DIPLOMA OF ASSOCIATE ENGINEER
3RD YEAR
COMPUTER INFORMATION TECHNOLOGY
APPROVED BY TEVTA PUNJAB

A TEXT BOOK OF

OPERATING SYSTEM

CIT-333

Developed By
Academics Wing
Technical Education & Vocational
Training Authority Punjab

OPERATING SYSTEM

CIT-333

FOR DAE 3rd YEAR

TECHNICAL EDUCATION & VOCATIONAL TRAINING AUTHORITY PUNJAB

PREFACE

The text book has been written to cover the syllabus of Operating System, 3rd Year D.A.E (Computer Information Technology) according to the new scheme of studies. The book has been written in order to cater the needs of latest concepts and needs of the course i.e. Operating System and to be able to attempt D.A.E Examination of PBTE Lahore.

The aim of bringing out this book is to enable the students to have sound knowledge of the subject. Every aspect has been discussed to present the subject matter in the most concise, compact lucid & simple manner to help the subject without any difficulty. Frequent use of illustrative figures has been made for clarity. Short Questions and Self-tests have also been included at the end of each chapter which will serve as a quick learning tool for students.

The author would like to thank the reviewers whose valuable recommendations have made the book more readable and understandable. Constructive criticisms and suggestions for the improvements in future are welcome.

AUTHORS

MANUAL DEVELOPMENT COMMITTEE

Miss Ayesha Iqbal

Dy. Director (R&D) TEVTA Secretariat Lahore
(CONVENER)

Engr. Tahir Javed

Instructor CIT- Govt. College of Technology, Taxila.
(MEMBER)

ENGR. HAFIZ TAIMOOR UL HASSAN

Instructor CIT- Govt. College of Technology, Bahawalpur.
(MEMBER)

CIT -333 OPERATING SYSTEM

Total Contact Hours		T	P	C
Theory:	64 Hours	2	3	3
Practical:	96 Hours			

Pre-requisites: **Operating System Concepts**

AIMS: This course has been designed to enable students to be familiar with:

1. Understand the concepts and issues of managing the resources of a computer by an operating system.
2. Understand the operational principles and implementation of Unix/Linux/Linux operating system
3. Use shell commands to administer the system.
4. Run application programs in Unix/Linux/Linux environment
5. Perform system administer
6. Demonstrate motivation to use and administer systems in Unix/Linux/Linux platform.

Table of Contents

TEXT/REFERENCE BOOKS

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar

CHAPTER 01 COMPUTER SYSTEM STRUCTURE

Objectives
After completion of this chapter students will be able to:

1.1. Computer System Operation
1.2. I/O Structure
1.3. Storage Structure and Hierarchy
1.4. Hardware Protection

1. INTRODUCTION

An operating system is a program that acts as an intermediary between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs. The primary goal of an operating system is thus to make the computer system convenient to use. A secondary goal is to use the computer hardware in an efficient manner.

Basic Concept of Operating system

Software

A set of instruction given to the computer to perform a specific task or solve a problem called software or program.

Software is the combination of instructions to perform the specific task. Software is also called program.

Hierarchy and types of software

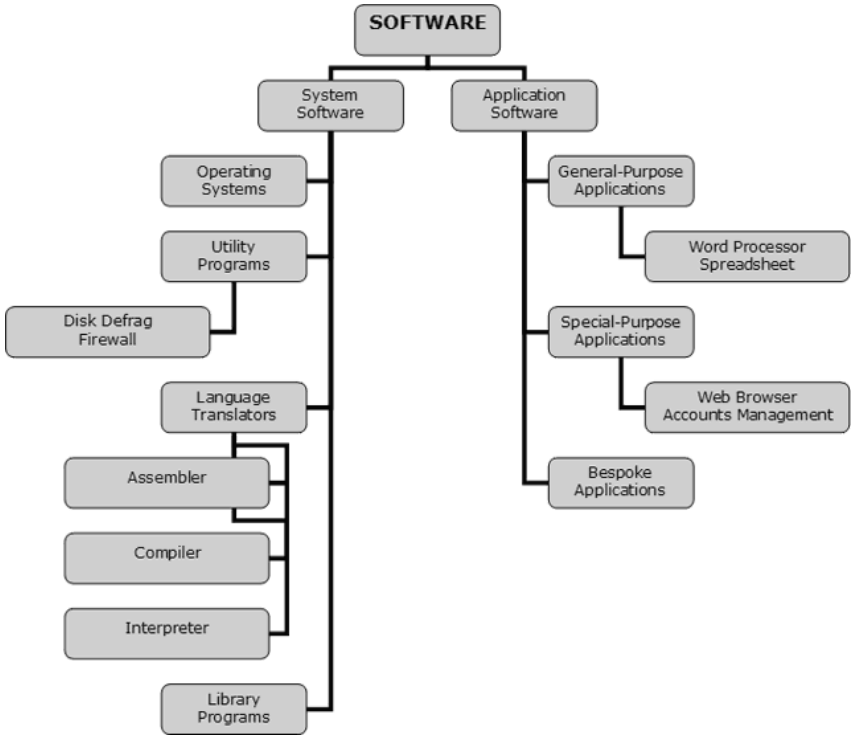


Figure 1 Hierarchy and types of software

Operating system

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs. It enables the computer hardware to communicate and operate. DOS and Windows example of OS.

Utility programs

A program that performs a specific task related to the management of computer function, resources, or file, password protection and virus protection. Utility program help manage, maintain and control computer

resources. Example: File manager, image viewer, Diagnostic utility, PC maintenance utility, Antivirus, Disk cleanup and Firewall etc.

Device Driver

A device Driver is a computer program that operates or controls a particular type of device that is attached to a computer like printers

Language Translators

whatever language or type of language we use to write our programs, they need to be in machine code in order to be executed by the computer. There are 3 main categories of translator.

- **Compiler**

The language translator program that translates the complete source program into machine code as a whole is called compiler. The C and C++ compilers are best examples are compilers.

- **Interpreter**

The language translator program that translates the source code into machine code statement by statement is called interpreter.

- **Assembler**

The language translator program that translates the program written in assembly language into machine code is called Assembler.

Library Programs

Library programs are compiled libraries of commonly-used routines. On a Windows system they usually carry the file extension dll and are often referred to as run-time libraries. The libraries are run-time because they are called upon by running programs when they are needed. When you program using a run-time library, you typically add a reference to it either in your code or through the IDE in which you are programming. Using library programs saves

time when programming. It also allows the programmer to interact with proprietary software without having access to its source code.

Application software

An application software or group of programs that is designed for end user. It is used to perform various applications on computer, also known as application package. Ms Word, Access Photoshop is the example of Application software.

- **General-Purpose Software**

Software is general-purpose if it can be used for lots of different tasks. You can use a word processor to write letters, memos, essays, instructions, notes, faxes, invoices and lots more. These days we tend to use integrate suites of office software where a range of general-purpose software is provided, usually with the facility to combine elements from each application in a single file.

- **Special-Purpose Software**

This software performs a single specific task. This task might be complex like payroll calculation, stock control etc. but will be based on a single task.

- **Bespoke Software**

Bespoke software is written for a single client. Large organizations have a need for well-developed applications suited to their specific needs. Such software is often expensive to develop since the development costs are not shared among a large number of people purchasing the software. It is also known as custom software.

- **Packaged Software**

The software that is developed for sale to the general public is called packaged software. Packaged software is used to solve some common problem of many people or users.

WHAT IS AN OPERATING SYSTEM?

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

Operating system is a computer software and interface between hardware and user, it enables the computer hardware to communicate and operate. DOS, Windows, UNIX, Linux and Mac are today's core operating systems; they all run from big systems to Smart phone &handheld devices to ease human work. A computer system can be divided roughly into four components shown in figure 1.

