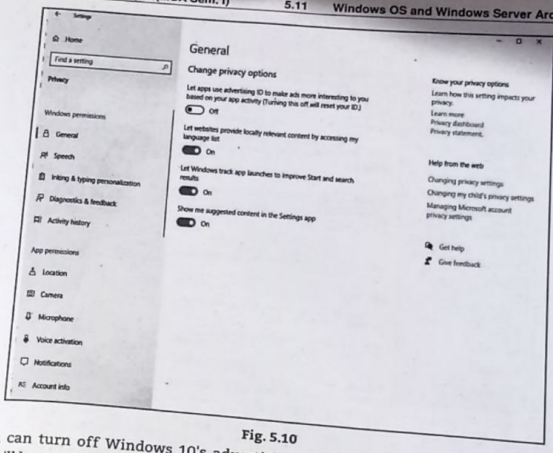


Fig. 5.10

- You can turn off Windows 10's advertising ID if you want. You'll still get ads, but they'll be generic ones.
- To make absolutely sure you're not tracked online when you use Windows 10, and to turn off any other ways Microsoft will use information about you to target ads, head to the Ad Settings section of Microsoft's Privacy Dashboard. Sign into your Microsoft account toward the top of the page.
- Then go to the "See ads that interest you" section at the top of the page and move the slider from On to Off. After that, scroll down to the "See personalized ads in your browser" section and move the slider from On to Off. Note that you need to go to every browser you use and make sure the slider for "See personalized ads in your browser" is set to Off.

(B) Turn off location tracking:

- Wherever you go, Windows 10 knows you're there. Some people don't mind this, because it helps the operating system give you relevant information, such as your local weather, what restaurants are nearby and so on. But if you don't want Windows 10 to track your location, you can tell it to stop.
- Launch the Settings app and go to Privacy > Location. Underneath "Allow access to location on this device," click Change and, on the screen that appears, move the slider from On to Off. Doing that turns off all location tracking for every user on the PC.

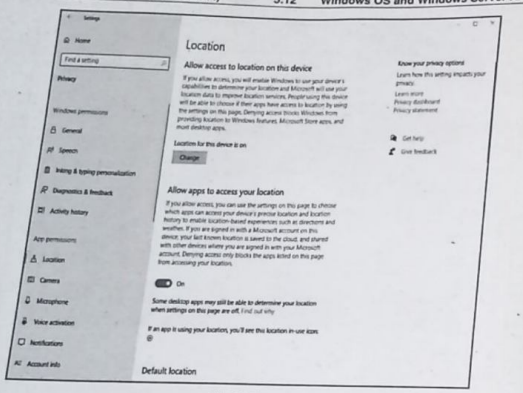


Fig. 5.11

- *If you click the Change button, you can turn off location tracking for every user on the Windows 10 device.*
- *This doesn't have to be all-or-nothing affair: you can turn off location tracking on an app-by-app basis. If you want your location to be used only for some apps and not others, make sure location tracking is turned on, then scroll down to the "Choose apps that can use your precise location" section. You'll see a list of every app that can use your location. Move the slider to On for the apps you want to allow to use your location.*
- *When you turn off location tracking, Windows 10 will still keep a record of your past location history. To clear your location history, go to the Privacy Dashboard, scroll down to the Location Activity section, click View and Clear Location Activity, and delete all or some of your location history.*

(C) Turn off Timeline:

- *Timeline lets you review and then resume activities and open files you've started on your Windows 10 PC, as well as any other Windows PCs and devices you have.*
- *In order to do that, Windows needs to gather information about all your activities on each of your machines. If that worries you, it's easy to turn Timeline off. To do it, go to Settings > Privacy > Activity History and uncheck the boxes next to Store my activity history on this device and Send my activity history to Microsoft.*

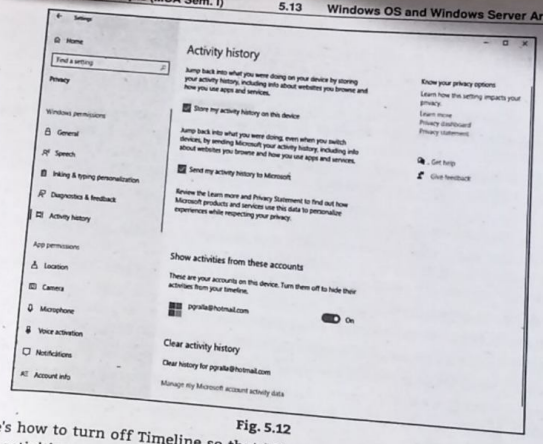


Fig. 5.12

- *Here's how to turn off Timeline so that Microsoft doesn't gather information about your activities on your PC.*
- *At that point, Windows 10 no longer gathers information about your old activities and shows them in your Timeline on all your PCs. To get rid of that old information, in the "Clear activity history" section of the screen, click Manage my Microsoft account activity data. You'll be sent to Microsoft's Privacy Dashboard, where you can clear your data. See the section later in this article on how to use the privacy dashboard to do that.*

5.1.6 Identify Accessibility Settings

Adjust Size and Color:

- *To adjust the size of text, apps, and other items, select the Start button, then select Settings ->Ease of Access -> Display.*
- *Use the slider under Make text bigger to make just the text on your screen larger. Or, select an option from the drop-down menu under Make everything bigger to change the size of everything on your screen.*

Make everything bigger:

- *If there's not enough contrast between the elements on your screen, try using a high contrast theme. Select the Start button, then select Settings > Ease of Access > High contrast, and switch on the toggle under Turn on high contrast.*

Turn on high contrast:

- To make the apps in your Start menu appear larger, right-click (or tap and hold) the app tile you want to resize, select Resize, and then choose the size that you want.

Magnify your screen:

- Magnifier enlarges part or all of your screen so you can see words and images better. To open Magnifier quickly, press the *Windows logo key + Plus sign (+)*.
- When Magnifier is open, use *Windows logo key + Plus sign (+)* or *Windows logo key + Minus sign (–)* to zoom in or out. To close Magnifier, press *Windows logo key + Esc*.

Narrator to navigate your PC:

- Narrator is the built-in screen reader in Windows that reads aloud what's on your screen so you can use that information to navigate your PC.
- To start or stop Narrator, press the *Windows logo key + Ctrl + Enter*.

Scan Mode to get going fast:

- Scan Mode can help you navigate apps and webpages more quickly.
- To turn Scan Mode on or off, press *Caps lock + Spacebar*.
- When Scan Mode is turned on, press the up and down arrow keys to move to the next or previous line of text in an app or webpage. Press the right and left arrow keys to move to the next or previous character. To activate an item like a button, press the Spacebar.
- You can also navigate by word or paragraph. Press *Ctrl + Left arrow* and *Ctrl + Right arrow* to move by word. Press *Ctrl + Up arrow* and *Ctrl + Down arrow* to move by paragraph.

5.1.7 Service Management

Definition of Windows Services:

- Windows Services are a core component of the Microsoft Windows operating system and enable the creation and management of long-running processes.
- Unlike regular software that is launched by the end user and only runs when the user is logged on, Windows Services can start without user intervention and may continue to run long after the user has logged off. The services run in the background and will usually kick in when the machine is booted.
- Developers can create Services by creating applications that are installed as a Service, an option ideal for use on servers when long-running functionality is needed without interference with other users on the same system.
- The services manage a wide variety of functions including network connections, speaker sound, data backup, user credentials and display colors. Windows Services perform a similar function as UNIX daemons.

Windows Services Control Manager:

- Windows Services are managed via the Services Control Manager panel. The panel shows a list of services and for each, name, description, status (running, stopped or paused) and the type of service. Double clicking on a service reveals its properties in greater detail. You can stop, pause, start, delay start, or resume each service as appropriate. You can also modify the start mechanism (Manual or Automatic) or specify an account.
- Windows Services broadly fall into three categories depending on the actions and applications they control: Local Services, Network Services and System. Third party applications such as antivirus software may also install their own services.
- Services can be deleted by a user with administrative privileges, but as doing so can render the operating system unstable, it should be done only when necessary and with caution.

Open Windows Services Manager:

- Windows operating system provides services in order to complete tasks in the foreground. There may become different types of job services in order to accomplish tasks like printing, network authentication, encryption, etc. Windows provides a lot of services by default. But more services can be added by third-party applications or other tools.
- Services Manager is used to managing operating systems services. There are different ways to open the Services Manager.
  1. By writing Services to the search we will be listed the Services Manager. This will work with Recent Windows Operating systems like Windows 8, Windows 10, Windows Server 2012, Windows Server 2016.
  2. More compatible way to open Service Manager is running services.msc command in the Run . This will work all windows versions like Windows XP, Windows Vista, Windows 8, Windows 10, Windows Server 2003, Windows Server 2008, Windows Server 2012, Windows Server 2016.

5.1.8 Syncing Devices and File Sharing

5.1.8.1 Syncing Devices

Sync settings on Windows:

- When Sync settings is turned on, **Windows syncs** the settings you choose across all your Windows 10 devices that you've signed in to with your Microsoft account.
- To find Sync settings, select the **Start** button, then select **Settings > Accounts > Sync your settings**.
- To stop syncing your settings and remove them from the cloud, turn off synced settings on all the devices connected to your Microsoft account, and then go to the **Devices page**, select **More actions** for the device you want to manage, and then select **Remove cloud backup of personal settings**.

- Choose from settings such as language preferences, passwords, and color themes. If you turn on Other Windows settings, Windows also syncs some device settings (for things like printers and mouse options), File Explorer settings, and notification preferences.

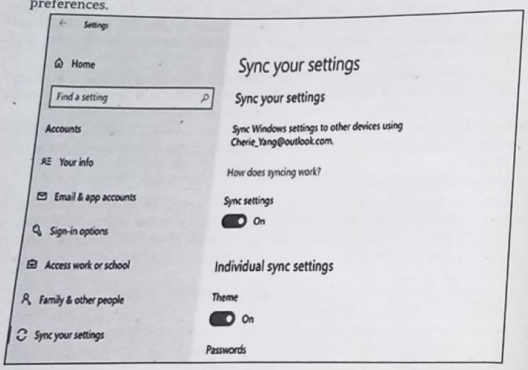


Fig. 5.13: Sync settings

- Synchronize your Windows settings and data files across multiple computers so that changes made on one computer automatically update all your computers.
- Syncing settings: To sync your Windows settings, on your primary Windows computer search for Settings, and from the Settings window select Accounts, Sync your settings to display the dialog box pictured at right, and then set all of the items you wish to sync to the ON position.
- When Sync Settings is turned on, Windows automatically syncs the settings you choose across all your Windows 10 desktops and laptops that you sign in to using your same Microsoft account. A listing of most of the Windows setting categories along with example settings that can be synced across your computers.

Synchronize your devices using local Wi-Fi Sync:

- The Local sync option allows you to synchronize your encrypted Sticky Password database between any two of your devices over your local Wi-Fi or other local network (Wired LAN).

- Sticky Password offers two types of secure data synchronization:
  1. **Synchronization via the cloud:** Allows you to backup and sync your encrypted data over our secure cloud servers.
  2. **Local synchronization:** sync your devices over your local Wi-Fi or other local network: your encrypted data never leaves your network.

5.1.8.2 File Sharing

File permissions to files or folders on a storage volume that you format with **NTFS** or **ReFS**.

- The permissions that you assign to files and folders govern user access to them.
- There are several key points to remember, with respect to file permissions, including that you can:
  - Configure file permissions for an individual file or folder, or sets of files or folders.
  - Assign file permissions individually, to objects that include users, groups, and computers.
  - Control file permissions by granting or denying specific types of file and folder access, such as Read or Write.
  - Configure inheritance of file permissions from parent folders. By default, the file permissions that you assign to a folder also are assigned to new folders or files within that parent folder.

Simple steps to implement file sharing permissions in Windows Server 2012:

1. Structure your Folder, for example in our case, we have few folder in our **E: drive the 1 i will use is OSI IT Tech DATA folder** for this demo.
2. Now lets **configure file permissions on the our existing folder**, what we going to do next is to restrict access to the Training folders.
   Right-click the Training folder, and then click **Properties**, then in the Training Properties dialog box, click **Security**, and then click **Advanced**.
3. In the Advanced Security Settings for Training dialog box, click **Disable Inheritance**.
4. In the Block Inheritance dialog box, click **Convert inherited permissions into explicit permissions on this object**.
5. **Remove the 2 permissions** entries for Users (SVR01\Users).
6. Then click **OK**.
7. On the **Security** tab, click **Edit**.
8. In the Permissions for Training dialog box, click **Add**.
9. In the Select Users, Computers, Service Accounts, or Group dialog box, type Training, then click OK.
10. In the Permissions for Training dialog box, under **Allow**, select **Modify** permission.

- **Work Folders** is a quite interesting feature introduced by **Microsoft** with **Windows Server 2012:**
  - Access the synced data from Windows PCs or from mobile devices.
  - Set up the server side of Work Folders and create a simple Sync Share.
  - To create the **Sync Share**. Go to the File and Storage Services area of the Server Manager and click on TASKS. Select New Sync Share.
  - Click on Next. Specify the local path – must be NTFS – where the data will be synced:
    - ✓ Define folder structure and name of the Sync Share
    - ✓ It's time to select the users or the groups who will be able to access the Sync Share
    - ✓ You can also encrypt the data if you desire

Steps to sync files between Windows Servers:

Step 1 : **Create a shared folder** and set up permission for the servers you want to share files with.

Step 2 : **Download** AOMEI Backupper Server, install and launch it. Then, click **Sync** and **Basic Sync** subsequently. To sync the changed files immediately, you can choose **Real-Time Sync**.

Step 3 : Click **+ Add Folder** to select files or folders you want to sync. Then click the folder shaped button > Share/NAS > Add Network Location and select the shared folder as destination.

Step 4 : To automatically sync your files, click **Schedule** and select one or more settings. After that, click "**Start Sync**".

- **Daily/weekly/monthly**: To sync files regularly always give users the latest file versioning and avoid data loss owing to forgetting to synchronize.
- **Event triggers**: To sync files at a specific event, you can enable event triggers and then choose between system shutdown, user logoff, system start up and user logon.
- **USB plug in**: To sync files more flexible, you can prepare a USB drive and create a schedule backup with "USB plug in" feature, then this software will sync files from or to the flash drive when connected to your computer.

5.1.9 Windows Utilities

- **Windows Utility** programs come with the OS and can help you better control your system so it is optimized specifically for you. Although other utilities like anti-virus, backup software, disk managers or memory testers can also be useful to you.

5.1.9.1 Accessories

- One of the most useful features Windows offers is the ability to use data created in one file in another file, even if the two files were created in different Windows

programs. To work with more than one program or file at a time, you simply need to open them on your desktop. A program button on the taskbar represents any window that is open on the desktop. When you want to switch from one open window to another, click the program button on the taskbar. If you tile, or arrange open windows on the desktop so that they are visible, you can switch among them simply by clicking in the window in which you want to work.

Table 5.1: Program and its Function

Program	Description
Calculator	Performs arithmetic calculations
Command Prompt	Executes MS-DOS commands
Internet Explorer	Displays web (HTML) pages
Notepad	Creates, edits, and displays text only documents
Paint	Creates and edits bitmap pictures
Snipping Tool	Captures different parts of the screen
Sound Recorder	Creates and plays digital sound files
Sticky Notes	Creates color notes on the screen
Windows DVD Maker	Burns pictures and videos to DVDs
Windows Fax and Scan	Sends and receives faxes or scanned pictures and documents
Windows Live Messenger	Sends and receives instant messages to online contacts; you need to download the program from the web (http://download.live.com)
Windows Live Mail	Provides e-mail, newsgroup, and directory services; you need to download the program from the web (http://download.live.com)
Windows Live Movie Maker	Creates movies using audio and video files; you need to download the program from the web (http://download.live.com)
Windows Live Photo Gallery	Views, edits, organizes, and shares photos and videos; you need to download the program from the web (http://download.live.com)
Windows Live Writer	Creates blogs with text, photos, and videos; you need to download the program from the web (http://download.live.com)
Windows Media Center	Provides entertainment options for digital and on-demand media
Windows Media Player	Plays sound, music, and video
WordPad	Creates, edits, and displays text, Rich Text Format, and Word documents

5.1.9.2 Disk Management

- **Disk Management** is a system utility in Windows that enables you to perform advanced storage tasks.
- Here are some of the things Disk Management is good for:
  - To setup a new drive, see Initializing a new drive.
  - To extend a volume into space that's not already part of a volume on the same drive, see Extend a basic volume.
  - To shrink a partition, usually so that you can extend a neighboring partition.
  - To change a drive letter or assign a new drive letter.

5.1.9.3 Resource Monitor

- **Resource Monitor**, a utility in Windows Vista and later, displays information about the use of hardware (CPU, memory, disk, and network) and software (file handles and modules) **resources** in real time.
- When Windows slows down or freezes, we usually turn to the Task Manager to figure out what is going on. If you need a better understanding of the way Windows and its apps use the resources of your computer, Resource Monitor (resmon.exe) is the right tool for the task. The information is concise and real-time with graphs and grouping by resources. You can monitor the use of the CPU, memory, disk, and network.

To start the Resource Monitor in Windows:

- An easy way to start Resource Monitor that works in all Windows versions is to search for the name of its executable file resource monitor or for Resource Monitor, and click or tap the search result with the same name.
- When you open the Resource Monitor (resmon), the application window is filled by the Overview tab. The Overview tab shows the CPU section by default. You can see that there are other tabs for Memory, Disk, and Network. In the lower part of the left-hand panel, there are collapsed sections for Disk, Network, and Memory.
- You can expand the Resource Monitor window to full-screen size so you can see more of what's going on. You can also drag the bottom border of the individual windows on the left to show more or less data at a time.
- Each window shows a list of programs that are currently using resources, and the graphs on the right give you a visual representation of the totals. You can change the size of the graphs, by clicking or tapping on the Views button. The default view is "Large."
- Watch the changing lists and graphs and see which programs are using your resources. We found it interesting that opera.exe (which is the executable of the Opera web browser).