- The hardware
- The operating system
- The application programs
- The users

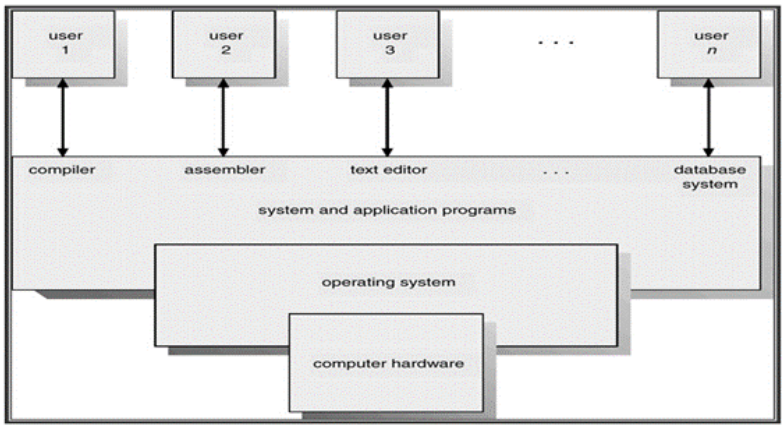


Figure 2. Abstract view of the components of a computer system.

Components of Computer System

A computer system can be divided into four components, which are as follows

Hardware: The hardware is the physical part which we can touch and feel, the central processing unit (CPU), the memory, and the input/output (I/O) devices are the basic computing resources of a computer system.

Application programs: Application programs are user or programmer created programs like compilers, database systems, games, and business programs that define the ways in which these resources can be used to solve the computing problems of the users.

Users: There are different types of users like people, machines, and even other computers which are trying to solve different problems.

Operating system: An operating system is the interface between the user and the machine which controls and coordinates the use of the hardware among the various application programs for the various users.

The hardware is the central processing unit (CPU), the memory, and the input/output (I/O) devices provides the basic computing resources. The application programs such as compilers, database systems, games, and business programs define the ways in which these resources are used to solve the computing problems of the users. There may be many different users (people, machines, other computers) trying to solve different problems. Accordingly, there may be many different application programs. The operating system controls and coordinates the use of the hardware among the various application programs for the various users.

An operating system is similar to a government. The components of a computer system are its hardware, software, and data. The operating system provides the means for the proper use of these resources in the operation of the computer system. Like a government, the operating system performs no useful function by itself. It simply provides an environment within which other programs can do useful.

We can view an operating system as a resource allocator. A computer system has many resources (hardware and software) that may be required to solve a problem: CPU time, memory space, file storage space, I/O devices, and so on.

The operating system acts as the manager of these resources and allocates them to specific programs and users as necessary for tasks. Since there may be many possibly conflicting requests for resources, the operating system must decide which requests are allocated resources to operate the computer system efficiently and fairly.

OS Environment

Operating systems are designed to meet multiple applications in various environments. OS nature can be Stand-alone, Multiple Users, Multitasking or Real Time Operating System.

Types of operating systems

There are different types of operating system because of the architectural difference between computers

- **Batch systems (early ones):**

A batch operating system is a type of operating system that processes a large volume of similar jobs in batches without manual intervention. In a batch operating system, jobs are collected in a batch and then processed in a sequence without requiring any user interaction.

In a batch operating system, jobs are typically submitted to a queue, where they are held until the system is ready to process them. Once a job is selected for processing, it is loaded into the system's memory, and the operating system allocates the necessary resources such as CPU time, memory, and I/O devices to execute the job.

- **Multi programmed systems:**

when two or more programs are in memory sharing the processor. It increases CPU utilization/The act of using/ by organizing jobs so that it always has one to execute.

- **Timesharing (multi-tasking):**

supports interactive users by allowing many users to share the computer simultaneously. In time sharing systems each user is given a time slice for executing his job until the time slice ends.

- **Multiprocessor systems:**

Have more than one processor in close communication. They share the computer bus, system clock and I/O devices and sometimes memory. It is possible for two processes run in parallel.

- **Distributed Systems:**

Depend on networking for their operation and runs on and controls the resources of multiple machines. It provides resource sharing across the boundaries of single computer systems. Distributed operating system looks to users like a single machine operating systems that owns the whole network and makes it look like a virtual uniprocessor.

- **Real Time Systems:**

Were originally used to control autonomous/Existing as an independent entity/ system such as satellites, robots and communication systems. It is one that must react to input and responds them quickly. It also has well defined, fixed time constraint.

1.1. COMPUTER SYSTEM OPERATION.

A modern, general-purpose computer system consists of a CPU and a number of device controllers that are connected through a common bus that provides access to shared memory (Figure 2). Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, and video

displays). The CPU and the device controllers can execute concurrently, competing for memory cycles. To ensure orderly access to the shared memory, a memory controller is provided whose function is to synchronize access to the memory.

computer to start running: for instance, when it is powered up -or rebooted it needs to have an initial program to run. This initial program, or bootstrap program, tends to be simple. It initializes all aspects of the system, from CPU registers to device controllers to memory contents. The bootstrap program must know how to load the operating system and to start executing that system.

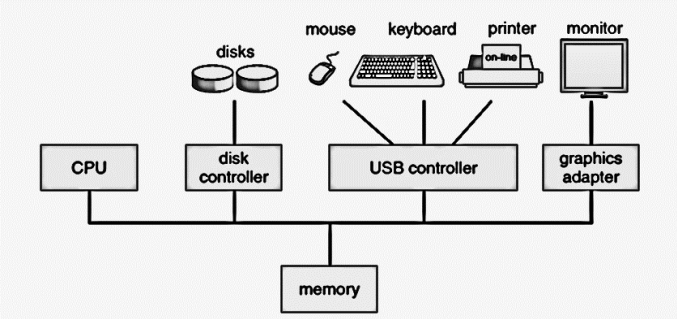


Figure 2- A Modern Computer System.

To accomplish this goal, the bootstrap program must locate and load into memory the operating-system kernel. The operating system then starts executing the first process, such as "initialize," and waits for some event to occur. The occurrence of an event is usually signaled by an interrupt from either the hardware or the software. Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus. Software may trigger an interrupt by executing a special operation called a system call (also called a monitor call).

DESCRIBE THE GENERAL OPERATIONS OF COMPUTER

Computers perform a wide range of operations, but at a high level, the general operations of a computer can be broken down into four main stages: input, processing, output, and storage.

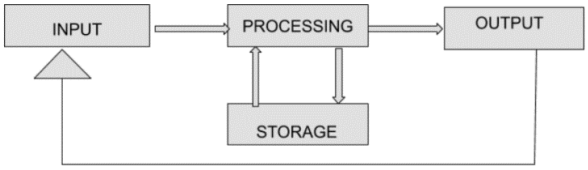


Figure 3 General Purpose Operations in Computer System

Input: This is the stage where the computer receives information or data from an external source, such as a keyboard, mouse, scanner, or microphone. The input is usually in the form of binary digits (bits) that the computer can process.

Processing: In this stage, the computer performs arithmetic, logical, and other operations on the input data using its central processing unit (CPU), which is the "brain" of the computer. The CPU fetches instructions from memory and executes them to manipulate the input data according to the program or algorithm being run.

Output: Once the processing is complete, the computer produces output in a format that can be understood by humans. This output can be displayed on a screen, printed on paper, or converted into sound or other forms of communication.

Storage: The computer also has the ability to store data and information for later use. This data can be stored temporarily in RAM (random access memory) or more permanently on a hard drive or other types of storage media, such as USB drives or cloud storage.

These four stages of input, processing, output, and storage are the basic building blocks of all computer operations.

DESCRIBE INTERRUPTS

An interrupt is a signal sent to the CPU by an external device, such as a keyboard or mouse, to temporarily halt the normal processing of a program and redirect the CPU's attention to a specific task or event. Interrupts are an important feature of modern computer systems and are used to handle a

variety of tasks, including handling input/output operations, managing system resources, and responding to user input.

When an interrupt occurs, the CPU immediately stops executing the current instruction and saves its current state, including the program counter (the address of the next instruction to be executed), processor status, and any other relevant registers or data. The CPU then switches to an interrupt handler routine, which is a program that is specifically designed to handle the interrupt. The interrupt handler routine typically performs some specific action related to the interrupt, such as reading data from an input device, writing data to an output device, or updating a system resource.

IDENTIFY DIFFERENT I/O DEVICES

Input/output (I/O) devices are hardware components that allow a computer to communicate with the outside world, either by receiving data or transmitting data. Some common examples of I/O devices include:

Keyboard: A keyboard is an input device that allows a user to input alphanumeric characters and other commands into a computer.

Mouse: A mouse is an input device that allows a user to control the movement of a cursor on a computer screen and select items by clicking buttons.

Monitor: A monitor is an output device that displays images and text on a computer screen.

Printer: A printer is an output device that prints text and images on paper or other media.

Scanner: A scanner is an input device that converts printed documents or images into digital format for use by a computer.

Speakers: Speakers are output devices that produce sound and allow a computer to play music, videos, and other audio content.

Microphone: A microphone is an input device that allows a computer to record audio.

Webcam: A webcam is an input device that captures video and allows a computer to transmit video content over the internet.

Joystick: A joystick is an input device that allows a user to control the movement of objects in computer games or simulations.

Touchscreen: A touchscreen is an input/output device that allows a user to input commands and receive feedback by touching a display screen.

These are just a few examples of the many different types of I/O devices that can be used with a computer. The specific I/O devices used will depend on the intended purpose of the computer and the needs of the user.

1.2. I/O STRUCTURE.

A general-purpose computer system consists of a CPU and a number of device controllers that are connected through a common bus. Each device controller is in charge of a specific type of device. Depending on the controller, there may be more than one attached device. For instance, the Small Computer Systems Interface (SCSI) controller found on many small to medium sized computers, can have seven or more devices attached to it. A device controller maintains some local buffer storage and a set of special purpose registers. The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage. The size of the local buffer within a device controller varies from one controller to another, depending on the particular device being controlled. For example, the size of the buffer of a disk controller is the same as or a multiple of the size of the smallest addressable portion of a disk, called a sector, which is usually 512 bytes.

The I/O (input/output) structure in an operating system is responsible for managing the flow of data between the CPU and the external devices

connected to the computer. The I/O structure in an operating system consists of several components, including:

Device Drivers: Device drivers are software programs that enable the operating system to communicate with hardware devices. Device drivers translate requests from the operating system into commands that the hardware device can understand, and they also manage device interrupts and input/output operations.

I/O Scheduler: The I/O scheduler is a software component that determines the order in which I/O requests are serviced by the system. The scheduler optimizes the use of system resources by grouping I/O requests together and scheduling them to minimize the movement of disk heads.

Interrupt Handler: The interrupt handler is a software routine that handles interrupts generated by hardware devices. When a hardware device needs attention, it generates an interrupt, which causes the CPU to stop executing its current task and start processing the interrupt. The interrupt handler manages the processing of the interrupt, which may involve reading or writing data to the device.

I/O Control: The I/O control component is responsible for managing the access to I/O devices. It ensures that only authorized users and processes can access the devices and that they use them correctly.

The I/O structure is an operating system in critical to the performance and reliability of the system. The components work together to ensure that data is transferred between the CPU and external devices efficiently, reliably, and securely.

DESCRIBE THE STRUCTURE OF INPUT/OUTPUT SYSTEM

The input/output (I/O) system of a computer consists of several components that work together to facilitate the communication between the computer and its peripherals, such as keyboards, printers, and storage devices. The structure of the I/O system can be described as follows:

Device driver: A device driver is a software component that communicates with a specific hardware device and provides a standardized interface for other software components to interact with the device. Each device in the system has its own device driver, which handles all the low-level details of the device's operation, such as reading and writing data and responding to interrupts.

Interrupt handler: An interrupt handler is a software routine that is executed when an interrupt signal is received from a hardware device. The interrupt handler is responsible for handling the interrupt, communicating with the device driver, and initiating any necessary actions, such as reading data from the device or writing data to a buffer.

I/O controller: An I/O controller is a hardware component that manages the flow of data between the computer and its peripherals. The I/O controller may contain one or more ports, which allow devices to be connected to the computer. The I/O controller communicates with the device driver to send and receive data, and may also provide additional features such as error checking and data compression.

Bus: A bus is a physical pathway that connects the different components of the computer system, including the CPU, memory, and I/O devices. The bus provides a standardized protocol for transferring data between components, and may use different types of interfaces, such as USB, PCI, or SATA.

User interface: The user interface is the part of the system that allows the user to interact with the computer and its peripherals. The user interface may take many forms, including graphical user interfaces (GUIs), command-line interfaces, and application programming interfaces (APIs).

These components work together to create a flexible I/O system that allows the computer to communicate with a wide range of devices and respond to a variety of input and output events. By providing a standardized interface for device communication, the I/O system simplifies the development of software applications and ensures compatibility between different hardware devices.

INTERRUPTS

An interrupt is a signal sent to the CPU by an external device, such as a keyboard or mouse, to temporarily halt the normal processing of a program and redirect the CPU's attention to a specific task or event. Interrupts are an important feature of modern computer systems and are used to handle a variety of tasks, including handling input/output operations, managing system resources, and responding to user input.

When an interrupt occurs, the CPU immediately stops executing the current instruction and saves its current state, including the program counter (the address of the next instruction to be executed), processor status, and any other relevant registers or data. The CPU then switches to an interrupt handler routine, which is a program that is specifically designed to handle the interrupt. The interrupt handler routine typically performs some specific action related to the interrupt, such as reading data from an input device, writing data to an output device, or updating a system resource.

There are many different types of events that may trigger an interrupt for example, the completion of an I/O operation, "division by zero, invalid memory access, and a request for some operating-system service. For each such interrupt, a service routine is provided that is responsible for dealing with the interrupt. When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. The fixed location usually contains the starting address where the service routine for the interrupt is located.

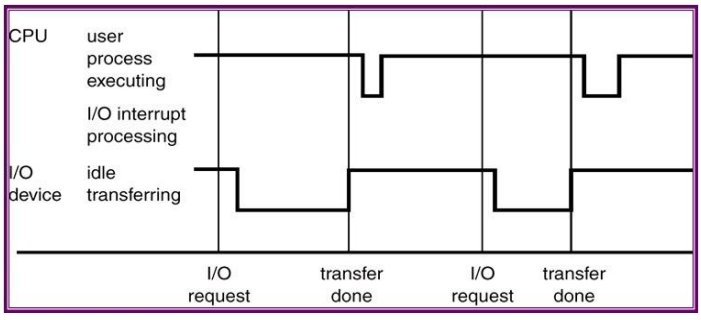


Figure 3. Interrupt time line for a single process doing output.

The interrupt service routine executes; on completion, the CPU resumes the interrupted computation. A time line of this operation is shown in Figure 3.

Hardware Interrupts: Hardware interrupts are generated by the hardware devices connected to the computer. These interrupts are used to signal the CPU that the device requires attention or that an error has occurred.

Software Interrupts: Software interrupts are generated by software applications running on the computer. These interrupts are used to request **services** from the operating system, such as I/O operations or memory management.

Interrupt handling

Once the interrupt handler routine is complete, the CPU restores its saved state and returns to executing the original program at the point where it was interrupted. The ability to handle interrupts enables a computer system to be highly responsive to external events and allows it to perform multiple tasks simultaneously, or in parallel, which is essential for modern multitasking operating systems. There are two common ways in which buses implement interrupts vectored and polling.

Vectored

Devices that use vectored interrupts are assigned an interrupt vector. This is a number that identifies a particular interrupt handler. This vector may be fixed, configurable (using jumpers or switches), or programmable. In the case of programmable devices, an interrupt device cookie is used to program the device interrupt vector. When the interrupt handler is registered, the kernel saves the vector in a table. When the device interrupts, the system enters the interrupt **acknowledge cycle**, asking the interrupting device to identify itself. The device responds with its interrupt vector. The kernel then uses this vector to find the responsible interrupt handler.

Polling

Polling is the process where the computer or controlling device waits for an external device to check for its readiness or state, often which low level hardware. for example when a printer is connected via parallel port the computer waits until the printer has receive the next character. These processes can be as mints as only reading one bit.

Interrupts are an important part of a computer architecture. Each computer design has its own interrupt mechanism, but several functions are common. The interrupt must transfer control to the appropriate interrupt service routine. The straight forward method for handling this transfer would be to invoke a generic routine to examine the interrupt information the routine, in turn, would call the interrupt-specific handler. However, interrupts must be handled quickly, and, given that there is a predefined number of possible interrupts, a table of pointers to interrupt routines can be used instead. The interrupt routine is then called indirectly through the table, with no intermediate routine needed.