5.1.9.4 Backup and Recovery

- Step-by-step procedure to backup & restore using System State. It is very important to know that you can restore a system state backup to the same physical computer from which the system state backup was created.

Backup:

- Windows Server Backup replaces the good old NTBACKUP.EXE. This software is not installed by default: you must install it by using the Add Features option in Server Manager before you can use or from the Power Shell prompt, execute the following commands.

```
add-windowsfeature windows-server-backup -includeallsubfeature
```

- In general, make sure you have a volume, or disk or network share designated to be the backup destination of the other than your C: drive.
- Backup destination of the other than your C: drive.: in this example we use E:.
  1. Using the Graphical User Interface from Server Manager select Tools
  2. Click Windows Server Backup
  3. Select Local Backup
  4. Click Backup Once
  5. Click Next in Getting Started screen
  6. Click Custom and then click Next
  7. Click Add Items in Select Items for Backup screen
  8. Select System State and then click OK
  9. Click Next in Select Items for Backup screen
  10. Select Local Drivers and then Click Next
  11. Select the backup destination in Specify Destination and then click Next
  12. Click Backup

Restore:

1. Install a basic windows installation in usual way, and apply all windows updates, install all driver and all the application supported on the server, and then restart Windows.
   This installation must be similar to the original: recreate all partitions, with the same drive letters, as that were present on the system you are trying to restore, same language, etc.
2. Add the role "Active Directory Domain Services" and "Windows Server Backup".
3. Reboot in Directory Services Restore Mode *(Tools -> System Configuration -> Boot -> Select* Safe Boot *and* Active Directory Repair).
4. Login and launch the Restore procedure. Using the Graphical User Interface from Server Manager select *Tools -> Windows Server Backup -> Local Backup -> Recover.*
5. Select A Backup stored on another location & Next.
6. Select Local drive & Next.

7. In backup location select E: & Next.
8. Select the name of the server to restore.
9. Select the date to restore & Next.
10. Select System State & Next.
11. Select Original Location and Perform an authoritative restore of Active Directory Files & Next.
12. Confirm all the warnings and at the end click on Restore: the restore process begin. It can take several hours to end.
13. When finished restart Windows.
14. At the reboot you have to logon using Active Directory Service Restore Mode Password: click on the Switch User and then Other Password, and use like User Name .\Administrator, and like password the Active Directory Service Restore Mode Password.
15. Toggle to normal boot: Remove the flag to boot in Directory Services Restore Mode (normal boot): *Tools -> System Configuration -> Boot -> Remove* the flag on Safe Boot.

5.1.10 Basic Troubleshooting

- **Troubleshooting** is the process of diagnosing the source of a problem. It is used to fix problems with hardware, software, and many other products. The basic theory of troubleshooting is that you start with the most general (and often most obvious) possible problems, and then narrow it down to more specific issues.
- **The six steps of troubleshooting.**
  1. Identify the problem.
  2. Establish a theory of probable cause.
  3. Test probable cause theory to determine actual cause.
  4. Establish an action plan and execute the plan.
  5. Verify full system functionality.
  6. Document the process.

5.1.10.1 Network Troubleshooting

- A faulty Wi-Fi connection doesn't have to ruin your day. There are plenty of ways you can restore a lost internet connection. Follow these network troubleshooting tips and you'll be up and running in no time.

Open Windows Network Diagnostics:

- Windows has a tool called Windows Network Diagnostics that lets users troubleshoot connection issues.
- Go to *Windows Settings > Network & Internet > Status*. Under Change Your Network Settings, click *Network Troubleshooter*.

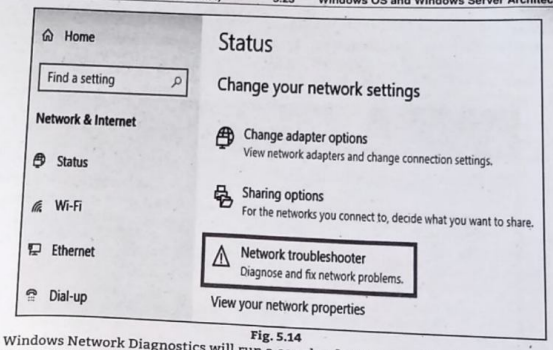


Fig. 5.14

- Windows Network Diagnostics will run a couple of tests to see what's possibly causing your Wi-Fi issues.

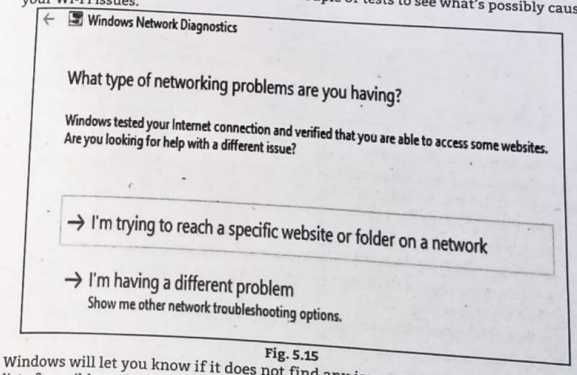


Fig. 5.15

- Windows will let you know if it does not find any issue. Otherwise, you will be given a list of possible actions to take to resolve the problem.
- This tool, or a version of it, should be available in Windows 7 to Windows 10.

5.1.10.2 Security Troubleshooting

- **Windows Security Troubleshooter** from Microsoft Automated Troubleshooting Services troubleshoots, identifies and fixes Windows security problems automatically. If you are facing any security-related issues on your computer, you definitely want to run it and see if it can help fix your problems.

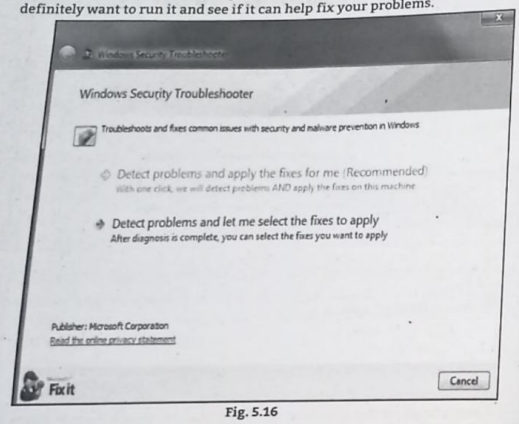


Fig. 5.16

Windows Security Troubleshooter:

- The Troubleshooter will automatically diagnose and fix the following problems with security in Windows, such as:
  - User Account Control – UAC
  - Windows Firewall
  - Data Execution Protection (DEP) to prevent security threats.
  - The Troubleshooter will automatically check Windows security features and enable them if needed.
  - The Troubleshooter will automatically fix Phishing or SmartScreen filters.
  - The Troubleshooter will automatically fix and reset User Account Control (UAC).
  - The Troubleshooter will automatically fix Data Execution Prevention (DEP).
  - The Troubleshooter will automatically reset and fix Windows Firewall.

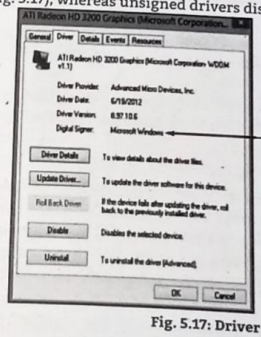
5.1.10.3 Device Driver

Troubleshooting Device Driver Problems:

- Device drivers are small chunks of software that Windows uses to contact and control (that is, "drive") your PC's hardware.
- Other than problems with the hardware itself, device drivers are the cause of most device woes. This is true even if your device doesn't have one of the problem icons mentioned in the preceding section. That is, if you open the device's properties sheet, Windows might tell you that the device is "working properly," but all that means is that Windows can establish a simple communications channel with the device.

Basic Device Driver Troubleshooting:

- Here are a few basic techniques for correcting device driver problems:
  - **Reinstall the driver:** A driver might be malfunctioning because one or more of its files have become corrupted. You can usually solve this problem by reinstalling the driver. Just in case a disk fault caused the corruption, you should check the hard drive where the driver is installed (usually drive C) for errors before reinstalling.
  - **Upgrade to a signed driver:** Unsigned drivers—that is, device drivers that don't come with a security signature from Microsoft that verifies the drivers are safe to install—are accidents waiting for a place to happen in Windows, so you should upgrade to a signed driver, if possible. How can you tell whether an installed driver is unsigned? In Device Manager, double-click the device to open its Properties dialog box and then display the Driver tab. Signed driver files display a name (such as "Microsoft Windows") beside the Digital Signer label (see Fig. 5.17), whereas unsigned drivers display "Not digitally signed" instead.



The name of the digital signer appears here

Fig. 5.17: Driver tab

- **Disable an unsigned driver:** If an unsigned driver is causing system instability and you can't upgrade the driver, try disabling it. In Device Manager, double-click the device to open its Properties dialog box, click the Driver tab, and then click Disable.
- **Try the manufacturer's driver supplied with the device:** If the device came with its own driver (say, on a CD or DVD), either try updating the driver to the manufacturer's or try running the device's setup program from the disc.
- **Download the latest driver from the manufacturer:** Device manufacturers often update drivers to fix bugs, add new features, and tweak performance. Go to the manufacturer's website to see whether an updated driver is available.
- **Roll back a driver:** If the device stops working properly after you update the driver, try rolling it back to the old driver.

5.2 INTRODUCTION TO UBUNTU

5.2.1 Introduction

- Ubuntu is a Linux-based operating system. It is designed for computers, smartphones, and network servers. The system is developed by a UK based company called Canonical Ltd. All the principles used to develop the Ubuntu software are based on the principles of Open Source software development.
- Following are some of the significant features of Ubuntu:
  - The desktop version of Ubuntu supports all the normal software on Windows such as Firefox, Chrome, VLC, etc.
  - It supports the office suite called LibreOffice.
  - Ubuntu has in-built email software called Thunderbird, which gives the user access to email such as Exchange, Gmail, Hotmail, etc.
  - There are a host of free applications for users to view and edit photos.
  - There are also applications to manage videos and it also allows the users to share videos.
  - It is easy to find content on Ubuntu with the smart searching facility.
  - The best feature is, it is a free operating system and is backed by a huge open source community.

5.2.2 Overview of Kernel

- The kernel is the heart of the system. It manages the communication between the underlying hardware and the peripherals. The kernel also makes sure that processes and daemons (server processes) are started and stopped at the exact right times. The kernel has a lot of other important tasks, so many that there is a special kernel-development mailing list on this subject only, where huge amounts of information are shared.
- The kernel manages the system resources, including file systems, processes, and physical devices.

- The kernel provides applications with system services such as I/O management, virtual memory, and scheduling. The kernel assigns priorities, services resource requests, and services hardware interrupts and exceptions. The kernel schedules and switches threads, pages memory, and swaps processes.
- The kernel has four jobs:
  1. **Memory management:** Keep track of how much memory is used to store what, and where.
  2. **Process management:** Determine which processes can use the Central Processing Unit (CPU), when, and for how long.
  3. **Device drivers:** Act as mediator/interpreter between the hardware and processes.
  4. **System calls and security:** Receive requests for service from the processes.

Kernel:

- The kernel is the software that directly manages your hardware, allowing application libraries and software like GNOME and Firefox to run on many types of hardware without much difficulty. Because the Linux kernel is the core component of a GNU/Linux system, a full restart is required to complete the kernel update.

Types of Kernel:

- Ubuntu packages the Linux kernel for a variety of architectures, including several variants of the x86 architecture. These include a 386 version, a 686 version, and versions for the AMD k6 and k7 processors. While most software for x86 processors in Ubuntu is compiled for 386 or better instruction sets, the kernel and a few other packages are specifically compiled for certain processors for speed reasons. Check the package documentation to determine what type of kernel will perform best for your processor.

Versions:

- Ubuntu currently packages the 3.8 kernel for optimal desktop speed and features.

SMP:

- Some motherboards have more than one processor on them, and some processors have multiple cores. If your computer is like this, then the SMP kernel is for you. Non-SMP kernels will not be able to take advantage of your multiple processors. However, if you do not have multiple processors, the additional code in an SMP kernel will only slow you down. Naturally, Ubuntu provides both SMP and non-SMP kernels for all supported architectures.

PAE:

- PAE (Physical Address Extension) allows the 32 bit version of Ubuntu to access up to 64 GB of memory and is the standard for all members of the Ubuntu family from release 12.10 and beyond, as the non-PAE version has been dropped.
- To Upgrade or Recompile Kernel.

- The precompiled kernels that are supplied with your distro should be fine however if you wish to update or optimise (or standardise) for your platform:
  1. You can Kernel/Upgrade easily using Ubuntu.
  2. You can also Kernel/Compile it yourself from Linux Kernel Source or Ubuntu Kernel Source.
  3. Run the following Terminal commands to install a new Ubuntu kernel from

```
http://kernel.ubuntu.com/~kernel-ppa/mainline/
sudo apt-get update
sudo apt-get install python-bs4 python-apt
cd /tmp; rm -rf medigeek-kmp*; wget --no-check-certificate
https://github.com/medigeek/kmp-downloader/tarball/master -O
kmpd.tar.gz; tar xzf kmpd.tar.gz; cd medigeek-*
python kmpd.py
```

  Just press <ENTER> instead of a number if you get stuck on a certain question in the python script.
- The script by default filters out (i.e. does not show) the release candidates. If you want the latest release candidates, please use: python kmpd.py -d

Remove unwanted Kernels from your System:

- Open the Synaptic package manager from the System->Administration menu. Click the "Search" button on the toolbar and search for "linux-image-2". The results should show every available and installed kernel. A green box on the left indicates that the package is installed. The only linux-image you want installed is the latest one. Find the package corresponding to the kernel to you running currently (this is the kernel you found in the terminal window). Make sure you keep that one. Now you can uninstall the old kernels from the list by clicking their boxes and selecting "Mark for Removal".
- Or you can uninstall kernels using the ubuntu-tweak Ubuntu PPA package.

Differences Between Kernel Modules and User Programs:

- **Kernel modules have separate address space.** A module runs in **kernel space**. An application runs in **user space**. System software is protected from user programs. Kernel space and user space have their own memory address spaces.
- **Kernel modules have higher execution privilege.** Code that runs in kernel space has greater privilege than code that runs in user space. Driver modules potentially have a much greater impact on the system than user programs. Test and debug your driver modules carefully and thoroughly to avoid adverse impact on the system.
- **Kernel modules do not execute sequentially.** A user program typically executes sequentially and performs a single task from beginning to end. A kernel module does not execute sequentially. A kernel module registers itself in order to serve future requests.

- **Kernel modules must be pre-emptible.** You cannot assume that your driver code is safe just because your driver code does not block. Design your driver assuming your driver might be pre-empted.
- **Kernel modules can share data.** Different threads of an application program usually do not share data. By contrast, the data structures and routines that constitute a driver are shared by all threads that use the driver. Your driver must be able to handle contention issues that result from multiple requests. Design your driver data structures carefully to keep multiple threads of execution separate. Driver code must access shared data without corrupting the data.

5.2.3 Installation of Ubuntu

- Follow the steps below to install Ubuntu in dual boot with Windows:
- Install Ubuntu in dual boot with Windows 10 & Windows 8

Step 1 : **Create a live USB or disk**

Download and create a live USB or DVD

Step 2 : **Boot in to live USB**

Plug the live USB or disk in to the computer and restart the computer. While booting the computer press F10 or F12 function key (defers from computer to computer) to go to the boot menu. Now, choose the option to boot from **USB** or **Removable Media**.

Step 3 : **Start the installation**

It will take some time to boot in to the live USB or disk. Once booted, you will be immediately provided with option to either try Ubuntu or install Ubuntu. Even if you choose to try, you can find the option to install on the desktop:

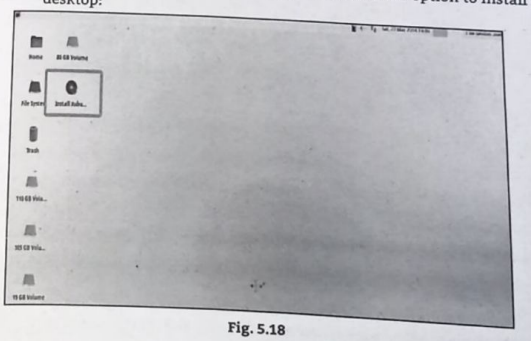


Fig. 5.18

First few screens are pretty straight forward. Just choose press continue:

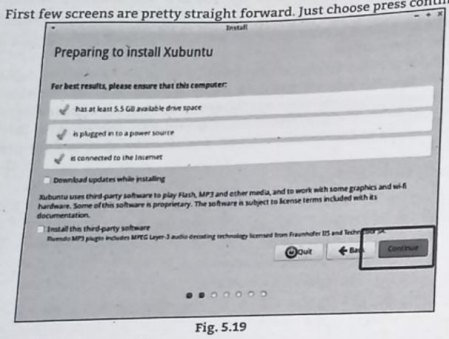


Fig. 5.19

Step 4 : Prepare the partition

This is the most important part of the whole dual boot installation. Where to install Ubuntu? Windows is already installed here, so, we'll prepare a new partition for Ubuntu. In the Installation Type window, choose **Something Else**.

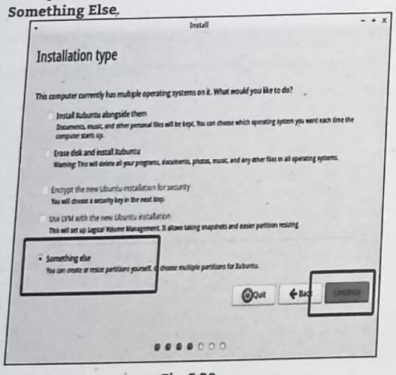


Fig. 5.20

As you can see, I have 3 NTFS and some ext4 partitions. If you don't have ext4 partition, don't worry, we don't need that. As you can see in the picture below, one of the NTFS partition consists of Windows installation. This should be untouched if you want to keep your Windows installation safe.