1.2.1. I/O INTERRUPTS

In an operating system, I/O interrupts are signals generated by external devices to the CPU to indicate that they require attention or have completed an operation. When an I/O interrupt occurs, the CPU interrupts its current task and switches to an interrupt handling routine to service the interrupt. The interrupt handling routine reads or writes data to the I/O device and then resumes the interrupted task.

I/O interrupts can be either synchronous or asynchronous. Synchronous interrupts occur in response to a specific event, such as a user input. Asynchronous interrupts occur at unpredictable times, such as when a data transfer to or from an I/O device is complete.

The efficient handling of I/O interrupts is critical to the performance and reliability of an operating system. By allowing external devices to signal the CPU for attention, they enable the system to handle multiple tasks

simultaneously and respond quickly to events as they occur. The operating system must manage I/O interrupts efficiently to ensure the smooth operation of the system.

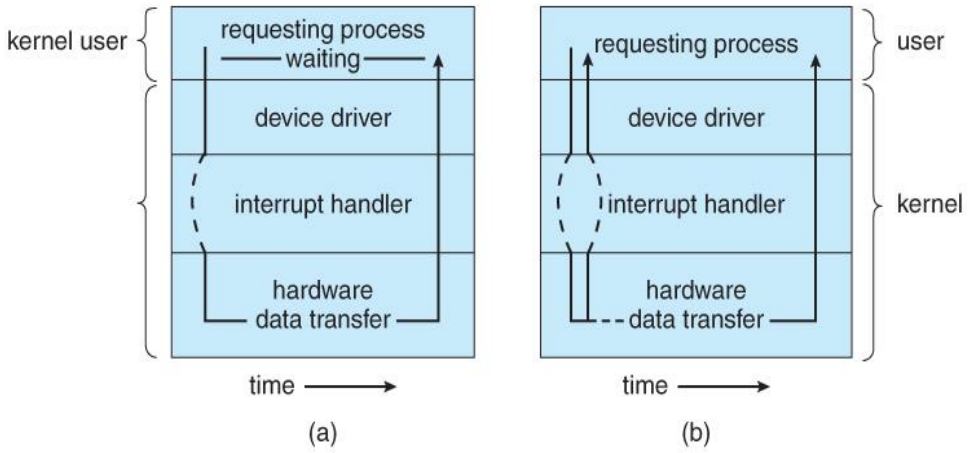


Figure 4 Two I/O methods: (a) synchronous, and (b) asynchronous.

1.2.2. Direct Memory Access (DMA)

DMA (Direct Memory Access) in operating systems is a mechanism that allows input/output (I/O) devices to directly access system memory without the involvement of the CPU. DMA can significantly improve the performance of I/O operations, as it allows I/O devices to transfer data to or from memory without requiring the CPU to be involved in each individual data transfer.

When an I/O device needs to transfer data to or from memory, it sends a DMA request to the DMA controller in the system. The DMA controller then takes over control of the memory bus and transfers the data between the I/O device and memory directly, without involving the CPU. DMA requires careful management and protection to ensure the security and stability of the operating system. For example, DMA controllers can be used by malicious actors to gain unauthorized access to system memory, so operating systems typically use memory protection mechanisms, such as virtual memory and memory segmentation, to prevent unauthorized access to system memory by DMA controllers.

Another potential issue with DMA is that it can interfere with the scheduling of CPU tasks. While a DMA transfer is taking place, the CPU is unable to access the memory bus, which can lead to delays or interference with other CPU tasks. To prevent this, operating systems typically use interrupt mechanisms to allow the CPU to regain control of the memory bus when necessary.

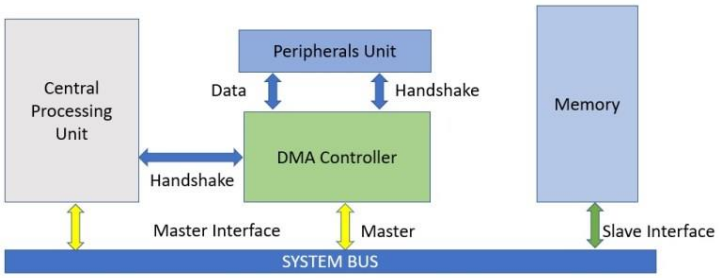


Figure 4 Direct Memory Access

DMA is an important mechanism for improving I/O performance in operating systems, but it requires careful management and protection to ensure the security and stability of the system. The operating system uses a combination of memory protection mechanisms, interrupt mechanisms, and DMA controllers to provide DMA support and ensure that I/O devices can directly access system memory without compromising system security or performance.

1.3. STORAGE STRUCTURE AND HIERARCHY

The programs must be in main memory to. be executed. Main memory is the only large storage area that the processor can access directly. It is an array of words or bytes, ranging in size from hundreds of thousands to hundreds of millions. Each word has its own address. Interaction is achieved through a sequence of load or store instructions to specific memory addresses. The load instruction moves a word from main memory to an internal register within the CPU/whereas the store instruction moves the content of a register to main

memory. Aside from explicit loads and stores, the CPU automatically loads instructions from main memory for execution.

A typical instruction-execution cycle, as executed on a system with a von Neumann architecture, will first fetch an instruction from memory and will store that instruction in the instruction register. The instruction is then decoded and may cause operands to be fetched from memory and stored in some internal register. After the instruction on the operands has been executed, the result may be stored back in memory. Note that the memory unit sees only a stream of memory addresses it does not know how they are generated (the instruction counter, indexing, indirection, literal addresses, and so on) or what they are for (instructions or data). Accordingly, we can ignore how a memory address is generated by a program. We are interested in only the sequence of memory addresses generated by the running program.

Ideally, we would want the programs and data to reside in main memory permanently. This arrangement is not possible for the following two reasons:

1. Main memory is usually too small to store all needed programs and data permanently.
2. Main memory is a volatile storage device that loses its contents when power is turned off or otherwise lost.

Thus, most computer systems provide secondary storage as an extension of main memory. The main requirement for secondary storage is that it be able to hold large quantities of data permanently. The most common secondary-storage device is a magnetic disk, which provides storage of both programs and data. Most programs (web browsers, compilers, word processors, spreadsheets, and so on) are stored on a disk until they are loaded into memory. Many programs then use the disk as both a source and destination of the information for their processing.

DESCRIBE DIFFERENT STORAGE DEVICES

There are several types of storage devices that are commonly used in computer systems, each with their own advantages and disadvantages. Here are some examples:

Hard Disk Drive (HDD): An HDD is a traditional magnetic storage device that uses spinning disks to store data. It has high capacity and relatively low cost, but is slower than newer solid-state storage technologies.

Solid State Drive (SSD): An SSD uses flash memory to store data and has no moving parts, making it faster and more reliable than an HDD. It is also more expensive and typically has lower capacity.

USB Flash Drive: A USB flash drive is a small, portable storage device that uses flash memory to store data. It is easy to use and highly portable, but has lower capacity than larger storage devices.

Memory Card: A memory card is a small, portable storage device that is commonly used in cameras, smartphones, and other mobile devices. It uses flash memory to store data and comes in a variety of sizes and formats.

Optical Disc: An optical disc is a type of storage device that uses laser technology to read and write data on a disc. Examples include CDs, DVDs, and Blu-ray discs. They have high capacity, but are slower than other types of storage and are becoming less commonly used.

Cloud Storage: Cloud storage refers to storing data on remote servers accessed via the internet. It allows for easy access to data from multiple devices and locations, but can have security and privacy concerns.

The choice of storage device depends on the specific needs and use cases of the user. Factors such as capacity, speed, portability, and cost should be considered when selecting a storage device.

1.3.1. MAIN MEMORY

Main memory and the registers built into the processor itself are the only storage that the CPU can access directly. Therefore, any instructions in execution, and any data being used by the instructions, must be in one of these direct-access storage devices. If the data are not in memory, they must be moved there before the CPU can operate on them.

Each I/O controller includes registers to hold commands and the data being transferred. Usually, special I/O instructions allow data transfers between these registers and system memory. To allow more convenient access to I/O devices, many computer architectures provide memory-mapped I/O. In this case, ranges of memory addresses are set aside, and are mapped to the device registers. Reads and writes to these memory addresses cause the data to be transferred to and from the device registers. This method is appropriate for devices with fast response times, such as video controllers. In the IBM PC, each location on the screen is mapped to a memory location. Displaying text on the screen is almost as easy as writing the text into the appropriate memory-mapped locations.

Memory-mapped I/O is also convenient for other devices, such as the serial and parallel ports used to connect modems and printers to a computer. The CPU transfers data through these kinds of devices by reading and writing a few device registers called an I/O port. To send out a long string of bytes through a memory-mapped serial port, the CPU writes one data byte to the data register, then sets a bit in the control register to signal that the byte is available. The device takes the data byte, and then clears the bit in the control register to signal that it is ready for the next byte. Then, the CPU can transfer the next byte. If the CPU uses polling to watch the control bit, constantly looping to see whether the device is ready, this method of operation is called programmed I/O (PIO). If the CPU does not poll the control bit, but instead receives an interrupt when the device is ready for the next byte, the data transfer is said to be interrupt driven.

Registers that are built into the GPU are generally accessible within one cycle of the CPU clock. Most CPUs can decode instructions and perform simple operations on register contents at the rate of one or more operations per clock tick. The same cannot be said for main memory, which is accessed via a transaction on the memory bus. Memory access may take many cycles to complete, in which case the processor normally needs to stall, since it does not have the data required to complete the instruction that it is executing. This situation is intolerable because of the frequency of memory accesses. The remedy is to add fast memory between the CPU and main memory.

1.3.2. MAGNETIC DISKS

Magnetic disks provide the bulk of secondary storage for modern computer systems. Conceptually, disks are relatively simple (Figure 5). Each disk platter has a flat circular shape, like a CD. Common platter diameters range from 1.8 to 5.25 inches. The two surfaces of a platter are covered with a magnetic material, similar to magnetic tape. We store information by recording it magnetically on the platters.

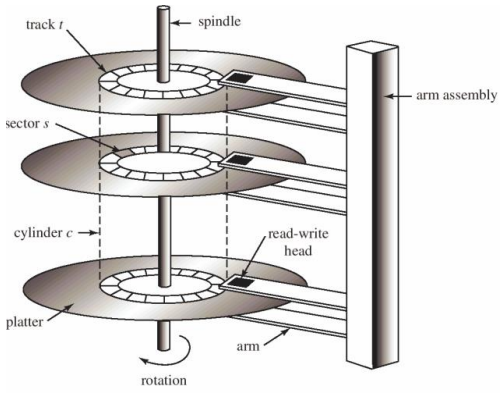


Figure 5. Moving-head disk mechanism.

A read-write head flies just above each surface of every platter. The heads are attached to a disk arm, which moves all the heads as a unit. The surface of a platter is logically divided into circular tracks, which are subdivided info sectors. The set of tracks that are at one arm position forms a cylinder. There may be thousands of concentric cylinders in a disk drive, and each track may

contain hundreds of sectors. The storage capacity of common disk drives is measured in gigabytes. (A kilobyte is 1024 bytes, a megabyte is 1024^2 bytes, and a gigabyte is 1024^3 bytes, but disk manufacturers often round off these numbers and say that a megabyte is 1 million bytes, and a gigabyte is 1 billion bytes.)

When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 150 times per second. Disk speed has two parts. The transfer rate is the rate at which data flow between the drive and the computer. The positioning time, sometimes called the random-access time, consists of the time to move the disk arm to the desired cylinder, called the seek time, and the time for the desired sector to rotate to the disk head, called the rotational latency. Typical disks can transfer several megabytes of data per second, and have seek times and rotational latencies of several milliseconds. Because the disk head flies on an extremely thin cushion of air (measured in microns), there is a danger of the head contacting the disk surface. Although. the disk platters are coated with a thin protective layer, sometimes the head will damage the magnetic surface. This accident is called a head crash. A head crash normally cannot be repaired; the entire disk must be replaced.

A disk can be removable, allowing different disks to be mounted as needed. Removable magnetic disks generally consist of one platter, held in a plastic case to prevent damage while not in the disk drive. Floppy disks are inexpensive removable magnetic disks that have a soft plastic case containing a flexible platter. The head of a floppy-disk drive generally sits directly on the disk surface, so the drive is designed to rotate more slowly than a hard-disk drive, to reduce the wear on the disk surface. The storage capacity of a floppy disk is typically only 1 megabyte or so. Removable disks are available that work much like normal hard disks, and that have capacities measured in gigabytes.

1.3.3. MAGNETIC TAPES

Magnetic tape was used as an early secondary-storage medium. Although it Is relatively permanent and can hold large quantities of data its access time is slow in comparison to that of main memory. Random access to magnetic tape

is about a thousand times slower than random access to magnetic disk, so tapes are not useful for secondary storage. Tapes are used mainly for backup, for storage of infrequently used information, and as a medium for transferring information from one system to another.

A tape is kept in a spool, and is wound or rewound past a read-write head. Moving to the correct spot on a tape can take minutes, but once positioned, tape drives can write data at speeds comparable to disk drives. Tape capacities vary greatly, depending on the particular kind of tape drive. Some tapes hold 20 times more data than a large disk drive. Tapes are categorized by width, including: 4, 8, and 19 millimeter, 1/4 and 1/2 inch.

1.3.4. STORAGE HIERARCHY.

The wide variety of storage systems in a computer system can be organized in a hierarchy (Figure 5) according to speed and their cost. The higher levels are expensive, but are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases. This tradeoff is reasonable; if a given storage system were both faster and less expensive than another other property being the same then there would be no reason to use the slower, more expensive memory. In fact, many early storage devices, including paper tape and core memories, are relegated to museums now that magnetic tape and semiconductor memory have become faster and cheaper.

EXPLAIN THE HIERARCHY OF STORAGE DEVICES

The storage hierarchy refers to the different levels of storage devices and technologies that are used in computer systems, arranged in order of speed, cost, and capacity. The higher levels are expensive, but are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases. This tradeoff is reasonable; if a given storage system were both faster and less expensive than another other property being the same then there would be no reason to use the slower, more expensive memory. Each level of the hierarchy serves a specific purpose and provides different benefits, as described below:

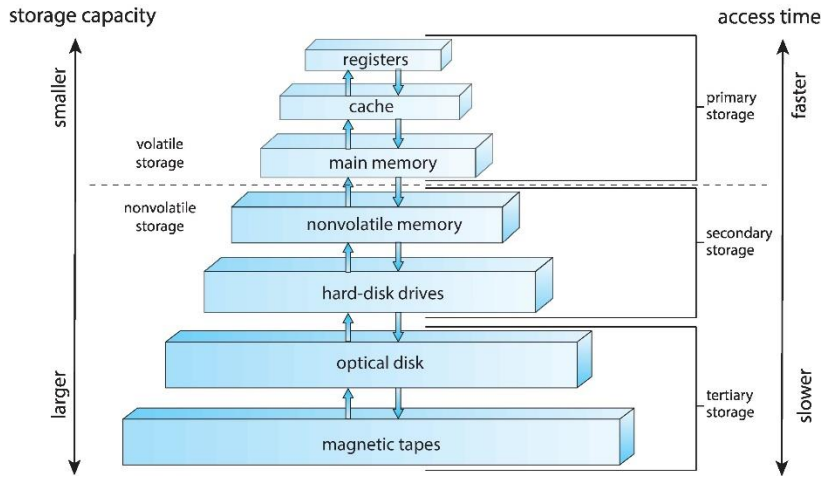


Figure 5 Storage Hierarchy

Registers: Registers are the fastest and smallest type of storage device in a computer system, located within the CPU itself. They are used to store data that is currently being processed by the CPU and are typically measured in bytes.

Cache Memory: Cache memory is a type of fast memory that sits between the CPU and main memory (RAM) in the hierarchy. It is designed to improve performance by reducing the number of times the CPU needs to access slower main memory. There are typically multiple levels of cache memory, with each level being larger and slower than the previous level.

Main Memory (RAM): Main memory, or RAM, is the primary storage area used by the CPU for running programs and storing data. It is faster than secondary storage devices such as hard drives, but slower than cache memory. RAM is volatile, meaning that it loses its contents when power is turned off.

Solid State Drives (SSD): Solid state drives are non-volatile storage devices that use flash memory to store data. They are faster and more reliable than hard disk drives, but are typically more expensive and have lower capacity.

Hard Disk Drives (HDD): Hard disk drives are magnetic storage devices that use spinning disks to store data. They are slower than SSDs, but have higher capacity and are typically less expensive.