Click on the desired partition and press the (-) to delete the partition.

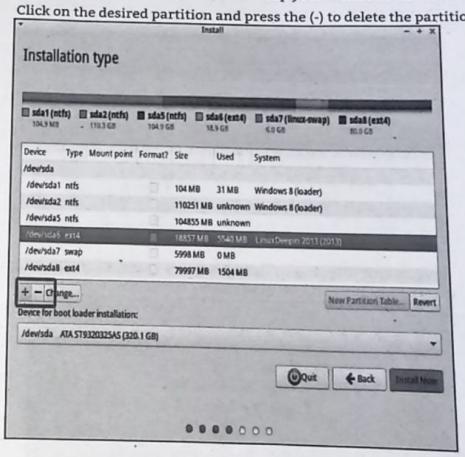


Fig. 5.21

Step 5 : Create root, swap and home

Once you have some free space on your hard drive, its time to install Ubuntu on it. Now, there are several ways to do it. But we prefer to a Root, a Swap and a Home.

The root should be at least 15 GB for a comfortable use. If you have more disk space, increase the root size. Suppose you have 100 GB of disk space. You can easily devote 30 GB of space to root.

Create a root partition first. Choose the free space available and click on (+).

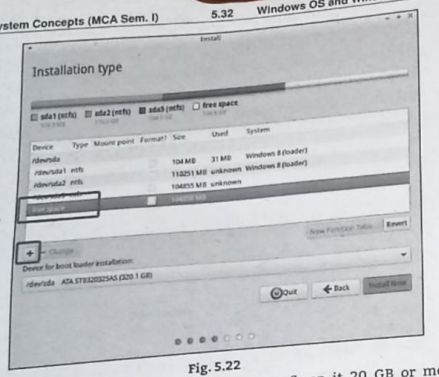


Fig. 5.22

Here, choose the size of root directory (keep it 20 GB or more), choose ext4 file system, and mount point as / (i.e. root):

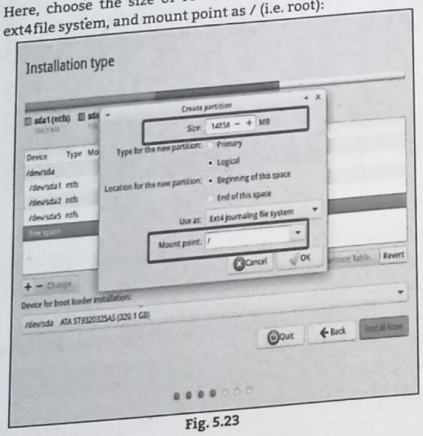


Fig. 5.23

Next step is to create swap partition. It is advised by many that Swap should be double of your system's RAM size. You can choose the swap size accordingly.

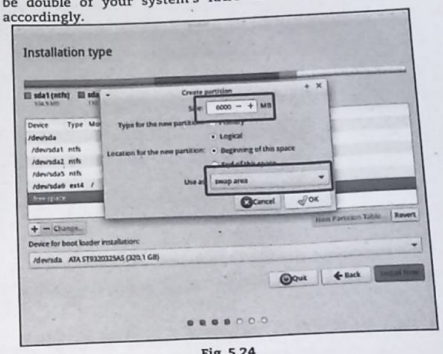


Fig. 5.24

The next step is to create Home. Try to allocate the maximum size to Home because this is where you'll be downloading and keeping the files.

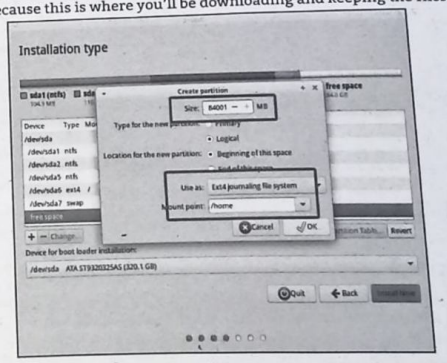


Fig. 5.25

Once you have created Root, Swap and Home partitions, click on Install Now button.

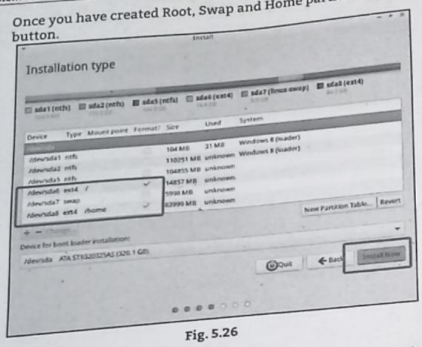


Fig. 5.26

Step 6 : Follow the trivial instructions

If you successfully created the partitions as mentioned above, you have almost won the battle. Rest is just some trivial steps to follow. You will be taken through a number of screens to select options like keyboard layout, login credentials etc.

Once the installation is over, you will be presented with the option to keep trying live version or to restart the system.

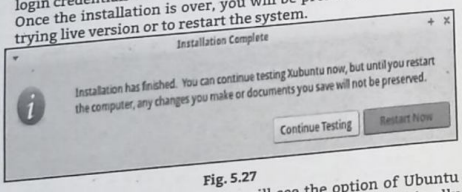


Fig. 5.27

And that's it. On next boot, you will see the option of Ubuntu in the grub screen. And thus you can enjoy the beautiful, beginner friendly and world's most popular Linux distribution.

5.2.4 File System

- Just like Windows, Ubuntu has a file system that is shared between software components and your own personal data, which you generate with your applications or download from the Internet. However, Ubuntu differs from Windows in a couple of important ways.

- **Note:** The Ubuntu file system is referred to as a Hierarchical file system. This means that it consists of a lot of directories that contain files. Windows also uses a hierarchical file system. Ubuntu refers to the very bottom level of the file system as the root. This has no connection with the root user or the directory named/root, which is the personal file storage area for the root user.

Table 5.2: Directories and Files in the Ubuntu Root File System

Directory	Contents
bin	Vital tools necessary to get the system running or for use when repairing the system and diagnosing problems.
boot	Boot loader programs and configuration files, (The boot loader is the menu that appears when you first boot Linux.)
cdrom → media/cdrom	Symbolic link (shortcut) to the entry for the CD- or DVD-ROM drive in the /dev folder. (Accessing this file lets you access the CD-or DVD-ROM drive.)
dev	Virtual files representing hardware installed on your system.
etc	Central repository of configuration files for your system.
lib	Shared system files used by Linux as well as the software that runs on it.
lost-found	Folder where salvaged scraps of files are saved in the event Ta problematic shutdown and subsequent file system check.
media	Where the directories representing various mounted storage systems are made available including Windows partitions on the disk).
mnt	Directory in which external file systems can be temporarily mounted.
opt	Software that is theoretically optional and not vital to the running of the system. (Many software packages you use daily can be found here.)
proc	Virtual directory containing data about your system and its current status.
root	The root user's personal directory.
sbin	Programs essential to administration of the system.
selinux	Commands used in the SELinux security subsystem.
srv	Configuration files for any network servers you might have running on your system.
sys	Mount point of the sysfs file system, which is used by the kernel to administer your system's hardware.
tmp	Temporary files stored by the system.
usr	Programs and data that might be shared with other systems (such as in a large networking setup with many users).

Types of File Systems:

- Linux all about choice and this extends the technology that makes the file system work. Unlike with Windows, where the only real choice these days NTFS, Linux offers many types file system. The basic features every file system is present all these types, but each optimized for different set tasks. Most are scalable, however, which means that they will work just as happily desktop on massive cluster of computers.
- Ubuntu uses the ext4 file system. This popular choice among distros, and nearly office-oriented distros use it. That said, people are constantly arguing about which file system best. The principal measuring stick performance. Your computer spends lot time writing and reading files, so the faster file system the faster your PC will overall (although, reality, the hardware equal importance).
- Here are the various types of file:
  - **ext4:** Understandably, and logically, this an extension ext3. Among other things, features support for much larger hard disks and also faster.
  - **reiserfs:** This is another journaling file system, which claims be faster than others and also offers better security features. It has fallen out favor recent years.
  - **jfs:** This is a journaling file system created by IBM. It's used on industrial implementations of UNIX.
  - **xfs:** This is 64-bit journaling file system created by Silicon Graphics, Inc. (SGI) and used on its own version of UNIX as well as Linux.
  - **zfs:** Another new file system technology (like ext4), its main benefit is support for huge storage systems. This is because of its 128-bit approach. It is used in the Sun Microsystems Solaris and OpenSolaris operating systems.

5.2.5 Basic Commands of Linux

- Linux is an operating system's kernel. Linux is a UNIX clone. But it was actually created by Linus Torvalds from Scratch. Linux is free and open-source, that means that you can simply change anything in Linux and redistribute it in your own name! There are several Linux Distributions, commonly called “distros”.

List of Linux Distros:

- Ubuntu Linux
- Red Hat Enterprise Linux
- Linux Mint
- Debian
- Fedora

Basic Commands:

1. **man command:** To know more about a command and how to use it, use the man command. It shows the manual pages of the command. For example, “man cd” shows the manual pages of the cd command. Typing in the command name and the argument helps it show which ways the command can be used (e.g., cd –help).


```
TOUCH(1)                    User Commands                    TOUCH(1)

NAME
       touch - change file timestamps

SYNOPSIS
       touch [OPTION]... FILE...

DESCRIPTION
       Update  the  access  and modification times of each FILE to the current
       time.

       A FILE argument that does not exist is created empty, unless -c  or  -h
       is supplied.

       A  FILE  argument  string of - is handled specially and causes touch to
       change the times of the file associated with standard output.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a     change only the access time

Manual page touch(1) line 1 (press h for help or q to quit)
```

Fig. 5.28: man command

1. **pwd command:** $ pwd displays the path of your current working directory. It gives us the absolute path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash(/). The user directory is usually something like "/home/username".

```
nayso@Alok-Aspire:~$ pwd
/home/nayso
```

Fig. 5.29: pwd command

2. **ls command:** Use the **"ls"** command to know what files are in the directory you are in. You can see all the hidden files by using the command **“ls -a”**.

```
nayso@Alok-Aspire:~$ ls
Desktop              itsuserguide.desktop  reset-settings     VCD_Copy
Documents            Music                 School_Resources   Videos
Downloads            Pictures              Students_Works_10
examples.desktop     Public                Templates
GplatesProject       Qgis Projects         TuxPaint-Pictures
```

Fig. 5.30: ls command

3. **cat command:** The cat command is used to print the contents of one or more files. It is usually used to easily view programs.

```
nayso@Alok-Aspire:~/Desktop$ echo hello, my name is alok >> new.txt
nayso@Alok-Aspire:~/Desktop$ cat new.txt
hello, my name is alok
nayso@Alok-Aspire:~/Desktop$ echo this is another line >> new.txt
nayso@Alok-Aspire:~/Desktop$ cat new.txt
hello, my name is alok
this is another line
```

Fig. 5.31: cat command

4. **touch command:** The **touch** command is used to create a file. It can be anything, from an empty txt file to an empty zip file. For example, "**touch new.txt**".

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ touch new.txt
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
```

Fig. 5.32: touch command

5. **mkdir & rmdir command:** Use the **mkdir** command when you need to create a folder or a directory. For example, if you want to make a directory called "DIY", then you can type "**mkdir DIY**". Remember, as told before, if you want to create a directory named "DIY Hacking", then you can type "mkdir **DIY\ Hacking**".

Use **rmdir** to delete a directory. But **rmdir** can only be used to delete an empty directory. To delete a directory containing files, use **rm**.

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ mkdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
DIY
nayso@Alok-Aspire:~/Desktop$ rmdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

Fig. 5.33: mkdir and rmdir command

6. **rm command:** Used to remove files and directories. Use "**rm -r**" to delete just the directory. It deletes both the folder and the files it contains when using only the **rm** command.


```
nayso@Alok-Aspire:~/Desktop$ ls
newer.py  New Folder
nayso@Alok-Aspire:~/Desktop$ rm newer.py
nayso@Alok-Aspire:~/Desktop$ ls
New Folder
nayso@Alok-Aspire:~/Desktop$ rm -r New\ Folder
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

Fig. 5.34: rm command

7. **cp command:** Use the **cp** command to copy files through the command line. It takes two arguments: The first is the location of the file to be copied, the second is where to copy.

```
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ cp new.txt /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
new.txt
```

Fig. 5.35: cp command

8. **mv command:** Use the **mv** command to move files through the command line. We can also use the **mv** command to rename a file. For example, if we want to rename the file "**text**" to "**new**", we can use "**mv text new**". It takes the two arguments, just like the **cp** command.

```
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
nayso@Alok-Aspire:~/Desktop$ mv new.txt newer.txt
nayso@Alok-Aspire:~/Desktop$ ls
newer.txt
```

Fig. 5.36: mv command

9. **cd command:** Use the "**cd**" command to go to a directory. For example, if you are in the home folder, and you want to go to the downloads folder, then you can type in "**cd Downloads**". Remember, this command is case sensitive, and you have to type in the name of the folder exactly as it is.

```
nayso@Alok-Aspire:~$ cd Downloads
nayso@Alok-Aspire:~/Downloads$ cd
nayso@Alok-Aspire:~$ cd Raspberry\ Pi
nayso@Alok-Aspire:~/Raspberry Pi$ cd ..
nayso@Alok-Aspire:~$
```

Fig. 5.37: cd command

5.2.6 Managing Process in Linux

- An instance of a program is called a Process. In simple terms, any command that you give to your Linux machine starts a new process.

Types of Processes:

- **Foreground Processes:** They run on the screen and need input from the user. For example Office Programs.
- **Background Processes:** They run in the background and usually do not need user input. For example Antivirus.

- The Linux terminal has a number of useful commands that can display running processes, kill them, and change their priority level.
- Many of the commands here perform a single function and can be combined — that's the Unix philosophy of designing programs. Other programs, like top, provide a friendly interface on top of the commands.

1. **Fg:**

- It use the command "fg" to continue a program which was stopped and bring it to the foreground.

Syntax: `fg jobname`

2. **Top:**

- This utility tells the user about all the running processes on the Linux machine.
- **top** command is the traditional way to view your system's resource usage and see the processes that are taking up the most system resources. Top displays a list of processes, with the ones using the most CPU at the top.
- To exit top or htop, use the **Ctrl-C** keyboard shortcut. This keyboard shortcut usually kills the currently running process in the terminal.

```
howtogeek@ubuntu: ~
top - 03:48:40 up 19 min,  1 user,  load average: 0.16, 0.09, 0.16
Tasks: 143 total,   1 running, 142 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.6%us,  0.7%sy,  0.0%ni, 96.7%id,  0.0%wa,  0.0%hi,  0.0%si,
Mem:   1025656k total,   678580k used,   347076k free,    79936k buffer
Swap:        0k total,        0k used,        0k free,   310528k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1216 root      20   0 32624 3460 2860 S  0.7  0.3   0:05.31 vmtoolsd
 2025 howtogee  20   0 81456  23m  17m S  0.7  2.3   0:01.41 unity-2d-p
   17 root      20   0     0    0    0 S  0.3  0.0   0:00.34 kworker/0:
   36 root      20   0     0    0    0 S  0.3  0.0   0:00.10 scsi_eh_1
 1081 root      20   0  199m  60m 7340 S  0.3  6.0   0:13.42 Xorg
 1973 howtogee  20   0 6568 2832  916 S  0.3  0.3   0:06.24 dbus-daemo
 2153 howtogee  20   0  147m  16m 9820 S  0.3  1.7   0:03.63 unity-pane
 2313 howtogee  20   0  136m  13m  10m S  0.3  1.4   0:00.84 gnome-term
 2697 howtogee  20   0 2820 1148  864 R  0.3  0.1   0:00.05 top
    1 root      20   0 3456 1976 1280 S  0.0  0.2   0:02.31 init
    2 root      20   0    0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0    0    0    0 S  0.0  0.0   0:00.07 ksoftirqd
```

Fig. 5.38: top command

3. **Ps:**

- This command stands for 'Process Status'. The following command lists all processes running on your system:

```
ps -A
```

```
howtogeek@ubuntu: ~
 2276 ?        00:00:00 unity-musicstor
 2313 ?        00:00:01 gnome-terminal
 2320 ?        00:00:00 gnome-pty-helpe
 2321 pts/0    00:00:00 bash
 2378 ?        00:00:00 update-notifier
 2425 ?        00:00:00 deja-dup-monito
 2441 ?        00:00:00 sh
 2442 ?        00:00:00 run-parts
 2448 ?        00:00:00 apt
 2550 ?        00:00:00 gvfsd-metadata
 2650 ?        00:00:12 firefox
 3177 ?        00:00:00 kworker/0:0
 3224 ?        00:00:00 kworker/0:1
 3248 ?        00:00:00 kworker/0:2
 3348 ?        00:00:39 update-apt-xapi
 3356 ?        00:00:01 oneconf-service
 3379 ?        00:00:00 cat
 3388 pts/0    00:00:00 ps
howtogeek@ubuntu:~$
```

Fig. 5.39: ps command

4. **pstree:**

- The **pstree** command is another way of visualizing processes. It displays them in tree format. So, for example, your X server and graphical environment would appear under the display manager that spawned them.

```
howtogeek@ubuntu: ~
├─gvfsd-trash
├─indicator-appli─┬─{indicator-appl}
├─indicator-datet─┬─2*[{indicator-date}]
├─indicator-messa─┬─{indicator-mess}
├─indicator-sessi─┬─2*[{indicator-sess}]
├─indicator-sound─┬─2*[{indicator-soun}]
├─lightdm─┬─Xorg
│         └─gnome-session─┬─applet.py
│                         ├─blueman-applet─┬─{blueman-applet}
│                         ├─bluetooth-apple─┬─{bluetooth-appl}
│                         ├─deja-dup-monito─┬─2*[{deja-dup-monit}]
│                         ├─gdu-notificatio
│                         ├─gnome-fallback-─┬─2*[{gnome-fallback}]
│                         ├─gnome-settings-─┬─2*[{gnome-settings}]
│                         ├─metacity─┬─{metacity}
│                         ├─nautilus─┬─3*[{nautilus}]
│                         ├─nm-applet─┬─{nm-applet}
│                         ├─polkit-gnome-au─┬─{polkit-gnome-a}
│                         └─ssh-agent
```

Fig. 5.40: pstree command

5. **Kill:**
- This command **terminates running processes** on a Linux machine.
- To use these utilities you need to know the PID (process id) of the process you want to kill.