Optical Discs: Optical discs, such as CDs, DVDs, and Blu-ray discs, are low-cost storage devices that can be used for archival or backup purposes. They are slower than hard drives or SSDs, but have high capacity and are easily portable.

Tape Drives: Tape drives are used for backup and archival storage and offer very high capacity at a low cost. They are slow and not suitable for frequent access or retrieval of data.

The storage hierarchy is designed to provide a balance between speed, cost, and capacity, with faster and more expensive technologies used for frequently accessed data and slower and cheaper technologies used for archival or backup storage.

When external power is restored, the controller copies the data back into the RAM. The design of a complete memory system must balance all these factors: It uses only as much expensive memory as necessary, while providing as much inexpensive, non-volatile memory as possible. Caches can be installed to ameliorate performance differences where there is a large access-time or transfer-rate disparity between two components.

1.4. HARDWARE PROTECTION

Hardware protection refers to the mechanisms implemented in an operating system to ensure that user processes cannot access or modify critical system resources, such as memory, disk drives, or other hardware devices, without proper authorization.

There are several techniques used in modern operating systems to provide hardware protection. One of the most fundamental is memory protection,

which involves dividing the system's memory into different regions and assigning access permissions to each region. For example, the operating system may allocate memory for the user process and prevent it from accessing the memory used by other processes or the kernel.

Another technique is the use of virtual memory, which allows the operating system to provide each process with its own virtual address space, while mapping these addresses to the physical memory in a controlled manner. This technique enables the operating system to isolate processes from each other and protect them from accessing each other's memory.

In addition to memory protection, the operating system also uses hardware protection mechanisms to control access to other system resources, such as disk drives or input/output devices. For example, the operating system may use access control lists (ACLs) to specify which users or processes are allowed to access a particular file or device.

The hardware protection is a crucial component of modern operating systems, as it helps ensure the security and stability of the system and prevents unauthorized access or modification of critical system resources.

1.1.1. DUAL-MODE OPERATION

Dual mode operation in operating systems is a feature that allows the system to differentiate between user mode and kernel mode. In this mode, the operating system works in two different levels of privileges or modes: user mode and kernel mode.

1.1.1.1. USER MODE

The OS mode in which all the user applications and programs will run. Here, the user instructions are worked on and Software’s like playing music is run.

User mode is the mode in which most application programs run. In this mode, the CPU has limited access to system resources, and programs cannot access critical system resources directly, such as hardware devices, system memory,

or CPU instructions. i.e., The mode bit is set to 1 in the user mode. It is changed from 1 to 0 when switching from user mode to kernel mode.

1.1.1.2. KERNEL MODE

Kernel mode is the mode in which the operating system and device drivers run. In this mode, the CPU has unrestricted access to system resources, and the operating system and device drivers can execute privileged instructions, access system memory and devices, and perform other privileged operations.

The OS mode in which the hardware loads and its computations are performed. Only privileged instructions are allowed to run in kernel mode. Some common privileged instructions are

- Input-Output Management
- Switching modes between user mode and kernel mode.
- Interrupt management

Dual mode operation provides a mechanism for protecting the operating system and its resources from user-level applications. The operating system uses hardware protection mechanisms, such as memory management units (MMUs) and privileged instructions, to ensure that only privileged code can access system resources in kernel mode.

The transition between user mode and kernel mode is usually triggered by system calls, interrupts, and exceptions. When a user-level program requests a system call, for example, the operating system switches to kernel mode to execute the system call, and then returns to user mode when the system call is completed. dual mode operation is a crucial feature of modern operating systems, as it allows the operating system to provide a secure and stable environment for executing user-level applications while protecting system resources from unauthorized access or modification.

1.1.2. CPU PROTECTION:

CPU protection in operating systems refers to the mechanisms that the operating system uses to ensure that the CPU is used efficiently and fairly by all running processes. One of the most important mechanisms for CPU protection is scheduling, which is the process of deciding which process should run on the CPU at a given time. The operating system uses different scheduling algorithms to allocate CPU time to processes based on their priority, their resource usage, and other factors.

Another mechanism for CPU protection is process isolation, which is the process of separating processes from each other to prevent one process from interfering with another. The operating system uses memory protection mechanisms, such as virtual memory and address translation, to isolate processes from each other and ensure that each process has its own virtual address space.

Buffering

A buffer is an area of main memory for holding data during input and output data transfers. A buffer contains data that is stored for a short amount of time typically in RAM. In short, A Temporary storage area, Usually in RAM.

A buffer contains data that is stored for a short amount of time, typically in the computer's memory (RAM). The purpose of a buffer is to hold data right before it is used. For example, when you download an audio or video file from the Internet, it may load the first 20% of it into a buffer and then begin to play. While the clip plays back, the computer continually downloads the rest of the clip and stores it in the buffer. Because the clip is being played from the buffer, not directly from the Internet, there is less of a chance that the audio or video will stall or skip when there is network congestion.

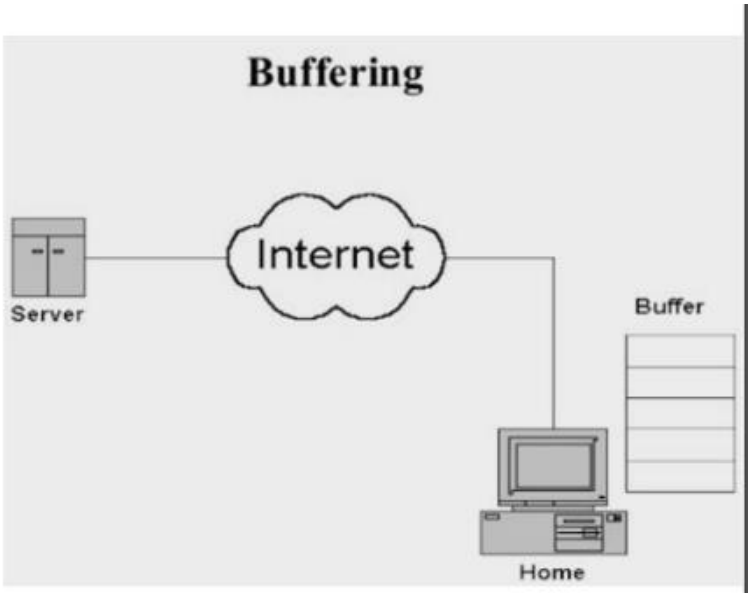


Figure 6 Buffering in OS

Buffering is used to improve several other areas of computer performance as well. Most hard disks use a buffer to enable more efficient access to the data on the disk. Video cards send images to a buffer before they are displayed on the screen (known as a screen buffer). Computer programs use buffers to store data while they are running. If it were not for buffers, computers would run a lot less efficiently and we would be waiting around a lot more.

Spooling

Spool: Simultaneous peripheral operation on line. A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. Spooling overlaps input of one job with the computation of other jobs.

The operating system also uses hardware protection mechanisms, such as interrupts and traps, to ensure that processes cannot execute privileged CPU instructions or access critical system resources without proper authorization.

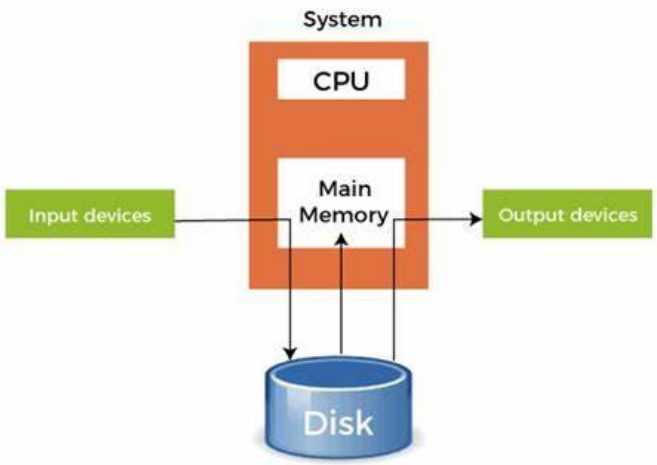


Figure 7 Spooling

CPU protection is essential for the efficient and fair use of the CPU by all running processes, as well as for ensuring the security and stability of the operating system. The operating system uses a combination of scheduling algorithms, process isolation, and hardware protection mechanisms to provide CPU protection and ensure that the CPU is used effectively and fairly by all running processes.