Syntax: `kill PID`
- To find the PID of a process simply type,

```
pidof Process name
```

```
howtogeek@ubuntu: ~
howtogeek@ubuntu:~$ kill firefox
bash: kill: firefox: arguments must be process or job IDs
howtogeek@ubuntu:~$ ps -A | grep firefox
 3684 ?        00:00:10 firefox
howtogeek@ubuntu:~$ kill 3684
howtogeek@ubuntu:~$
```

Fig. 5.41: kill command

5. **NICE:**
- Linux can run a lot of processes at a time, which can slow down the speed of some high priority processes and result in poor performance.
- To prioritize processes as per your requirements. This priority is called Niceness in Linux, and it has a value between -20 to 19.
- The lower the Niceness index, the higher would be a priority given to that task.
- The default value of all the processes is 0.
- To start a process with a niceness value other than the default value use the following syntax:

```
nice -n 'Nice value' process name
```
- If there is some process already running on the system, then you can 'Renice' its value using syntax.

```
renice 'nice value' -p 'PID'
```
- To change Niceness, you can use the 'top' command to determine the PID (process id) and its Nice value.

6. **DF:**
- This utility reports the free disk space(Hard Disk) on all the file systems. If you need information in a readable format, then use the command 'df -h'

7. **Free:**
- This command shows the free and used memory (RAM) on the Linux system.
- free -m to display output in MB.
- free -g to display output in GB.

8. **renice:**
- The **renice** command changes the nice value of an already running process.
- The nice value determines what priority the process runs with. A value of **–19** is very high priority, while a value of **19** is very low priority. A value of **0** is the default priority.
- The renice command requires a process's PID. The following command makes a process run with very low priority: renice 19 PID
- If you're making a process run at a higher priority, you'll require root permissions. On Ubuntu, use **sudo** for that:

```
sudo renice -19 #
```

5.2.7 Installing and Deleting Software Packages

- Click on the Ubuntu Software icon in the Activities toolbar;
- This will open the Ubuntu Software manager through which you can search for, install and uninstall software from your computer.
- From the list of applications, look up for the one you want to uninstall and then click the Remove button against it

Installing & Deleting Software Packages:
- **Installing Package:** To Installing an rpm software package, use the following command with -i option.

Syntax: `rpm -i package_file_name`

Where, package_file_name specifies the name of the file that contains the package you want to install.

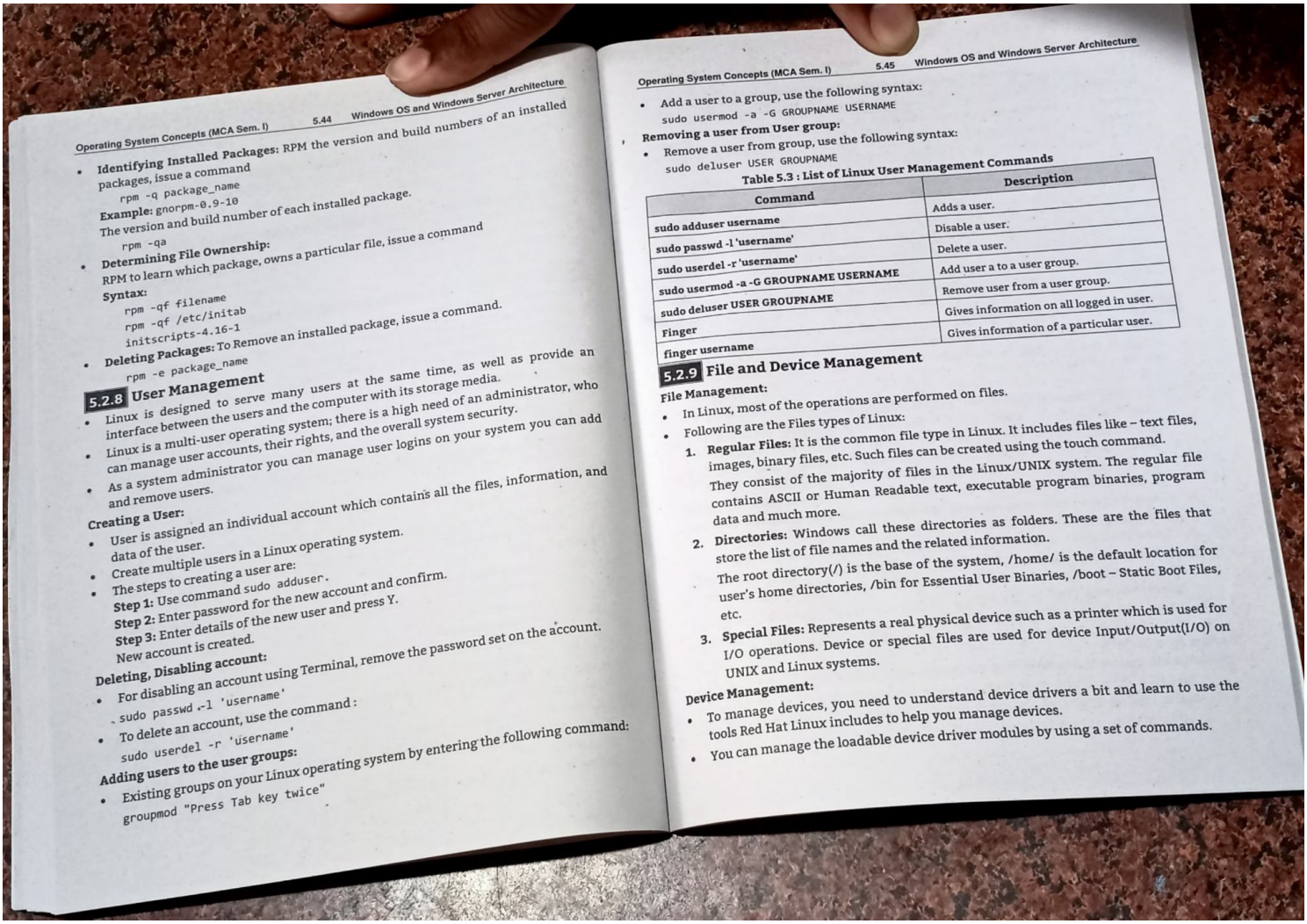
Options:
1. **-i :** Install a Package.
2. **-v :** Verbose for a nicer display.
3. **-h :** Print hash marks as the package archive is unpacked.

Example:
```
#rpm -ivh gnorom-0.-10.i386.rpm
Press Enter & Type
gnome-linuxconf-0.23-1.386.rpm
gnorpm
###############
gnome-linuxconf
###############
```
- **Performing Updates & Fixes:** rpm makes it easy to install new versions of package. issue a command,

```
rpm -uvh package_file_name
```

Where, package_file_name is the name of the file that contains the new version of the package.

Example: `rpm -Fvh gnorpm-0.9-10.i386.rpm`



- **Identifying Installed Packages:** RPM the version and build numbers of an installed packages, issue a command

```
rpm -q package_name
```

Example: gnorpm-0.9-10
The version and build number of each installed package.

```
rpm -qa
```

- **Determining File Ownership:**
RPM to learn which package, owns a particular file, issue a command
Syntax:

```
rpm -qf filename
rpm -qf /etc/initab
initscripts-4.16-1
```

- **Deleting Packages:** To Remove an installed package, issue a command.

```
rpm -e package_name
```

5.2.8 User Management

- Linux is designed to serve many users at the same time, as well as provide an interface between the users and the computer with its storage media.
- Linux is a multi-user operating system; there is a high need of an administrator, who can manage user accounts, their rights, and the overall system security.
- As a system administrator you can manage user logins on your system you can add and remove users.

Creating a User:

- User is assigned an individual account which contains all the files, information, and data of the user.
- Create multiple users in a Linux operating system.
- The steps to creating a user are:

Step 1: Use command `sudo adduser`.
Step 2: Enter password for the new account and confirm.
Step 3: Enter details of the new user and press Y.
New account is created.

Deleting, Disabling account:

- For disabling an account using Terminal, remove the password set on the account.

```
sudo passwd -l 'username'
```

- To delete an account, use the command :

```
sudo userdel -r 'username'
```

Adding users to the user groups:

- Existing groups on your Linux operating system by entering the following command:

```
groupmod "Press Tab key twice"
```


- Add a user to a group, use the following syntax:

```
sudo usermod -a -G GROUPNAME USERNAME
```

Removing a user from User group:

- Remove a user from group, use the following syntax:

```
sudo deluser USER GROUPNAME
```

Table 5.3 : List of Linux User Management Commands

Command	Description
sudo adduser username	Adds a user.
sudo passwd -l 'username'	Disable a user.
sudo userdel -r 'username'	Delete a user.
sudo usermod -a -G GROUPNAME USERNAME	Add user a to a user group.
sudo deluser USER GROUPNAME	Remove user from a user group.
Finger	Gives information on all logged in user.
finger username	Gives information of a particular user.

5.2.9 File and Device Management

File Management:

- In Linux, most of the operations are performed on files.
- Following are the Files types of Linux:

1. **Regular Files:** It is the common file type in Linux. It includes files like – text files, images, binary files, etc. Such files can be created using the touch command.
They consist of the majority of files in the Linux/UNIX system. The regular file contains ASCII or Human Readable text, executable program binaries, program data and much more.
2. **Directories:** Windows call these directories as folders. These are the files that store the list of file names and the related information.
The root directory(/) is the base of the system, /home/ is the default location for user's home directories, /bin for Essential User Binaries, /boot – Static Boot Files, etc.
3. **Special Files:** Represents a real physical device such as a printer which is used for I/O operations. Device or special files are used for device Input/Output(I/O) on UNIX and Linux systems.

Device Management:

- To manage devices, you need to understand device drivers a bit and learn to use the tools Red Hat Linux includes to help you manage devices.
- You can manage the loadable device driver modules by using a set of commands.

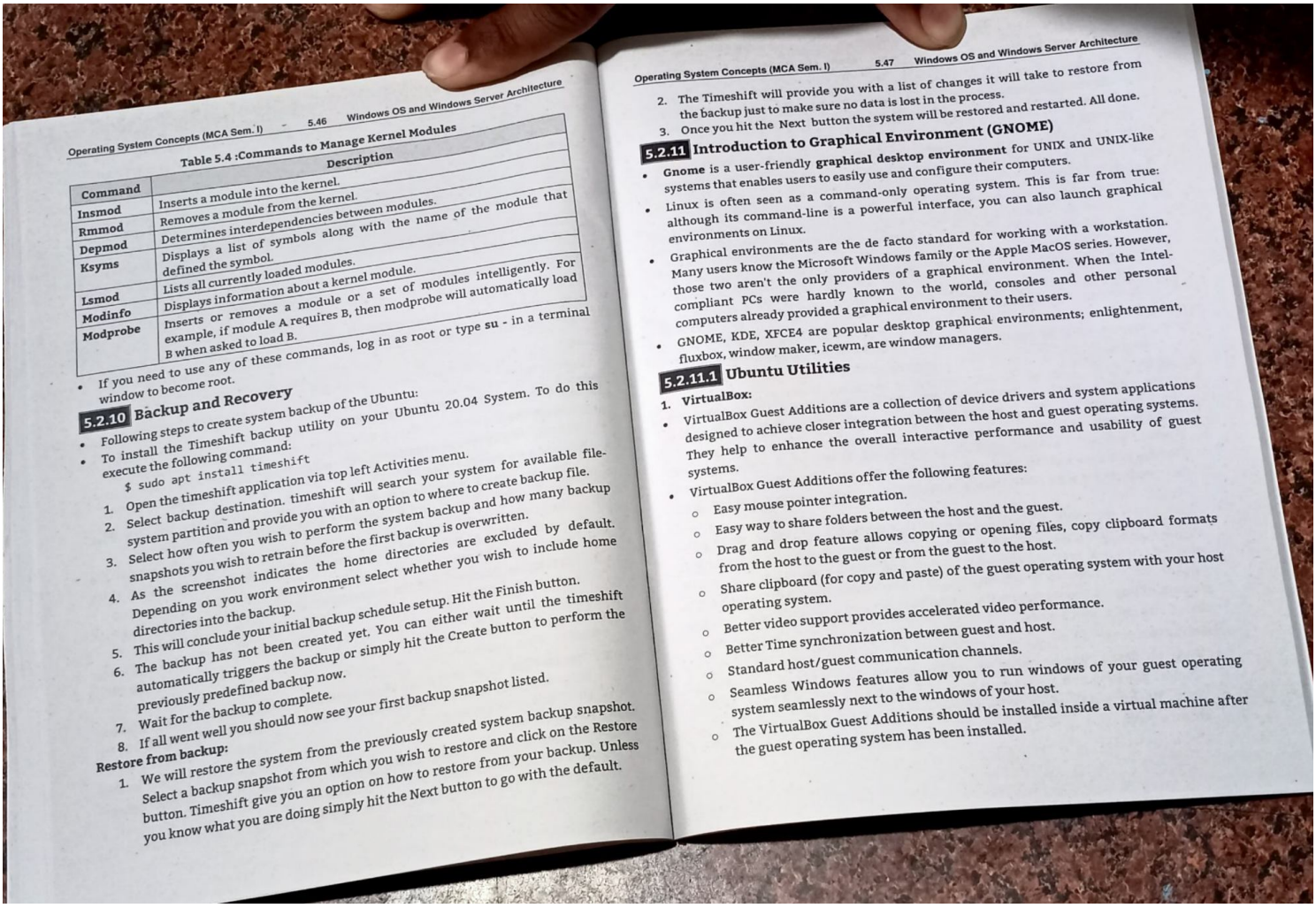


Table 5.4 :Commands to Manage Kernel Modules

Command	Description
Insmod	Inserts a module into the kernel.
Rmmod	Removes a module from the kernel.
Depmod	Determines interdependencies between modules.
Ksyms	Displays a list of symbols along with the name of the module that defined the symbol.
Lsmod	Lists all currently loaded modules.
Modinfo	Displays information about a kernel module.
Modprobe	Inserts or removes a module or a set of modules intelligently. For example, if module A requires B, then modprobe will automatically load B when asked to load B.

- If you need to use any of these commands, log in as root or type **su -** in a terminal window to become root.

5.2.10 Backup and Recovery

- Following steps to create system backup of the Ubuntu:
- To install the Timeshift backup utility on your Ubuntu 20.04 System. To do this execute the following command:

```
$ sudo apt install timeshift
```

1. Open the timeshift application via top left Activities menu.
2. Select backup destination. timeshift will search your system for available file-system partition and provide you with an option to where to create backup file.
3. Select how often you wish to perform the system backup and how many backup snapshots you wish to retrain before the first backup is overwritten.
4. As the screenshot indicates the home directories are excluded by default. Depending on you work environment select whether you wish to include home directories into the backup.
5. This will conclude your initial backup schedule setup. Hit the Finish button.
6. The backup has not been created yet. You can either wait until the timeshift automatically triggers the backup or simply hit the Create button to perform the previously predefined backup now.
7. Wait for the backup to complete.
8. If all went well you should now see your first backup snapshot listed.

Restore from backup:

1. We will restore the system from the previously created system backup snapshot. Select a backup snapshot from which you wish to restore and click on the Restore button. Timeshift give you an option on how to restore from your backup. Unless you know what you are doing simply hit the Next button to go with the default.

2. The Timeshift will provide you with a list of changes it will take to restore from the backup just to make sure no data is lost in the process.
3. Once you hit the Next button the system will be restored and restarted. All done.

5.2.11 Introduction to Graphical Environment (GNOME)

- **Gnome** is a user-friendly **graphical desktop environment** for UNIX and UNIX-like systems that enables users to easily use and configure their computers.
- Linux is often seen as a command-only operating system. This is far from true: although its command-line is a powerful interface, you can also launch graphical environments on Linux.
- Graphical environments are the de facto standard for working with a workstation. Many users know the Microsoft Windows family or the Apple MacOS series. However, those two aren't the only providers of a graphical environment. When the Intel-compliant PCs were hardly known to the world, consoles and other personal computers already provided a graphical environment to their users.
- GNOME, KDE, XFCE4 are popular desktop graphical environments; enlightenment, fluxbox, window maker, icewm, are window managers.

5.2.11.1 Ubuntu Utilities

1. **VirtualBox:**

- VirtualBox Guest Additions are a collection of device drivers and system applications designed to achieve closer integration between the host and guest operating systems. They help to enhance the overall interactive performance and usability of guest systems.
- VirtualBox Guest Additions offer the following features:
  - Easy mouse pointer integration.
  - Easy way to share folders between the host and the guest.
  - Drag and drop feature allows copying or opening files, copy clipboard formats from the host to the guest or from the guest to the host.
  - Share clipboard (for copy and paste) of the guest operating system with your host operating system.
  - Better video support provides accelerated video performance.
  - Better Time synchronization between guest and host.
  - Standard host/guest communication channels.
  - Seamless Windows features allow you to run windows of your guest operating system seamlessly next to the windows of your host.
  - The VirtualBox Guest Additions should be installed inside a virtual machine after the guest operating system has been installed.