1.1.3. MEMORY PROTECTION:

In memory protection, we are talking about that situation when two or more processes are in memory and one process may access the other process memory. and

Memory protection in operating systems refers to the mechanisms that the operating system uses to prevent one process from accessing or modifying the memory used by another process without proper authorization. Memory protection is essential for the security and stability of the operating system, as it helps prevent processes from interfering with each other and from accessing critical system resources. and to prevent this situation we are using two registers as:

1. Base register

2. Limit register

basically, Base register store the starting address of program and limit register store the size of the process, so when a process wants to access the memory then it is checked that it can access or can not access the memory.

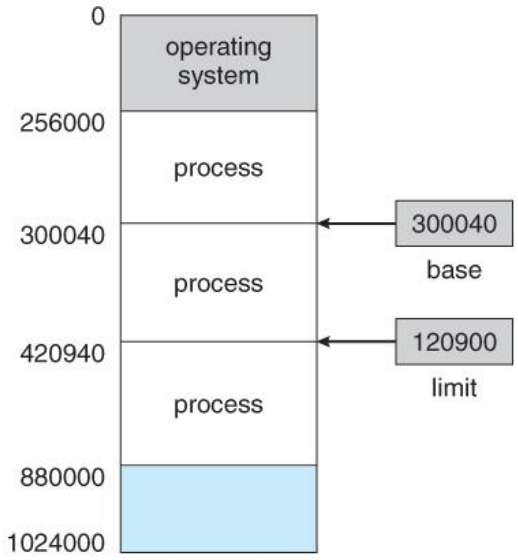


Figure 7. A base and a limit register define a logical address space

What we need to separate each program's memory space is an ability to determine the range of legal addresses that the program may access, and to protect the memory outside that space. We can provide this protection by using two registers, usually a base and a limit, as illustrated in Figure 7. The base register holds the smallest legal physical memory address; the limit register contains the size of the range. For example, if the base register holds 300040 and limit register is 120900, then the program can legally access all addresses from 300040 through 420940 inclusive.

This protection is accomplished by the CPU hardware comparing every address generated in user mode with the registers. Any attempt by a program executing in user mode to access monitor memory or other users' memory results in a trap to the monitor, which treats the attempt as a fatal error (Figure 8). This scheme prevents the user program from (accidentally or deliberately) modifying the code or data structures of either the operating system or other users. The base and limit registers can be loaded by only the operating system, which uses a special privileged instruction. Since privileged instructions can be executed in only monitor mode, and since only the operating system executes in monitor mode, only the operating system

can load the base and limit registers. This scheme allows the monitor to change the value of the registers but prevents user programs from changing the registers' contents.

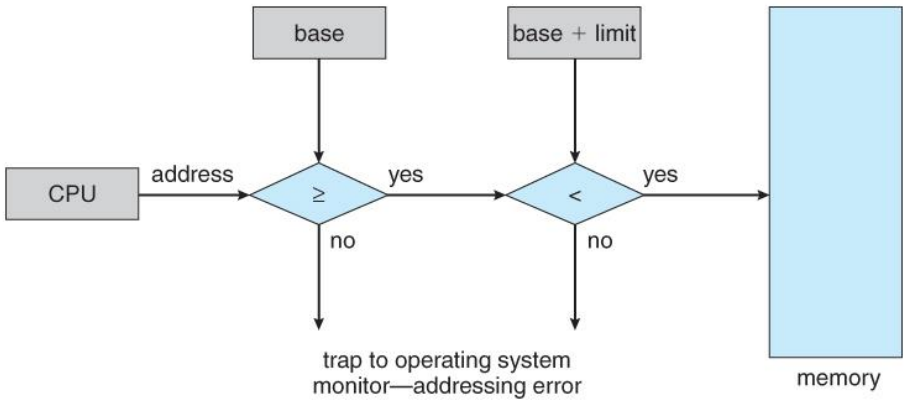


Figure 8. Hardware address protection with base and limit registers

The operating system, executing in monitor mode, is given unrestricted access to both monitor and users' memory. This provision allows the operating system to load users' programs into users' memory, to dump out those programs in case of errors, to access and modify parameters of system calls, and so on.

1.1.4. I/O PROTECTION:

I/O (Input/Output) protection in operating systems refers to the mechanisms that the operating system uses to prevent one process from accessing or modifying I/O devices used by another process without proper authorization. I/O protection is essential for the security and stability of the operating system, as it helps prevent processes from interfering with each other and from accessing critical I/O resources. One of the primary mechanisms for I/O protection is the use of access control lists (ACLs), which specify which users or processes are allowed to access a particular I/O device or file. The operating system uses ACLs to restrict access to critical I/O resources, such as network devices, disk drives, and printers.

The operating system also uses device drivers to manage I/O devices and to provide a standard interface for applications to access I/O devices. Device

drivers act as an intermediary between the operating system and the I/O devices, and they use hardware protection mechanisms, such as interrupts and DMA (Direct Memory Access), to ensure that processes can only access I/O devices that they are authorized to use.

Another mechanism for I/O protection is I/O scheduling, which is the process of deciding which process should have access to an I/O device at a given time. The operating system uses different I/O scheduling algorithms to allocate I/O resources to processes based on their priority, their resource usage, and other factors.

The user program could gain control of the computer in monitor mode. when we're ensuring the I/O protection then some cases will never have occurred in the system as:

1. Termination I/O of other process
2. View I/O of other process
3. Giving priority to a particular process I/O

IDENTIFY THE PROTECTION ISSUES ENCOUNTERED IN A COMPUTER SYSTEM

Computer systems face various protection issues that can compromise their security and integrity. Some of the key protection issues encountered in a computer system include:

Unauthorized Access: Unauthorized access occurs when an individual gains access to a computer system without proper authorization or authentication. This can lead to theft of sensitive data, tampering with critical system files, and other security breaches.

Malware: Malware refers to any software that is designed to harm or disrupt computer systems. Examples include viruses, worms, and Trojan horses. Malware can damage files, steal data, and create security vulnerabilities in the system.

Denial of Service (DoS) attacks: A DoS attack occurs when a system is overwhelmed with traffic or requests, causing it to crash or become unavailable. This type of attack is commonly used to disrupt the operations of a website or online service.

Data Breaches: Data breaches occur when sensitive information is stolen or compromised. This can occur due to weak passwords, unsecured networks, or other vulnerabilities in the system.

Phishing: Phishing is a type of social engineering attack in which an attacker attempts to trick individuals into revealing sensitive information, such as login credentials or credit card numbers. Phishing attacks can occur via email, phone calls, or text messages.

Insider Threats: Insider threats occur when individuals with authorized access to a system abuse their privileges for malicious purposes. This can include stealing sensitive data or tampering with critical system files.

Overall, protection issues in a computer system can arise from a variety of sources, including human error, software vulnerabilities, and malicious attacks. Implementing strong security protocols and regularly updating software and systems can help mitigate these risks and protect against potential threats.

DESCRIBE PROTECTION METHODS OF COMPUTER OPERATION

When several users share computer resources such as CPU, memory, and other resources, security is more crucial. It is the job of the operating system to provide a mechanism that protects each process from other processes. All assets that require protection in a multiuser environment are categorized as objects, and individuals who seek to access these things are referred to as subjects. Distinct 'access privileges are granted to different subjects by the operating system.

Protection is a method that limits the access of programs, processes, or users to the resources defined by a computer system. Protection can be used to allow several users to safely share a common logical namespace, such as a

directory or files, in multi-programming operating systems. It necessitates the safeguarding of computer resources such as software, memory, and processors. As assistance to multiprogramming OS, users should apply protective steps so that several users can safely access a common logical namespace like a directory or data. Maintaining secrecy, honesty, and availability in the OS might provide protection. The device must be protected against unauthorized access, viruses, worms, and other malware.

Operating systems provide a range of built-in protection methods to help ensure the security and integrity of computer operations. Some of the most common protection methods in operating systems include:

User Accounts and Permissions: Most operating systems provide user accounts and permissions to control access to system resources. User accounts can be assigned specific permissions to control access to files, directories, and other resources.

Access Controls: Operating systems can also provide access controls to limit access to critical system resources. These controls can include role-based access controls, discretionary access controls, and mandatory access controls.