2. **Evolution:**

- Evolution has been in the market for decades and in spite of its all Ups and Downs, as its name says, Evolution has really evolved a lot over the period of time and stands better than ever before.
- Evolution is a personal information manager which was developed for Gnome Environment. It comes with a number of features such as E-mail, Calendar, Address book, and more.
- Key Features of Evolution Linux:
  - It supports POP and IMAP Protocols for Email Retrievals and uses SMTP for Email transmission.
  - Auto Spam filtering option is there in the Evolution.
  - Manage your Contacts with the help of the Address Book.
  - Calendar Support is also there.
  - A number of Plugins are available with which you can further increase its functionality.
  - Connection to Microsoft Exchange server is possible with the help of suitable Plugins.

3. **GIMP:**

- GIMP is a cross-platform image editor available for GNU/Linux, OS X, Windows and more operating systems. It is free software; you can change its source code and distribute your changes,
- Whether you are a graphic designer, photographer, illustrator, or scientist, GIMP provides you with sophisticated tools to get your job done. You can further enhance your productivity with GIMP thanks to many customization options and 3rd party plugins.
- GIMP 2.10 is the result of six years of work that originally focused on porting the program to a new image processing engine, GEGL. However the new version ships with far more new features, including new and improved tools, better file formats support, various usability improvements, revamped color management support, a plethora of improvements targeted at digital painters and photographers, metadata editing, and much, much more.
- **Advantages:**
  - **High bit depth support** allows processing images with up to 32-bit per color channel precision and open/export PSD, TIFF, PNG, EXR, and RGBE files in their native fidelity. Additionally, FITS images can be opened with up to 64-bit per channel precision.

  - **Multi-threading** allows making use of multiple cores for processing. Not all features in GIMP make use of that, it's something we intend to work on further. A point of interest is that multi-threading happens through GEGL processing, but also in core GIMP itself, for instance to separate painting from display code.
  - **GPU-side processing** is still optional, but available for systems with stable OpenCL drivers.
  - You can find configuration options for multi-threading and hardware acceleration in *Edit > Preferences > System Resources*.

4. **Bleachbit:**

- BleachBit has many useful features designed to help you easily clean your computer to free space and maintain privacy.
- Delete your private files so completely that "even God can't read them" according to South Carolina Representative Trey Gowdy.
- Simple operation: read the descriptions, check the boxes you want, click preview, and click delete.
- Multi-platform: Linux and Windows.
- Free of charge and no money trail.
- Free to share, learn, and modify (open source)
- No adware, spyware, malware, backdoors, browser toolbars, or "value-added software".
- Translated to 64 languages besides American English.
- Shred files to hide their contents and prevent data recovery.
- Shred any file (such as a spreadsheet on your desktop).
- Overwrite free disk space to hide previously deleted files.
- Portable app for Windows: run without installation.
- Command line interface for scripting and automation.
- CleanerML write your own cleaners using XML.
- Automatically import and update winapp2.ini cleaner files (a separate download) giving Windows users access to 2500+ additional cleaners.
- Frequent software updates with new features.

Uses of BleachBit:

- Free disk space.
- Reduce the size of backups and the time to create them by removing unnecessary files.
- Maintain privacy.
- Improve system performance (by vacuuming the Firefox database, for example).
- Prepare whole disk images for compression (common for "ghost" backups and virtual machines) by wiping free disk space.

Cleaners:

- BleachBit includes a growing list of cleaners. Typically each cleaner represents an application such as Firefox or Internet Explorer. Within each cleaner, BleachBit gives options covering components that can be cleaned such as cache, cookies, and log files. Each option is given a description to help you make good decisions.

5. **Unity Tweak Tool:**

- To install the **Unity Tweak Tool** open the Ubuntu Software Center by clicking the suitcase icon on the launcher, and search for **Unity Tweak**. Click the Install button in the top right corner and enter your password when it is requested. To open the **Tweak Tool** open the Dash and search for **Tweak**.
- Unity is a now-obsolete desktop environment for Ubuntu that served as the default DE for Ubuntu for several years. It was retired in 2016 in favor of GNOME 3. Because Unity originally supported low-power netbooks, it was by design not open to significant customization. However, the Unity Tweak Tool supports a variety of special configurations that allow for some non-stock edits to Unity's look-and-feel.
- The Tweak Tool has a series of icons split into the following categories:
  - Unity
  - Window Manager
  - Appearance
  - System
- The Unity category tweaks the launcher, the search tool, the top panel, the switcher, web applications, and a few miscellaneous items to do with Unity.
- The Window Manager category governs the general Window Manager, Workspace Settings, Window Spread, Window Snapping,Hot Corners, and other miscellaneous Window Manager items.
- The Appearance category modifies the theme, icons, cursors, fonts, and window controls.
- The System category tweaks the desktop icons, security, and scrolling.

5.2.11.2 SAMBA Overview

- Samba is a suite of Unix applications that speak the Server Message Block (SMB) protocol. Microsoft Windows operating systems and the OS/2 operating system use SMB to perform client-server networking for file and printer sharing and associated operations. By supporting this protocol, Samba enables computers running Unix to get in on the action, communicating with the same networking protocol as Microsoft Windows and appearing as another Windows system on the network from the perspective of a Windows client.
- A Samba server offers the following services:
  - Share one or more directory trees.
  - Share one or more Distributed file system (Dfs) trees.

  - Share printers installed on the server among Windows clients on the network.
  - Assist clients with network browsing.
  - Authenticate clients logging onto a Windows domain.
  - Provide or assist with Windows Internet Name Service (WINS) name-server resolution.
- The Samba suite also includes client tools that allow users on a Unix system to access folders and printers that Windows systems and Samba servers offer on the network.
- Samba is an extremely useful networking tool for anyone who has both Windows and Unix systems on his network. Running on a Unix system, it allows Windows to share files and printers on the Unix host, and it also allows Unix users to access resources shared by Windows systems.
- Although it might seem natural to use a Windows server to serve files and printers to a network containing Windows clients, there are good reasons for preferring a Samba server for this duty. Samba is reliable software that runs on reliable Unix operating systems, resulting in fewer problems and a low cost of maintenance. Samba also offers better performance under heavy loads, outperforming Windows 2000 Server by a factor of 2 to 1 on identical PC hardware, according to published third-party benchmarks. When common, inexpensive PC hardware fails to meet the demands of a huge client load, the Samba server can easily be moved to a proprietary "big iron" Unix mainframe, which can outperform Windows running on a PC many times. If all that weren't enough, Samba has a very nice cost advantage: it's free. Not only is the software itself freely available, but also no client licenses are required, and it runs on high-quality, free operating systems such as Linux and FreeBSD.
- Samba is commonly used by large organizations with thousands of users on their networksBut Samba's user base includes organizations all over the planet, of all types and sizes: from international corporations, to medium and small businesses, to individuals who run Samba on their Linux laptops. In the last case, a tool such as VMware is used to run Windows on the same computer, with Samba enabling the two operating systems to share files.
- The types of users vary even more—Samba is used by corporations, banks and other financial institutions, government and military organizations, schools, public libraries, art galleries, families, and even authors! This book was developed on a Linux system running VMware and Windows 2000, with Adobe FrameMaker running on Windows and the document files served by Samba from the Linux filesystem.

Summary

- Windows is a series of operating systems developed by Microsoft. Each version of Windows includes a graphical user interface, with a desktop that allows users to view files and folders in windows. For the past two decades, Windows has been the most widely used operating system for personal computers PCs.
- Windows Setup is an installer that prepares a hard disk drive for a Microsoft Windows operating system installation by executing two processes: (a) initializing the drive and (b) copying system files to that drive in order for the operating system to be run local
- Process management is an approach to organizational design that implies that activities performed within the company are organized and optimized in processes. It supports all model types, such as the business model or the organizational chart that you need to ensure coherent and transparent process documentation.
- The Control Panel is a component of Microsoft Windows that provides the ability to view and change system settings. It consists of a set of applets that include adding or removing hardware and software, controlling user accounts, changing accessibility options, and accessing networking settings.
- The kernel is the collection of these pieces that are considered inseparable hence the name and which run independently of any user/process. It acts as a bridge between applications and the data processing performed at the hardware level. The kernel is the central module of an operating system (OS).
- Service management and assurance requires the unification of different systems into a seamless network, ensuring consistency and that CSPs fully benefit from their deployed assets, leveraging the information they generate to ensure full optimization and quality.
- Ubuntu is a Linux-based operating system. It is designed for computers, smartphones, and network servers.
- Any running program or a command given to a Linux system is called a process.
- A process could run in foreground or background.
- The priority index of a process is called Nice in Linux. Its default value is 0, and it can vary between 20 to -19.
- The lower the Niceness index, the higher would be priority given to that task.

Check Your Understanding

1. Two ways to arrange multiple windows on the desktop are _____.
   (a) drag and drop (b) point and click
   (c) cascade and tile (d) minimize and maximize
2. When a command is not available for selection until certain other conditions are met, it is called _____.
   (a) dimmed command (b) unavailable command
   (c) dialog box (d) cascade

3. The systems which allow only one process execution at a time, are called _____.
   (a) uniprogramming systems (b) uniprocessing systems
   (c) unitasking systems (d) none of the mentioned
4. In operating system, each process has its own _____.
   (a) address space and global variables
   (b) open files
   (c) pending alarms, signals and signal handlers
   (d) all of the mentioned
5. What is the ready state of a process?
   (a) when process is scheduled to run after some execution.
   (b) when process is unable to run until some task has been completed.
   (c) when process is using the CPU.
   (d) none of the mentioned.
6. In a pure kernel level thread facility all of the work of thread management is done by the _____.
   (a) Application (b) Program
   (c) Kernel (d) Threads
7. Kernal mode of operating system runs when the mode bit is ____.
   (a) 1 (b) 0
   (c) x (d) undefined
8. Which of the following does authentication aim to accomplish?
   (a) Restrict what operations/data the user can access.
   (b) Determine if the user is an attacker.
   (c) Flag the user if he/she misbehaves.
   (d) Determine who the user is.
   (e) None of the above.
9. Which of the following is not a recovery technique?
   (a) Deferred update (b) Immediate update
   (c) Two-phase commit (d) Recovery management
10. ____ deals with soft errors, such as power failures.
    (a) system recovery (b) media recovery
    (c) database recovery (d) failure recovery

Answer Key

1. (c)	2. (a)	3. (b)	4. (d)	5. (a)	6. (c)	7.(b)	8. (b)	9. (c)	10. (d)

Practice Questions

Q.I Answer the following Questions in short.

1. Explain the main purpose of an operating system?
2. What is kernel?
3. What are real-time systems?
4. What is a virtual memory?

5. What is an Operating System?
6. Define a process.
7. Define interrupt.

Q.II Answer the following Questions:

1. What is Multiprogramming?
2. Which command is used to enter the file system image? Explain it.
3. Explain the process of installation software.
4. Describe the objective of multiprogramming.
5. List out features of Process Management.
6. What is process management?
7. Explain the process of Backup and recovery of Ubuntu.

Q.III Write short note on.

1. Process Management
2. Basic Troubleshooting
3. Kernal
4. Installation of Ubuntu
5. Backup and Recovery

■■■

6... Linux Shell Scripting

Objectives ...

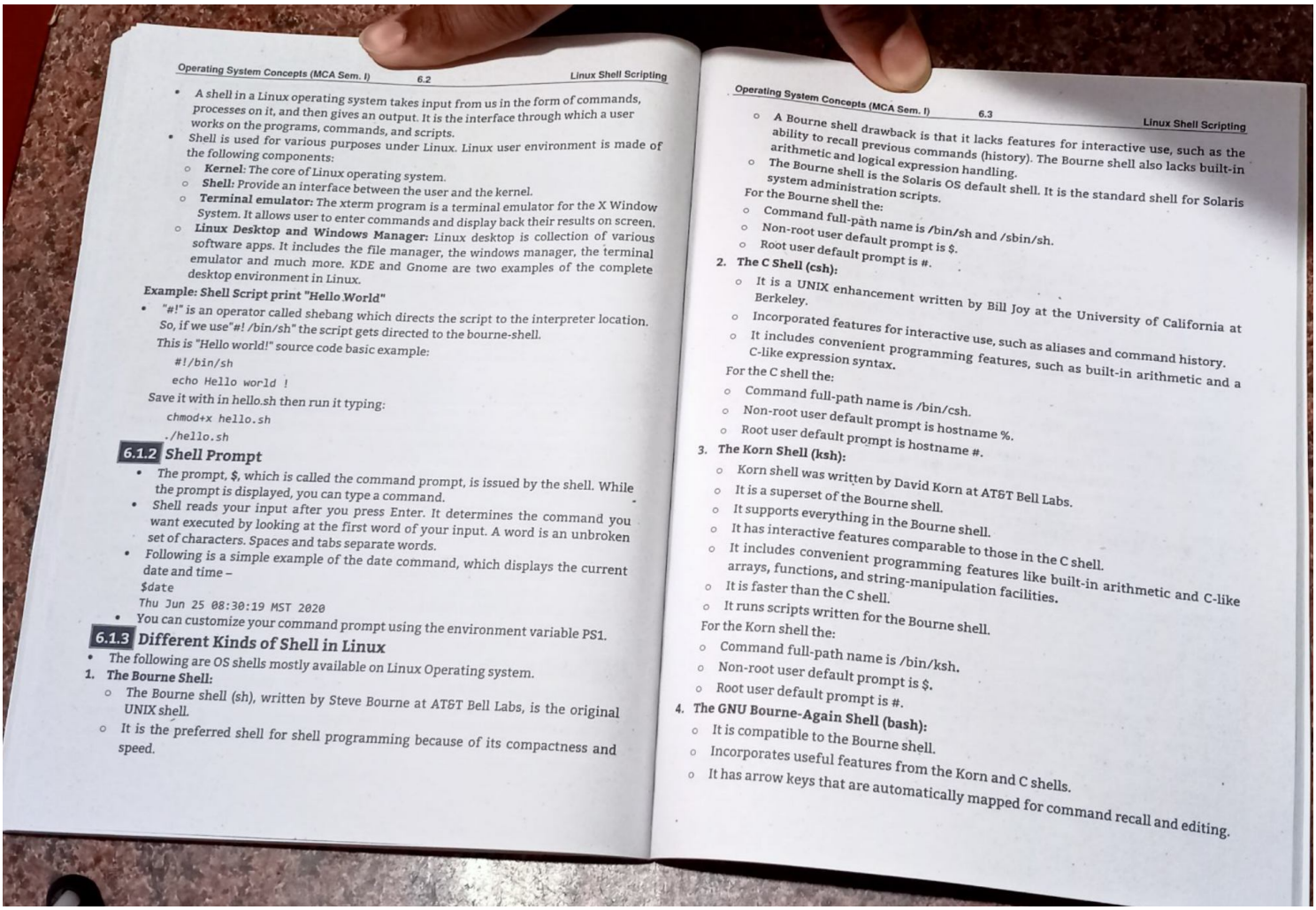
- To study Linux Shell Scripting in detailed.
- To learn variable, flow controls, loops and functions in Linux shell scripting.
- To study list in Linux shell scripting.
- To understand reading and writing files.
- To understand Real time scripts for different system administration activities

6.1 INTRODUCTION

- A shell is special user program which provides an interface to user to use operating system services. Shell accepts human readable commands from user and convert them into something which kernel can understand.
- Shell is a command language interpreter that execute commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or start the terminal.
- Shell is a unique and multifaceted program. It is also a process that creates an environment for you to work. It allows users to execute commands by typing them manually, or automatically in program called shell script.
- Shell is a programming language that constructs such as conditional execution, loops, variables, functions and many more.
- There can be Command line shells and Graphical shells. Command line Shell can be accessed by user using a command line interface. A special program called **Terminal** in Linux/MacOS or **Command Prompt** in Windows OS. It is provided to type in the human readable commands such as "cat", "ls" etc.

6.1.1 Shell Scripts

- A shell script is a computer program, designed to be run by the Unix shell command line interpreter.
- The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text.



- A shell in a Linux operating system takes input from us in the form of commands, processes on it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts.
- Shell is used for various purposes under Linux. Linux user environment is made of the following components:
  - **Kernel:** The core of Linux operating system.
  - **Shell:** Provide an interface between the user and the kernel.
  - **Terminal emulator:** The xterm program is a terminal emulator for the X Window System. It allows user to enter commands and display back their results on screen.
  - **Linux Desktop and Windows Manager:** Linux desktop is collection of various software apps. It includes the file manager, the windows manager, the terminal emulator and much more. KDE and Gnome are two examples of the complete desktop environment in Linux.

Example: Shell Script print "Hello World"

- "#!" is an operator called shebang which directs the script to the interpreter location. So, if we use"#! /bin/sh" the script gets directed to the bourne-shell.

This is "Hello world!" source code basic example:

```
#!/bin/sh
echo Hello world !
```

Save it with in hello.sh then run it typing:

```
chmod+x hello.sh
./hello.sh
```

6.1.2 Shell Prompt

- The prompt, **$**, which is called the command prompt, is issued by the shell. While the prompt is displayed, you can type a command.
- Shell reads your input after you press Enter. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.
- Following is a simple example of the date command, which displays the current date and time –

```
$date
Thu Jun 25 08:30:19 MST 2020
```

- You can customize your command prompt using the environment variable PS1.

6.1.3 Different Kinds of Shell in Linux

- The following are OS shells mostly available on Linux Operating system.

1. **The Bourne Shell:**
   - The Bourne shell (sh), written by Steve Bourne at AT&T Bell Labs, is the original UNIX shell.
   - It is the preferred shell for shell programming because of its compactness and speed.

   - A Bourne shell drawback is that it lacks features for interactive use, such as the ability to recall previous commands (history). The Bourne shell also lacks built-in arithmetic and logical expression handling.
   - The Bourne shell is the Solaris OS default shell. It is the standard shell for Solaris system administration scripts.

   For the Bourne shell the:
   - Command full-path name is /bin/sh and /sbin/sh.
   - Non-root user default prompt is $.
   - Root user default prompt is #.

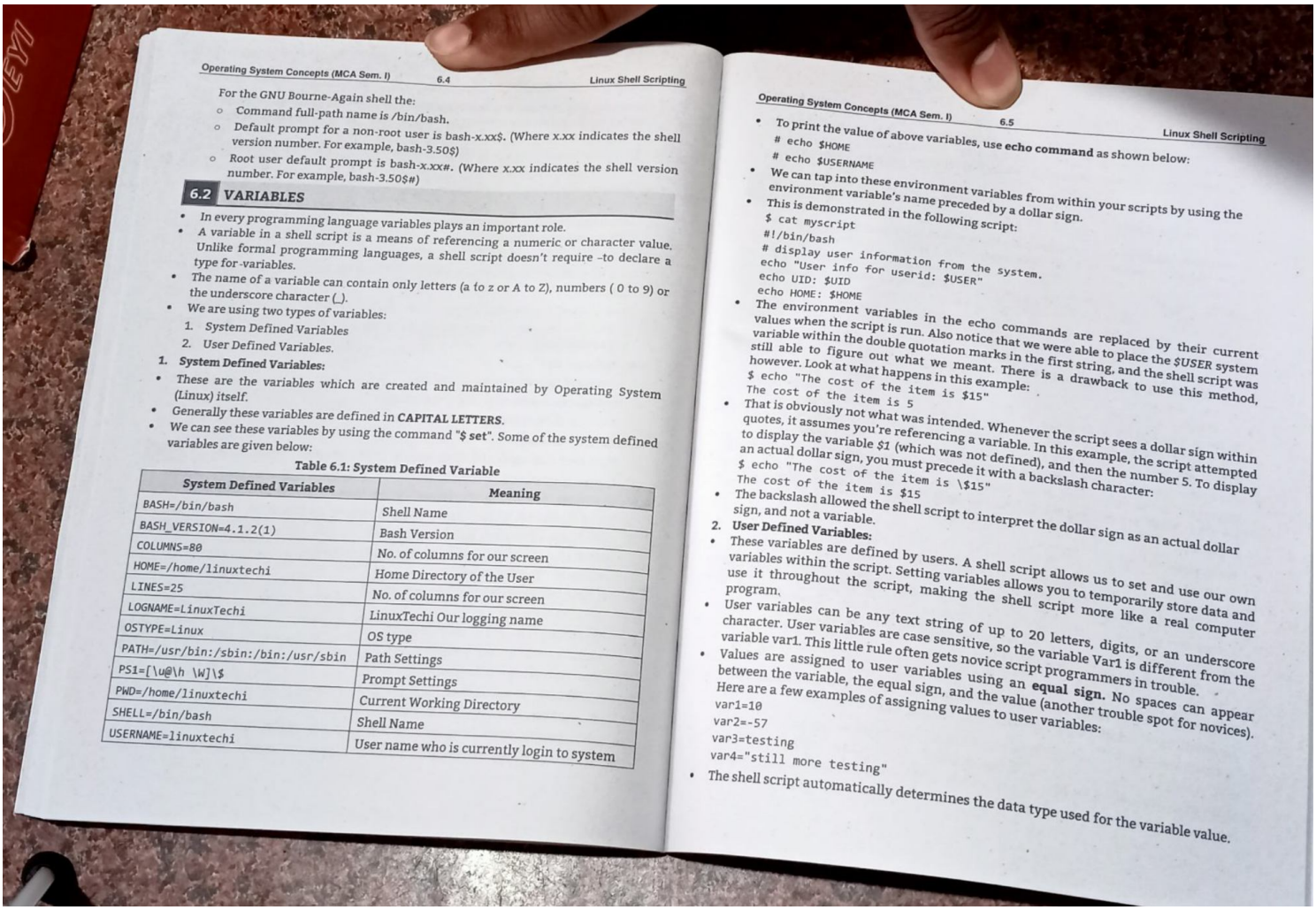
2. **The C Shell (csh):**
   - It is a UNIX enhancement written by Bill Joy at the University of California at Berkeley.
   - Incorporated features for interactive use, such as aliases and command history.
   - It includes convenient programming features, such as built-in arithmetic and a C-like expression syntax.

   For the C shell the:
   - Command full-path name is /bin/csh.
   - Non-root user default prompt is hostname %.
   - Root user default prompt is hostname #.

3. **The Korn Shell (ksh):**
   - Korn shell was written by David Korn at AT&T Bell Labs.
   - It is a superset of the Bourne shell.
   - It supports everything in the Bourne shell.
   - It has interactive features comparable to those in the C shell.
   - It includes convenient programming features like built-in arithmetic and C-like arrays, functions, and string-manipulation facilities.
   - It is faster than the C shell.
   - It runs scripts written for the Bourne shell.

   For the Korn shell the:
   - Command full-path name is /bin/ksh.
   - Non-root user default prompt is $.
   - Root user default prompt is #.

4. **The GNU Bourne-Again Shell (bash):**
   - It is compatible to the Bourne shell.
   - Incorporates useful features from the Korn and C shells.
   - It has arrow keys that are automatically mapped for command recall and editing.



For the GNU Bourne-Again shell the:

- Command full-path name is /bin/bash.
- Default prompt for a non-root user is bash-x.xx$. (Where x.xx indicates the shell version number. For example, bash-3.50$)
- Root user default prompt is bash-x.xx#. (Where x.xx indicates the shell version number. For example, bash-3.50$#)

6.2 VARIABLES

- In every programming language variables plays an important role.
- A variable in a shell script is a means of referencing a numeric or character value. Unlike formal programming languages, a shell script doesn't require –to declare a type for-variables.
- The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_).
- We are using two types of variables:
  1. System Defined Variables
  2. User Defined Variables.

1. System Defined Variables:

- These are the variables which are created and maintained by Operating System (Linux) itself.
- Generally these variables are defined in **CAPITAL LETTERS**.
- We can see these variables by using the command "**$ set**". Some of the system defined variables are given below:

Table 6.1: System Defined Variable

System Defined Variables	Meaning
BASH=/bin/bash	Shell Name
BASH_VERSION=4.1.2(1)	Bash Version
COLUMNS=80	No. of columns for our screen
HOME=/home/linuxtechi	Home Directory of the User
LINES=25	No. of columns for our screen
LOGNAME=LinuxTechi	LinuxTechi Our logging name
OSTYPE=Linux	OS type
PATH=/usr/bin:/sbin:/bin:/usr/sbin	Path Settings
PS1=[\u@\h \W]\$	Prompt Settings
PWD=/home/linuxtechi	Current Working Directory
SHELL=/bin/bash	Shell Name
USERNAME=linuxtechi	User name who is currently login to system

- To print the value of above variables, use **echo command** as shown below:

```
# echo $HOME
# echo $USERNAME
```

- We can tap into these environment variables from within your scripts by using the environment variable's name preceded by a dollar sign.
- This is demonstrated in the following script:

```
$ cat myscript
#!/bin/bash
# display user information from the system.
echo "User info for userid: $USER"
echo UID: $UID
echo HOME: $HOME
```

- The environment variables in the echo commands are replaced by their current values when the script is run. Also notice that we were able to place the *$USER* system variable within the double quotation marks in the first string, and the shell script was still able to figure out what we meant. There is a drawback to use this method, however. Look at what happens in this example:

```
$ echo "The cost of the item is $15"
The cost of the item is 5
```

- That is obviously not what was intended. Whenever the script sees a dollar sign within quotes, it assumes you're referencing a variable. In this example, the script attempted to display the variable *$1* (which was not defined), and then the number 5. To display an actual dollar sign, you must precede it with a backslash character:

```
$ echo "The cost of the item is \$15"
The cost of the item is $15
```

- The backslash allowed the shell script to interpret the dollar sign as an actual dollar sign, and not a variable.

2. User Defined Variables:

- These variables are defined by users. A shell script allows us to set and use our own variables within the script. Setting variables allows you to temporarily store data and use it throughout the script, making the shell script more like a real computer program.
- User variables can be any text string of up to 20 letters, digits, or an underscore character. User variables are case sensitive, so the variable Var1 is different from the variable var1. This little rule often gets novice script programmers in trouble.
- Values are assigned to user variables using an **equal sign.** No spaces can appear between the variable, the equal sign, and the value (another trouble spot for novices). Here are a few examples of assigning values to user variables:

```
var1=10
var2=-57
var3=testing
var4="still more testing"
```

- The shell script automatically determines the data type used for the variable value.

- Variables defined within the shell script maintain their values throughout the life of the shell script but are deleted when the shell script completes.
- Just like system variables, user variables can be referenced using the dollar sign:

```
$ cat test3
#!/bin/bash
# testing variables
days=2
guest="Pushkar"
echo "$guest checked in $days days ago"
days=5
guest="Aditya"
echo "$guest checked in $days days ago"
days=3
guest="Achal"
echo "$guest checked in $days days ago"
$
```

Running the script produces the following output:

```
$ chmod u+x test3
$ ./test3
Pushkar checked in 2 days ago
Aditya checked in 5 days ago
Achal checked in 3 days ago
```

- Each time the variable is referenced, it produces the value currently assigned to it. It's important to remember that when referencing a variable value you use the **dollar sign**, but when referencing the variable to assign a value to it, you do not use the dollar sign.
- For example:

```
$ cat test4
#!/bin/bash
# assigning a variable value to another variable
value1=10
value2=$value1
echo "The resulting value is $value2"
$
```

- When you use the **value** of the **value1** variable in the assignment statement, you must still use the dollar sign. This code produces the following output:

```
$ chmod u+x test4
$ ./test4
The resulting value is 10
$
```


- If you forget the dollar sign, and make the value2 assignment line look like:

```
value2=value1
```

you get the following output:

```
$ ./test4
The resulting value is value1
$
```

- Without the dollar sign the shell interprets the variable name as a normal text string, which is most likely not what you wanted.

Use of Backtick symbol (`) in shell variables:

- The **backtick** allows you to assign the output of a shell command to a variable. While this doesn't seem like much, it is a major building block in script programming. You must surround the entire command line command with backtick characters:

```
testing=`date`
```

- The shell runs the command within the **backticks** and assigns the output to the variable testing. Here's an example of creating a variable using the output from a normal shell command:

```
$ cat test5
#!/bin/bash
# using the backtick character
testing=`date`
echo "The date and time are:" $testing
$
```

- The variable testing receives the output from the date command, and it is used in the echo statement to display it. Running the shell script produces the following output:

```
$ chmod u+x test5
$ ./test5
The date and time are: Fri Aug 21 20:23:25 EDT 2020
```

6.3 FLOW CONTROLS

- The shell provides several commands that we can use to control the flow of execution in our program. we will look at the following:
  - if
  - if...else...fi
  - if...elif...fi
  - test

1. if command:

- if command is used to execute command based on condition.
- if statement allow us to make decisions in our shell scripts.
- A basic if statement effectively says, if a particular condition is true, then perform a given set of actions. If it is not true then don't perform those actions.

- **Syntax:**

```
if [ expression ]
then
   Statement(s) to be executed if expression is true
fi
```

Example 6.1: Shell script for *if* command.

```
#!/bin/sh
a=100
b=200
if [ $a == $b ]
then
   echo "a is equal to b"
fi
if [ $a != $b ]
then
   echo "a is not equal to b"
fi
```

Output:

```
a is not equal to b
```

2. if...else...fi command:

- The if...else...fi statement is the next form of control statement that allows shell to execute statements in a controlled way and make the right choice.
- **Syntax:**

```
if [ expression ]
then
   Statement(s) to be executed if expression is true
else
   Statement(s) to be executed if expression is not true
fi
```

- The Shell expression is evaluated in the above syntax. If the resulting value is true, given statement(s) are executed.

Example 6.2: Shell script for *if...else...fi* command.

```
#!/bin/sh
a=100
b=200
if [ $a == $b ]
then
   echo "a is equal to b"
else
   echo "a is not equal to b"
fi
```

Output:

```
a is not equal to b
```


3. if...elif...fi command:

- You can nest a new if inside an else with elif.
- **Syntax:**

```
if [ expression 1 ]
then
   Statement(s) to be executed if expression 1 is true
elif [ expression 2 ]
then
   Statement(s) to be executed if expression 2 is true
else
   Statement(s) to be executed if no expression is true
fi
```

Example 6.3: Shell script for *if...elif...fi* command.

```
#!/bin/sh
a=100
b=200
if [ $a == $b ]
then
   echo "a is equal to b"
elif [ $a -gt $b ]
then
   echo "a is greater than b"
elif [ $a -lt $b ]
then
   echo "a is less than b"
else
   echo "None of the condition met"
fi
```

Output:

```
a is less than b
```

4. test command[ ]:

- test command or [expr] is used to see if an expression is true, and if it is true it return zero(0), otherwise returns nonzero for false.
- The test command is used to check file types and compare values. Test is used in conditional execution. It is used for:
  - File attributes comparisons.
  - Perform string comparisons.
  - Basic arithmetic comparisons.
- **Syntax:** `test expression OR [ expression ]`

Example 6.4: Shell script for *test* command.

```
test 5 -gt 2 && echo "Yes"
test 1 -lt 2 && echo "Yes"
```

Output:

```
Yes
Yes
```

- You can use '**-ge**' to test greater than or equal to, '**-gt**' for greater than, '**-le**' for less than or equal to, '**-lt**' for less than, and '**-ne**' for not equal.

6.4 LOOPS

Loops in Shell Scripts:

- Computer can repeat particular instruction again and again, until particular condition satisfies. A group of instruction that is executed repeatedly is called a loop.

1. **for loop:**

- The for loop operate on lists of items. It repeats a set of commands for every item in a list.
- Check following syntax, var is the name of a variable and word1 to wordN are sequences of characters separated by spaces (words). Each time the for loop executes, the value of the variable var is set to the next word in the list of words, word1 to wordN.

Syntax:

```
for var in word1 word2 ...wordN
do
   Statement to be executed
done
```

Example 6.5: Shell script for *for* loop.

```
#!/bin/sh
for var in 10 9 8 7 6 5 4 3 2 1
do
   echo $var
done
```

Output:

```
10
9
8
7
6
5
4
3
2
1
```


2. **while loop:**

- The shell script while is a control loop statement that allows code or commands to be executed repeatedly based on a given condition.
- **Syntax:**

```
while command
do
   Statement to be executed
done
```

Example 6.6: Shell script for *while* loop.

```
#!/bin/sh
a=0
while [ $a -lt 5]
do
   echo $a
   a=`expr $a + 1`
done
```

Output:

```
0
1
2
3
4
```

3. **until loop:**

- It is similar to while loop.
- The only difference is that, until statement executes its code block while its conditional expression is false, and while statement executes its code block while its conditional expression is true.
- **Syntax:**

```
until command
do
   Statement to be executed until command is true
Done
```

Example 6.7: Shell script for *until* loop.

```
#!/bin/sh
a=5
until [ ! $a -gt 0 ]
do
   echo $a
   a=`expr $a - 1`
done
```

Output:

```
5
4
3
2
1
```

6.5 FUNCTIONS

- *Shell functions are a way to group commands for later execution, using a single name for this group, or routine.*
- *The name of the routine must be unique within the shell or script.*
- *All the commands that make up a function are executed like regular commands. When calling on a function as a simple command name, the list of commands associated with that function name is executed.*
- *A function is executed within the shell in which it has been declared: no new process is created to interpret the commands.*
- *In real-world scripts, we break down big tasks or scripts into smaller logical tasks. This modularization of scripts helps in the better development and understanding of code. The smaller logical blocks of script are called Functions.*
- *A function is a group of commands that are assigned a name that acts like a handle to that group of commands. To execute this group of commands defined in the function, you simply call the function by the name you provided.*
- *There will be cases where you need to execute a block of code that achieves a specific procedure several times in different places in your shell script. Shell functions are like subroutines, procedures, and functions in other programming languages.*

Advantages of functions:

- *Using functions, we can easily understand complex script through logical blocks or functions.*
- *When a big and complex script is divided into functions, then it becomes easy to develop and test the script.*
- *If a certain part of code is repeated again and again in the big script, then using functions to replace repetitive code is very practical, such as checking whether the file or directory is present or not.*
- *We define functions for specific tasks or activities. Such functions can be called as commands in scripts. Functions can be defined on a command line or inside scripts.*

Syntax of Function:

- *The syntax of the function declaration inside the shell script is as follows:*

```
function_name()
{
   block of code
}
```


An alternate function syntax is mentioned here:

```
function function_name
{
   block of code
}
```

- Functions should be defined at the beginning of a script.
- In single-line functions, every command should end with a semicolon.
- Let's write a very simple function to illustrate the preceding syntax:

```
$ hello() {echo 'Hello world!';}
$ hello
```

Output:

```
Hello world!
```

Example 6.8: The function to convert lowercase letters into uppercase letters.

```
#!/bin/bash
function Convert_Upper()
{
   echo $1 | tr 'abcdefghijklmnopqrstuvwxyz' \ 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
}
Convert_Upper "dr sunita patil - embedded android and linux training"
```

Output:

```
DR SUNITA PATIL - EMBEDDED ANDROID AND LINUX TRAINING
```

Different types of functions:

1. Functions that return a value to the calling section of the script using 'return' keyword. You can make certain calculations and return the result.
2. Functions that terminates the shell script using the 'exit' keyword. You can use the exit keyword to terminate the script based on certain conditions.
3. Functions that change the value of a variable/variables. You can assign/change the value of a variable declared outside the function block. An important thing to note here is that if you declare a variable outside a function, it will behave like a global variable and can be accessed inside a function.
4. Functions that echo output to the standard output. You can print anything to stdout inside the function.

Example 6.9: A shell function can receive parameters.

```
patil@solexp11$ cat addfunc.ksh
#!/bin/ksh
function plus
{
   let result="$1 + $2"
   echo $1 + $2 = $result
}
plus 3 10
plus 20 13
plus 20 22
```

Output:

```
patil@solexp11$ ./addfunc.ksh
3 + 10 = 13
20 + 13 = 33
20 + 22 = 42
```

Example 6.10: To calculate and return the average of numbers passed as argument.

```
find_average()
{
    sum=0
    i=1
    len=$#
    echo $len
    x=$((len + 1))
    echo "no of element in array $x"
    while [ $i -lt $x ]
    do
    arg=${!i}
    echo "No is $arg"
    sum=$((sum + arg))
    i=$((i + 1 ))
    done
    avg=$((sum / len))
    return $avg
}
find_average 10 20 30 40
echo "Average is $? "
read
```

Output:

```
4
no of element in array 5
No is 10
No is 20
No is 30
No is 40
Average is 25
```


Example 6.11: To terminate the script when the value of a certain variable becomes negative.

```
check_negative()
{
    a=5
    while:
    do
    if [ $a -lt 0 ]; then
    echo "terminating the script"
    exit 5
    fi
    a=$((a - 1))
    done
}
check_negative
```

Output:

```
~$ sh script.sh
~$ echo $?
5
```

Example 6.12: To write a program of Fibonacci Series.

```
fibonacci()
{
   echo $1
   n=$1
   a=0
   b=1
   i=2
   echo "Fibonacci Series is"
   for(( i=0; i<$n; i++ ))
   do
     echo -n "$a"
     fn=$(( a + b ))
     a=$b
     b=$fn
   done
}
echo "entet the no"
read no
fibonacci $no
read
```

Output:

```
enter the no
10
10
Fibonacci Series is
0112358132134
```

Example 6.13: Write a script for reverse order.

```
echo "Enter a Number:"
read a
rev=0
sd=0
or=$a
while [ $a -gt 0 ]
do
    sd=`expr $a % 10`
    temp=`expr $rev \* 10`
    rev=`expr $temp + $sd`
    a=`expr $a / 10`
done
echo "Reverse of $or is $rev"
read
```

Output:

```
Enter a Number:
123456
Reverse of 123456 is 654321
```

6.6 LIST

1. **# ls**

- **ls** command is one of the most frequently used command in Linux.
- **ls** command is the command you may use when you get into the command prompt of Linux Box.
- **ls** command daily basis and frequently even though we may not aware and never use all the **ls** option available.
- List Files using **ls** with no option. **ls** with no option list files and directories in bare format where we won't be able to view details like file types, size, modified date and time, permission and links etc.

2. **List Files With option –l**

- Here, **ls -l** (-l is character not one) shows file or directory, size, modified date and time, file or folder name and owner of file and its permission.

 # ls -l

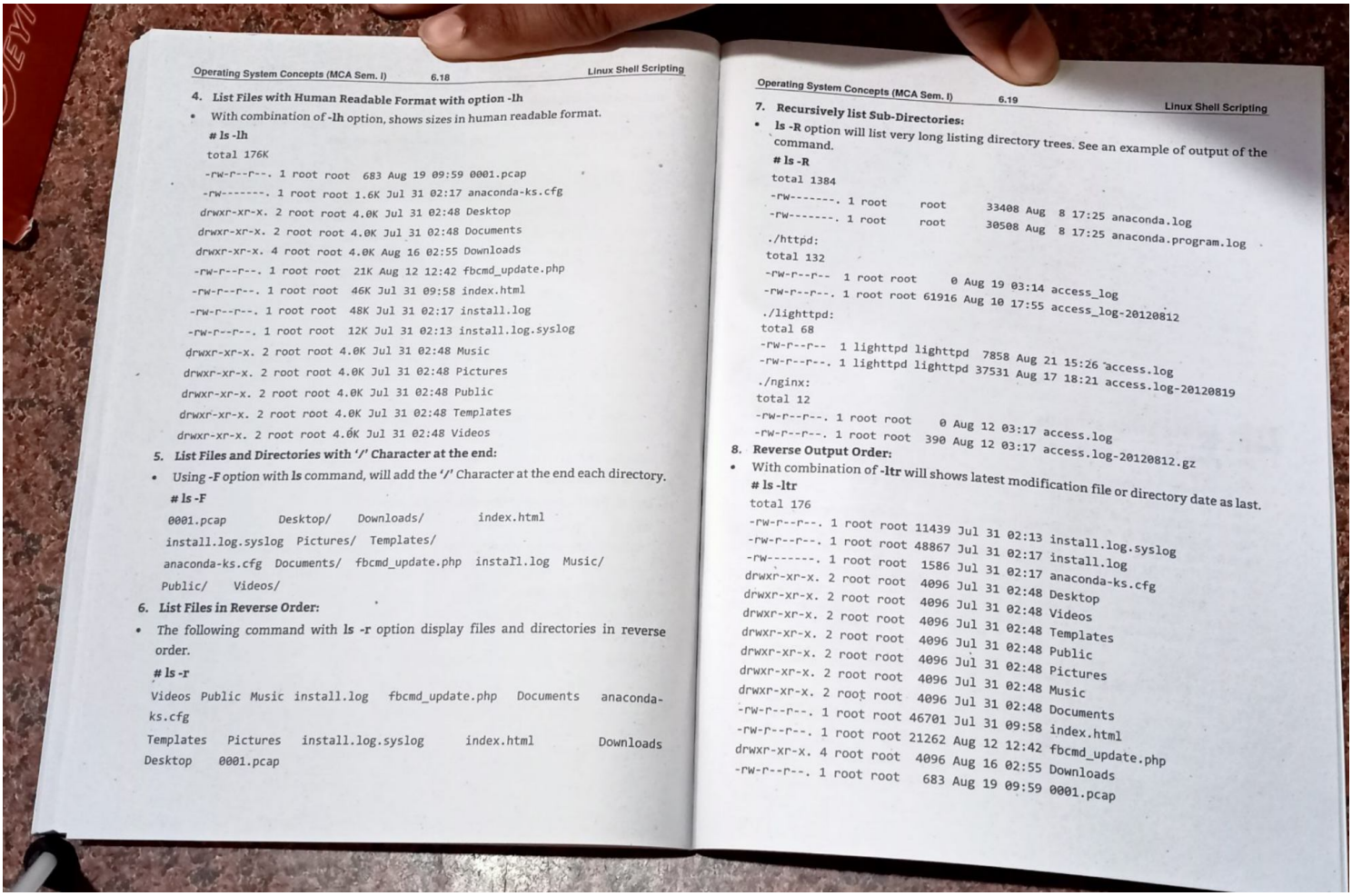
```
total 176
-rw-r--r--. 1 root root   683 Aug 19 09:59 0001.pcap
-rw-------. 1 root root  1586 Aug 31 02:17 anaconda-ks.cfg
drwxr-xr-x. 2 root root  4096 Aug 31 02:48 Desktop
drwxr-xr-x. 2 root root  4096 Aug 31 02:48 Documents
drwxr-xr-x. 4 root root  4096 Aug 16 02:55 Downloads
-rw-r--r--. 1 root root 21262 Aug 12 12:42 fbcmd_update.php
-rw-r--r--. 1 root root 46701 Aug 31 09:58 index.html
-rw-r--r--. 1 root root 48867 Aug 31 02:17 install.log
-rw-r--r--. 1 root root 11439 Aug 31 02:13 install.log.syslog
drwxr-xr-x. 2 root root  4096 Aug 31 02:48 Music
drwxr-xr-x. 2 root root  4096 Aug 31 02:48 Pictures
drwxr-xr-x. 2 root root  4096 Aug 31 02:48 Public
drwxr-xr-x. 2 root root  4096 Aug 31 02:48 Templates
drwxr-xr-x. 2 root root  4096 Aug 31 02:48 Videos
```

3. **View Hidden Files:**

- List all files including hidden file starting with '.'.

 # ls -a

```
.bashrc  Documents .gconfd install.log    .nautilus .pulse-cookie
.cache  Downloads  .gnome2  install.log.syslog  .netstat.swp
.recently-used.xbel
0001.pcap .config .elinks .gnome2_private   .kde   .opera .spice-vdagent
anaconda-ks.cfg   .cshrc  .esd_auth  .gtk-bookmarks
.libreoffice   Pictures   .tcshrc
.bash_history.dbus.fbcmd.gvfs.local.pki     Templates
.bash_logout Desktop  fbcmd_update.php.ICEauthority.mozilla  Public
Videos
.bash_profile.digrc.gconf   index.html  Music.pulse.wireshark
```



4. List Files with Human Readable Format with option -lh

- With combination of **-lh** option, shows sizes in human readable format.

```
# ls -lh
total 176K
-rw-r--r--. 1 root root  683 Aug 19 09:59 0001.pcap
-rw-------. 1 root root 1.6K Jul 31 02:17 anaconda-ks.cfg
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Desktop
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Documents
drwxr-xr-x. 4 root root 4.0K Aug 16 02:55 Downloads
-rw-r--r--. 1 root root  21K Aug 12 12:42 fbcmd_update.php
-rw-r--r--. 1 root root  46K Jul 31 09:58 index.html
-rw-r--r--. 1 root root  48K Jul 31 02:17 install.log
-rw-r--r--. 1 root root  12K Jul 31 02:13 install.log.syslog
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Music
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Pictures
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Public
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Templates
drwxr-xr-x. 2 root root 4.0K Jul 31 02:48 Videos
```

5. List Files and Directories with '/' Character at the end:

- Using **-F** option with **ls** command, will add the **'/'** Character at the end each directory.

```
# ls -F
0001.pcap        Desktop/    Downloads/        index.html
install.log.syslog  Pictures/  Templates/
anaconda-ks.cfg  Documents/  fbcmd_update.php  install.log  Music/
Public/    Videos/
```

6. List Files in Reverse Order:

- The following command with **ls -r** option display files and directories in reverse order.

```
# ls -r
Videos  Public  Music  install.log  fbcmd_update.php  Documents  anaconda-
ks.cfg
Templates  Pictures  install.log.syslog  index.html  Downloads
Desktop  0001.pcap
```


7. Recursively list Sub-Directories:

- **ls -R** option will list very long listing directory trees. See an example of output of the command.

```
# ls -R
total 1384
-rw-------. 1 root     root      33408 Aug  8 17:25 anaconda.log
-rw-------. 1 root     root      30508 Aug  8 17:25 anaconda.program.log
./httpd:
total 132
-rw-r--r--  1 root root     0 Aug 19 03:14 access_log
-rw-r--r--. 1 root root 61916 Aug 10 17:55 access_log-20120812
./lighttpd:
total 68
-rw-r--r--  1 lighttpd lighttpd  7858 Aug 21 15:26 access.log
-rw-r--r--. 1 lighttpd lighttpd 37531 Aug 17 18:21 access.log-20120819
./nginx:
total 12
-rw-r--r--. 1 root root    0 Aug 12 03:17 access.log
-rw-r--r--. 1 root root  390 Aug 12 03:17 access.log-20120812.gz
```

8. Reverse Output Order:

- With combination of **-ltr** will shows latest modification file or directory date as last.

```
# ls -ltr
total 176
-rw-r--r--. 1 root root 11439 Jul 31 02:13 install.log.syslog
-rw-r--r--. 1 root root 48867 Jul 31 02:17 install.log
-rw-------. 1 root root  1586 Jul 31 02:17 anaconda-ks.cfg
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Desktop
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Videos
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Templates
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Public
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Pictures
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Music
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Documents
-rw-r--r--. 1 root root 46701 Jul 31 09:58 index.html
-rw-r--r--. 1 root root 21262 Aug 12 12:42 fbcmd_update.php
drwxr-xr-x. 4 root root  4096 Aug 16 02:55 Downloads
-rw-r--r--. 1 root root   683 Aug 19 09:59 0001.pcap
```

9. **Sort Files by File Size:**

- With combination of **-lS** displays file size in order, will display big in size first.

```
# ls -lS
total 176
-rw-r--r--. 1 root root 48867 Jul 31 02:17 install.log
-rw-r--r--. 1 root root 46701 Jul 31 09:58 index.html
-rw-r--r--. 1 root root 21262 Aug 12 12:42 fbcmd_update.php
-rw-r--r--. 1 root root 11439 Jul 31 02:13 install.log.syslog
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Desktop
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Documents
drwxr-xr-x. 4 root root  4096 Aug 16 02:55 Downloads
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Music
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Pictures
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Public
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Templates
drwxr-xr-x. 2 root root  4096 Jul 31 02:48 Videos
-rw-------. 1 root root  1586 Jul 31 02:17 anaconda-ks.cfg
-rw-r--r--. 1 root root   683 Aug 19 09:59 0001.pcap
```

6.7 MANIPULATING STRINGS

- **String manipulation** (or **string handling**) is the process of changing, parsing, splicing, pasting, or analysing strings. String manipulation typically comes as a mechanism or a library feature of in most programming languages.
- Typically, most programming languages provide a **string data type** that holds a sequence of characters.
- Following are list of Bash String Operations explained in detail with examples:

1. **Bash String Length:**

- To find String Length in Bash Scripting use one of the following syntax:

(i) ${#string}

(ii) expr length "$string" Observe the double quotes around $string . If your string has spaces, double quotes around $string is kind of mandatory, else in other cases you may ignore. However, to be on the safe side, always try including double quotes around.

(iii) expr "$string":'*'

Example 6.14: Write a Program to find string length. Using ${#string_variable_name}.

```
#!/bin/bash
str="Good morning"
length=${#str}
echo "Length of '$str' is $length"
```

Output:

```
Length of 'Good morning' is 12
```


2. **Bash Strings Equal:**

- To check if two strings are equal in bash scripting.
- To check if two strings are equal in bash scripting, use **bash if** statement and double equal to== operator.
- To check if two strings are not equal in bash scripting, use bash if statement and not equal to!= operator.

Example 6.15: Script to check if two strings are equal in Bash Scripting.

```
bash-strings-equal-example
#!/bin/bash
str1="Learn Bash"
str2="Learn Bash"
if [ "$str1" == "$str2" ]; then
    echo "Both Strings are Equal."
else
    echo "Both Strings are not Equal."
fi
```

Output:

```
Both Strings are Equal.
```

3. **Bash Split String:**

- When working with string literals or message streams, we come across a necessity to split a string into tokens using a delimiter.
- The delimiter could be a single character or a string with multiple characters. In this tutorial, we shall learn how to split a string in bash shell scripting with a delimiter of single and multiple character lengths.

(a) Split String with single character delimiter(s) in Bash using IFS

- To split a string in bash using IFS, follow the below steps:

1. Set IFS to the delimiter you would want IFS='<delimiter>' IFS is an internal variable that determines how Bash recognizes word boundaries. The default value of IFS is white space. If you set it to some other value, reset it to default whitespace.
2. Read your string to a variable with options -ra. Read-ra ARR<<<"$str".

Table 6.2: Options

Option	Description
-r	Backslash does not act as an escape character.
-a ARR	The words (separated by IFS) are assigned to the sequential index of array ARR beginning at zero.

3. Now you have your string split by the delimiter (set in IFS) stored in array ARR. ARR is just array name. Any string literal that is valid could be used as an array name.
4. You may now access the tokens split into an array using a **bash for loop.**

Example 6.16: Bash Split String by Space. In this example, we split a string using space as a character delimiter.

```
bash-split-string-example
#!/bin/bash
str="Learn to Split a String in Bash Scripting"
IFS=' ' # space is set as delimiter
read -ra ADDR <<< "$str" # str is read into an array as tokens separated by IFS
for i in "${ADDR[@]}"; do # access each element of array
    echo "$i"
done
```

Output:

```
Learn
to
Split
a
String
in
Bash
Scripting
```

Bash Split a String with multiple character delimiter:

- To split a string using bash script with different scenarios based on delimiter: like single character delimiter and multiple character delimiters.
- To bash shell scripting and are not familiar with idiomatic expressions, following is an easily understandable shell script with basic **bash if**, **bash while** and **bash substring** methods. **Comments** are provided at each step to make the script file more readable.
- The split strings are stored in the array and could be accessed using an index.
- To compute substring of a string given starting position and length of substring.
- **Syntax:** ${string:position:length}
- Providing length is optional. If length is not specified, end of the string is considered end of the substring.

Example 6.17: Compute substring provided position and length of substring.

```
Bash Script File
#!/bin/bash
str="TutorialKart"
subStr=${str:4:6}
echo $subStr
```

Output:

```
rialKa
```


Example 6.18: Compute substring provided position Bash Script File.

```
#!/bin/bash
str="TutorialKart"
subStr=${str:6}
echo $subStr
```

Output:

```
alKart
```

4. **Concatenate Strings in Bash:**

- In the following example, we use the idea of including variables in a string to concatenate two variables.

Example 6.19: Script for Concatenate Strings.

```
#!/bin/bash
n1=10
str1="Number of Apples: "
str1="$str1$n1"
echo $str1
```

Output:

```
Number of Apples: 10
```

6.8 READING AND WRITING FILES

- Reading and writing files in Linux is simple, you just use the standard utilities for reading files such as **cat**, **grep**, **tail**, **head**, **awk** etc.. And you primarily use the output redirect operator > and standard commands like **sed** for writing files.
- To write output of Bash Command to Log File, you may use right angle bracket symbol (>) or double right angle symbol (>>).
- **Right angle braketsymbol (>)**: It is used to write output of a bash command to a disk file. If the file is not already present, it creates one with the name specified. If the file is already present, the content of the file would be overwritten.
- **Double right angle symbol (>>)**: It is used to append data to an existing file. If the file is not already present, a new one would be created.
- When you are writing to the file for the first time, and want no previous data to be in the file if it is already present, use (>) to make sure that it overwrites the content. And in the later script, you may use (>>) to append to the log file.
- To write a program that interacts with a user data file called users.dat. There are several options for running this script. If the first parameter of the script is -a then it should be followed by user id, first name, last name and age. Then it will write that information to the file. If the first parameter is -l then it will print out the list of user data.

```
#!/bin/sh
OP=$1
if [ "$1" == "-a" ]; then
   ID=$2
   FIRST=$3
   LAST=$4
   AGE=$5
   echo "$ID,$FIRST,$LAST,$AGE" >> users.dat
   echo "User Added"
elif [ "$1" == "-l" ]; then
   cat users.dat
fi
```

- The code for this is very simple. To add a user you simple use **echo** to print out the fields with commas between them, then you redirect the output using the **>>** operator. Using the **>** will redirect the output *STDOUT* to a file and overwrite the entire file, this is why we use **>>** instead, because it will append to a file instead of overwriting the file. And to print out the file, we simply use the **cat** command, which will print out a file to the console.
- Users to have a test data set. Let's call the script users.sh.

```
./users.sh -a apatil Achal Patil 17
./users.sh -a asalunkhe Ananya Salunkhe 18
./users.sh -a anisal Aditi Nisal 17
./users.sh -a anikam Aditya Nikam 20
./users.sh -a apatil Athara Patil 23
./users.sh -a ypawar Yash Pawar 12
./users.sh -a mpawar mrudul pawar 10
```

- This gave us a nice data set, so now if we want to print out the users, we can use the **-l** option which gives us this list of data:

```
apatil Achal Patil 17
asalunkhe Ananya Salunkhe 18
anisal Aditi Nisal 17
anikam Aditya Nikam 20
apatil Athara Patil 23
ypawar Yash Pawar 12
mpawar mrudul pawar 10
```

Writing a Script:

- A shell script is a file that contains ASCII text.
- To create a shell script, we use a text editor. A text editor is a program, like a word processor, that reads and writes ASCII text files.

- There are many text editors available for Linux systems, both for the command line and GUI environments. Here is a list of some common ones:

Table 6.3: Text editor available for Linux system.

Name	Description	Interface
vi, vim	The granddaddy of Unix text editors, **vi**, is infamous for its obtuse user interface. **vi** is powerful, lightweight, and fast. Learning **vi** is a Unix rite of passage, Linux distributions, an enhanced version of **vi** called **vim** is provided in place of vi. **vim** is a remarkable editor and well worth taking the time to learn it.	command line
Emacs	Emacs contains (or can be made to contain) every feature ever conceived of for a text editor. It should be noted that **vi** and Emacs fans fight bitter religious wars over which is better.	command line
nano	**nano** is a free clone of the text editor supplied with the **pine** email program. **nano** is very easy to use but is very short on features compared to **vim** and **emacs**. **nano** is recommended for first-time users who need a command line editor.	command line
gedit	**gedit** is the editor supplied with the GNOME desktop environment. gedit is easy to use and contains enough features to be a good beginners-level editor.	graphical
kwrite	**kwrite** is the "advanced editor" supplied with KDE. It has syntax highlighting, a helpful feature for programmers and script writers.	graphical

- Let's fire up our text editor and type in our first script as follows:

```
#!/bin/bash
# My first script
echo "Hello World!"
```

- The first line of the script is important. In this case, /bin/bash. Other scripting languages such as Perl, awk, tcl, Tk, and python also use this mechanism.
- The second line is a *comment*. Everything that appears after a "#" symbol is ignored by **bash**. As our scripts become bigger and more complicated, comments become vital. They are used by programmers to explain what is going on so that others can figure it out. The last line is the echo command. This command simply prints its arguments on the display.

Setting Permissions:

- The next thing we have to do is give the shell permission to execute our script. This is done with the chmod command as follows:

```
[me@linuxbox me]$ chmod 755 hello_world
```

- The "755" will give us read, write, and execute permission. Everybody else will get only read and execute permission. To make the script private, (i.e., only we can read and execute), use "700" instead.

```
[me@linuxbox me]$ ./hello_world
```

Output:

```
"Hello World!" displayed.
```

- This list of directories is called our *path*. List of directories with the following command:

```
[me@linuxbox me]$ echo $PATH
```

- To execute our new script, we specified a pathname ("./") to the file.
- We can add directories to our path with the following command, where *directory* is the name of the directory we want to add:

```
[me@linuxbox me]$ export PATH=$PATH:directory
```

- A better way would be to edit our .bash_profile file to include the above command. That way, it would be done automatically every time we log in.
- Linux distributions encourage a practice in which each user has a specific directory for the programs he/she personally uses.
- Directory is called bin and is a subdirectory of our home directory. If we do not already have one, we can create it with the following command:

```
[me@linuxbox me]$ mkdir ~/bin
```

- Script into our new bin directory and type and our script will run.

```
[me@linuxbox me]$ hello_world
```

6.9 POSITIONAL PARAMETERS

- Functions are like mini-scripts: they can accept parameters, they can use variables only known within the function (using the local shell built-in) and they can return values to the calling shell.
- A function also has a system for interpreting positional parameters. However, the positional parameters passed to a function are not the same as the ones passed to a command or script.
- When a function is executed, the arguments to the function become the positional parameters during its execution. The special parameter # that expands to the number of positional parameters is updated to reflect the change. Positional parameter 0 is unchanged.
- The Bash variable FUNCNAME is set to the name of the function, while it is executing.
- If the return built-in is executed in a function, the function completes and execution resumes with the next command after the function call. When a function completes, the values of the positional parameters and the special parameter # are restored to the values they had prior to the function's execution. If a numeric argument is given to return, that status is returned.

Example:

```
#!/bin/bash
echo "This script demonstrates function arguments." echo
 echo "Positional parameter 1 for the script is $1." echo
test ()
 {
echo "Positional parameter 1 in the function is $1." RETURN_VALUE=$? echo
"The exit code of this function is $RETURN_VALUE."
}
 test other_param
 [lydia@cointreau ~/test] ./showparams.sh parameter1
```

- This script demonstrates function arguments.
- Positional parameter 1 for the script is parameter1.
- Positional parameter 1 in the function is other_param.
- The exit code of this function is 0.
- The return value or exit code of the function is often store in a variable, so that it can be probed at a later point. The init scripts on your system often use the technique of probing the RETVAL variable in a conditional test, like this one:
- if [$RETVAL -eq 0]; then Or like this example from the /etc/init.d/amd script, where Bash's optimization features are used:

```
[ $RETVAL = 0 ] && touch /var/lock/subsys/amd
```

- The commands after && are only executed when the test proves to be true; this is a shorter way to represent an if/then/fi structure. The return code of the function is often used as exit code of the entire script.

6.10 CASE STATEMENT

- The case statement is another form of n-way selection statement which is similar in many ways to the if-then-elif-else statement. The instruction compares a value stored in a variable (or the result of an expression) against a series of enumerated lists. Upon finding a match, the corresponding action(s) is(are) executed. The case statement (known as a switch in C/C++/Java) is more restricted in other languages than it is a Bash.

Syntax:

```
case expression in
   list1) action(s);;
   list3) action(s);;
   ....
   listn) action(s);;
esac
```

- Notice the syntax. First, each action ends with two consecutive semicolons. Second, the enumerated list ends with the close parenthesis sign. Finally, the entire statement ends with an esac statement.

Example 6.20: Shell script for *case* statement.

```
sum=0
i=\"y\"
echo \" Enter one no.\"
read n1
echo \"Enter second no.\"
read n2
while [ $i = \"y\" ]
do
echo \"1.Addition\"
echo \"2.Subtraction\"
echo \"3.Multiplication\"
echo \"4.Division\"
echo \"Enter your choice\"
read ch
case $ch in
   1)sum=\`expr $n1 + $n2\`
   echo \"Sum =\"$sum;;
   2)sum=\`expr $n1 - $n2\`
   echo \"Sub = \"$sum;;
   3)sum=\`expr $n1 \\* $n2\`
   echo \"Mul = \"$sum;;
   4)sum=\`expr $n1 / $n2\`
   echo \"Div = \"$sum;;
   *)echo \"Invalid choice\";;
esac
echo \"Do u want to continue ?\"
read i
if [ $i != \"y\" ]
then
    exit
fi
done
```

Output:

```
Enter one no.
21
Enter second no.
58
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
1
Sum =79
```


6.11 REAL TIME SCRIPTS FOR DIFFERENT SYSTEM ADMINISTRATION ACTIVITIES

Linux administrative commands and its uses:

(a) Uptime Command:

- In Linux **uptime** command shows since how long your system is running and the number of users are currently logged in and also displays load average for **1, 5** and **15** minutes intervals.

1. **uptime command:** Tell how long the **Linux** system has been running.
2. **w command:** Show who is logged on and what they are doing including the **uptime** of a **Linux** box.
3. **top command:** Display **Linux** server processes and display system **Uptime** in **Linux** too.

```
# uptime
08:16:26 up 22 min, 1 user, load average: 0.00, 0.03, 0.22
```

Check Uptime Version:

- **Uptime** command don't have other options other than **uptime** and **version**. It gives information only in **hours:mins** if it less than **1** day.

```
[tecmint@tecmint ~]$ uptime -V
procps version 3.2.8
```

Example:

```
[mistersubha@server-1 ~]$uptime
 08:24:37 up 207 days, 11:10,  0 users,  load average: 0.00, 0.03, 0.05
Output:
[mistersubha@server-1 ~]$
[mistersubha@server-1 ~]$uptime
08:24:37 up 207 days, 11:10,  0 users,  load average: 0.00, 0.03, 0.05
[mistersubha@server-1 ~]$
```

(b) W Command:

- This command is used to will display users currently logged in and their process along-with shows **load averages**. Also shows the **login name**, **tty name**, **remote host**, **login time**, **idle time**, **JCPU**, **PCPU**, command and processes.

```
# w
08:27:44 up 34 min,  1 user,  load average: 0.00, 0.00, 0.08
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
tecmint  pts/0    192.168.50.1      07:59    0.00s  0.29s  0.09s w
```

Available options:

- **-h:** displays no header entries.
- **-s:** without JCPU and PCPU.
- **-f:** Removes from field.
- **-V:** (upper letter) – Shows versions.

(c) # users:

- Users command displays currently logged in users.
- **Syntax:**

```
useradd -m -p EncryptedPasswordHere username
```

Where,

-m: The user's home directory will be created if it does not exis

-p EncryptedPasswordHere: The encrypted password, as returned by crypt().

username: Add this user to the Linux system,

- **Create an encrypted password:** To create an encrypted password using Perl crypt() as follows:

```
crypt($plain, $salt)
## perl one liner ##
perl -e 'print crypt("Your-Clear-Text-Password-Here", "salt"),"\n"'
```

- crypt() is a one-way hash function. The PLAINTEXT ($plain) and SALT are turned into a short string, called a digest, which is returned. The same PLAINTEXT and SALT will always return the same string, but there is no (known) way to get the original PLAINTEXT from the hash. Small changes in the PLAINTEXT or SALT will result in large changes in the digest.

Example: `perl -e 'print crypt("2IL@ove19Pizza4_", "salt"),"\n"'`

Sample outputs: `sa.KT9zrGYeg2`

(d) Who Command:

- **who** command simply return **user name**, **date**, **time** and **host information**. who command is similar to **w** command. Unlike **w** command **who** doesn't print what users are doing.

Illustrate the different between **who** and **w** commands.

```
# who
tecmint  pts/0   2020-09-18 07:59 (192.168.50.1)
# w
08:43:58 up 50 min,  1 user,  load average: 0.64, 0.18, 0.06
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
tecmint  pts/0    192.168.50.1     07:59    0.00s  0.43s  0.10s w
```

Who command Options:

- **-b:** Displays last system reboot date and time.
- **-r:** Shows current runlet.
- **-a, –all:** Displays all information in cumulatively.

(e) Whoami Command:

- **whoami** command print the name of current user. You can also use "**who am i**" command to display the current user. If you are logged in as a root using sudo command "**whoami**" command return **root** as current user. Use "**who am i**" command if you want to know the exact user logged in.

```
# whoami
```


(f) Less Command:

- **less** command allows quickly view file. You can page up and down. Press '**q**' to quit from less window.

```
# less install.log
Installing setup-2.8.14-10.el6.noarch
warning: setup-2.8.14-10.el6.noarch: Header V3 RSA/SHA256 Signature, key
ID c105b9de: NOKEY
Installing filesystem-2.4.30-2.1.el6.i686
Installing ca-certificates-2010.63-3.el6.noarch
Installing xml-common-0.6.3-32.el6.noarch
Installing tzdata-2010l-1.el6.noarch
Installing iso-codes-3.16-2.el6.noarch
```

(g) More Command:

- **more** command allows quickly view file and shows details in percentage. You can page up and down. Press '**q**' to quit out from more window.

```
# more install.log
Installing setup-2.8.14-10.el6.noarch
warning: setup-2.8.14-10.el6.noarch: Header V3 RSA/SHA256 Signature, key
ID c105b9de: NOKEY
Installing filesystem-2.4.30-2.1.el6.i686
Installing ca-certificates-2010.63-3.el6.noarch
Installing xml-common-0.6.3-32.el6.noarch
Installing tzdata-2010l-1.el6.noarch
Installing iso-codes-3.16-2.el6.noarch
--More--(10%)
```

(h) CP Command:

- Copy file from source to destination preserving same mode.

```
# cp -p fileA fileB
```

You will be prompted before overwrite to file.

```
# cp -i fileA fileB
```

(i) Cat Command:

- **cat** command used to view multiple file at the same time.

```
# cat fileA fileB
```

- You combine **more** and **less** command with cat command to view file contain if that doesn't fit in single screen / page.

```
# cat install.log | less
# cat install.log | more
```

(j) Cd command (change directory):

- cd command (change directory) it will goes to **fileA** directory.

```
# cd /fileA
```

Writing a script to display system information:

Create a directory to store our shell scripts:

```
# mkdir scripts
# cd scripts
```

Open a new text file named system_info.sh with your preferred text editor. We will begin by inserting a few comments at the top and some commands afterwards:

```
#!/bin/bash
# Sample script written for Part 4 of the RHCE series
# This script will return the following set of system information:
# -Hostname information:
echo -e "\e[31;43m***** HOSTNAME INFORMATION *****\e[0m"
hostnamectl
echo ""
# -File system disk space usage:
echo -e "\e[31;43m***** FILE SYSTEM DISK SPACE USAGE *****\e[0m"
df -h
echo ""
# -Free and used memory in the system:
echo -e "\e[31;43m ***** FREE AND USED MEMORY *****\e[0m"
free
echo ""
# -System uptime and load:
echo -e "\e[31;43m***** SYSTEM UPTIME AND LOAD *****\e[0m"
uptime
echo ""
# -Logged-in users:
echo -e "\e[31;43m***** CURRENTLY LOGGED-IN USERS *****\e[0m"
who
echo ""
# -Top 5 processes as far as memory usage is concerned
echo -e "\e[31;43m***** TOP 5 MEMORY-CONSUMING PROCESSES *****\e[0m"
ps -eo %mem,%cpu,comm --sort=-%mem | head -n 6
echo ""
echo -e "\e[1;32mDone.\e[0m"
```

Summary

- *Linux Shell scripting is writing a series of command for the shell to execute. Shell variables store the value of a string or a number for the shell to read. Shell scripting in Linux can help you create complex programs containing conditional statements, loops, and functions.*
- *A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data. A variable is nothing more than a pointer to the actual data. The shell enables you to create, assign, and delete variables.*

- Flow control gives a programmer the power to specify that only certain portions of a program run, or that certain portions run repeatedly, according to conditions such as the values of variables, whether or not commands execute properly, and others. We call this the ability to control the flow of a program's execution.
- A for loop is classified as an iteration statement i.e. it is the repetition of a process within a bash script. For example, you can run UNIX command or task 5 times or read and process list of files using a for loop. A for loop can be used at a shell prompt or within a shell script itself.
- Shell functions are a way to group commands for later execution using a single name for the group. They are executed just like a "regular" command. When the name of a shell function is used as a simple command name, the list of commands associated with that function name is executed.
- Positional parameters are a series of special variables ($0 through $9) that contain the contents of the command line.
- case... esac statement is to give an expression to evaluate and to execute several different statements based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Check Your Understanding

1. Which of the following is not a type of shell?
   (a) The C Shell (b) The Korn Shell
   (c) The Bourne Shell (d) The Perl Shell
2. Copy all the .doc extension files with file name having only 3 characters into the directory called "confi", which is in parent directory.
   (a) cp ???.doc ../confi (b) cp [1-3].doc ../confi
   (c) cp ???.doc /confi (d) None of the above.
3. What is a shell script?
   (a) Group of commands.
   (b) A file containing special symbols.
   (c) A file containing a series of commands.
   (d) Group of functions.
4. Which command is used for making the scripts interactive?
   (a) Ip (b) Input
   (c) Read (d) Write
5. What are positional parameters?
   (a) Special variables for assigning arguments from the command line.
   (b) Pattern matching parameters.
   (c) Special variables for reading user input.
   (c) Special variables and patterns.
6. Which of the following is used for storing the number of positional parameters?
   (a) $n (b) $#
   (c) $* (d) $2

7. Which of the following loop statements uses do and done keyword?
   (a) For (b) While
   (c) Case (d) For and while
8. Which command is used by the shell for manipulating positional parameters?
   (a) Set (b) Cut
   (c) Case (d) paste
9. The system administrator is also known as----------.
   (a) master user (b) super user
   (c) root user (d) master and super user
10. Which of the following functionalities is carried by the root user?
   (a) Managing disk space.
   (b) Performing backup.
   (c) Changing attributes of a file.
   (d) Managing disk space, performing backup, changing attributes of a file.

Answer Key

1. (d)	2. (a)	3. (c)	4. (c)	5. (c)	6. (b)	7. (d)	8. (a)	9. (d)	10. (d)

Practice Questions

Q.I Answer the following Questions in short.

1. What are the real-life scenarios for implementing Input/ Output Redirection in shell scripting?
2. What is shell script? Why it is required?
3. Which commands is used if you want to perform a set of instructions repeatedly?
4. List all current aliases.
5. Explain the difference between the following commands: echo and /bin/echo.

Q.II Answer the following Questions:

1. How to learn shell scripting in Linux?
2. What extension is given to a Bash Script File?
3. Describe the system administrative command?
4. Explain the Function and its example?
5. What is case statement? With example?

Q.III Write short note on.

1. Bash variable
2. Reading and writing files
3. Manipulating strings
4. Loops
5. Test statement

■■■