Encryption: Operating systems can also provide encryption for data at rest and in transit to protect sensitive data from unauthorized access. Encryption can be used to protect files, directories, and other system resources.

Firewalls: Operating systems can provide built-in firewalls to control incoming and outgoing network traffic. Firewalls can be used to block unauthorized network traffic and prevent malware and other malicious software from accessing the system.

Anti-Malware and Antivirus Software: Most operating systems provide built-in anti-malware and antivirus software to detect and remove malicious software from the system. This software can be configured to scan files, emails, and other system resources for malware and viruses.

Regular Software and System Updates: Operating system updates can provide critical patches and security updates to address vulnerabilities and security flaws in the system. Regularly updating the operating system can help ensure that the system is protected against the latest security threats.

The protection methods for computer operation in operating systems require a combination of technical measures and regular maintenance and updates. By implementing these measures, organizations can enhance the security and integrity of their computer systems and reduce the risk of security breaches and data loss.

Multiple Choice Questions

Q.1: What is an operating system?

(a) interface between the hardware and application programs

(b) collection of programs that manages hardware resources

(c) system service provider to the application programs

(d) all of the mentioned

Q.2: Where is the operating system placed in the memory?

(a) either low or high memory (depending on the location of interrupt vector) (b) in the low memory

(c) in the high memory (d) none of the mentioned

Q.3: If a process fails, most operating system write the error information to a _____

(a) new file (b) another running process

(c) log file (c) none of the mentioned

Q.4: In operating system, each process has its own _____

(a) open files (b) address space and global variables

(c) pending alarms, signals, and signal handlers

(d) All of above

Q.5: The main memory accommodates _____

(a) CPU (b) user processes

(c) operating system (c) all of the mentioned

Q.6: The operating system is responsible for_____

(a) bad-block recovery (b) booting from disk

(c) disk initialization (d) all of the mentioned

Q.7: The operating system and the other processes are protected from being modified by an already running process because _____

(a) every address generated by the CPU is being checked against the relocation and limit registers

(b) they have a protection algorithm

(c) they are in different memory spaces

(d) they are in different logical addresses

Q.8: The operating system maintains a _____ table that keeps track of how many frames have been allocated, how many are there, and how many are available.

(a) memory (b) mapping

(c) page (d) frame

Q.9: In real time operating system _____.

(a) process scheduling can be done only once

(b) all processes have the same priority

(c) kernel is not required

(d) a task must be serviced by its deadline period

Q.10: The operating system keeps a small table containing information about all open files called _____

(a) file table (b) directory table

(c) open-file table (d) system table

Q.11: Whenever a process needs I/O to or from a disk it issues a _____

(a) system call to the operating system (b) a special procedure

(c) system call to the CPU (d) all of the mentioned

Q.12: The two steps the operating system takes to use a disk to hold its files are _____ and _____

(a) Caching & logical formatting (b) Logical formatting & swap space creation

(c) Swap space creation & caching (d) Partitioning & logical formatting

Q.13: The _____ program initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system.

(a) Bootstrap (b) Main

(c) Bootloader (d) Rom

Q.14: Which principle states that programs, users, and even the systems be given just enough privileges to perform their task?

(a) Principle of least privilege (b) Principle of process scheduling

(c) Principle of operating system (d) none of the mentioned

Q.15: Which one of the following errors will be handle by the operating system?

(a) power failure (b) lack of paper in printer
(c) connection failure in the network (d) all of the mentioned.

ANSWER KEY

Q.1 (d)	Q.2 (a)	Q.3 (c)	Q.4 (d)	Q.5 (c)
Q.6 (d)	Q.7 (a)	Q.8 (d)	Q.9 (d)	Q.10 (c)
Q.11 (a)	Q.12 (d)	Q.13 (a)	Q.14 (a)	Q.15 (d)

Short Questions

1. What is an Operating System?
2. Describe the general operations of a computer?
3. Draw a diagram of modern computer system?
4. Describe interrupts
5. Define I/O Interrupts?
6. Describe the structure of input/output system?
7. Define main memory?
8. Describe different storage devices?
9. What are two mode operation?
10. What is the Dual Mode Operation?
11. Describe CPU Protection?
12. Describe Memory protection?
13. Describe the function of I/O Protection?
14. What is DMA and also describe its functionality?
15. Describe protection methods of computer operation?

Long Questions

1. What is an operating system and also draw its abstract view of components of a computer system?
2. Explain the computer system operations?
3. What is storage hierarchy and also draw its diagram?
4. What is hardware protection and also explain its types?

Bibliography

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar

CHAPTER 02 OPERATING SYSTEM STRUCTURE

Objectives

After completion of this chapter students will be able to:

An operating system provides the environment within which programs are executed. Internally, operating systems vary greatly in their makeup, being organized along many different lines. The design of a new operating system is a major task. It is important that the goals of the system be well defined before the design begins. The type of system desired is the foundation for choices among various algorithms and strategies that will be necessary.

An operating system (OS) is a software program that manages computer hardware and software resources and provides common services for computer programs.

1.1. SYSTEM COMPONENTS

We can create a system as large and complex as an operating system only by partitioning it into smaller pieces. Each of these pieces should be a well-defined portion of the system, with carefully defined inputs, outputs, and function. Obviously, not all systems have the same structure. However, many modern systems share the goal of supporting the types of system components.

LIST FUNCTIONS OF AN OPERATING SYSTEM

1. Process management
2. Memory management
3. File management
4. Device management
5. Security management
6. Networking
7. User interface
8. Error detection and handling
9. Resource allocation

DESCRIBE THE FUNCTIONS OF AN OPERATING SYSTEM

An operating system (OS) is a software program that manages computer hardware and software resources and provides common services for computer programs. Here are some of the key functions performed by an operating system:

Process management: The OS manages the processes running on a computer, including the creation, scheduling, and termination of processes.

Memory management: The OS manages the allocation and deallocation of memory for processes, and ensures that processes can access the memory they need without interfering with each other.

File management: The OS manages the creation, modification, deletion, and organization of files on disk storage, and provides a common interface for programs to access files.

Device management: The OS manages the interaction between software programs and hardware devices, including input/output (I/O) operations and communication with peripheral devices such as printers and scanners.

Security management: The OS provides various security mechanisms to protect the system from unauthorized access, including access control, authentication, and encryption.

Networking: The OS provides networking support, allowing computers to communicate with each other over a network, and providing access to network resources such as files and printers.

User interface: The OS provides a graphical or command-line interface that allows users to interact with the computer and run programs.

Error detection and handling: The OS detects and handles errors that occur during system operation, such as hardware failures, software errors, and security breaches.

Resource allocation: The OS manages the allocation of resources, including CPU time, memory, and I/O bandwidth, to ensure that each process has access to the resources it needs.

The operating system plays a vital role in managing and coordinating the various components of a computer system, providing a common interface for programs to access resources, and ensuring that the system runs smoothly and securely.

EXPLAIN MEMORY MANAGEMENT TECHNIQUES.

1. Process management
2. Memory management
3. Storage management

4. File system management

2.1.1. PROCESS MANAGEMENT

Process management is a vital part of an operating system (OS) that is responsible for managing the execution of processes. A process is a program in execution that has its own memory space and other resources assigned to it by the OS. The following are some of the key tasks that the OS performs in process management:

What is Process?

A process is a program in execution. A process needs some certain resources, including CPU time, memory, files and I/O devices to accomplish its task. OR A process is defined as an entity which represents the basic unit of work to be implemented in the system. Program is a passive entity and process is an active entity.

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

Components of process are following.

Sr. No	Component & Description
1	**Object Program** Code to be executed.
2	**Data** Data to be used for executing the program.
3	**Resources** While executing the program, it may require some resources.
4	**Status** Verifies the status of the process execution. A process can run to completion only when all requested resources have been allocated to the process. Two or more processes could be executing the same program, each using their own data and resources.

A process includes:

- **Program counter:** Current activity including, processor register.
- **Stack:** containing temporary data
- **Data section:** containing global variables
- **Text section:** The program code, also called
- **Heap:** containing memory dynamically allocated during run time

Process states

As a process executes, it changes state. The state of a process is defined as the current activity of the process. Process can have one of the following five states at a time.

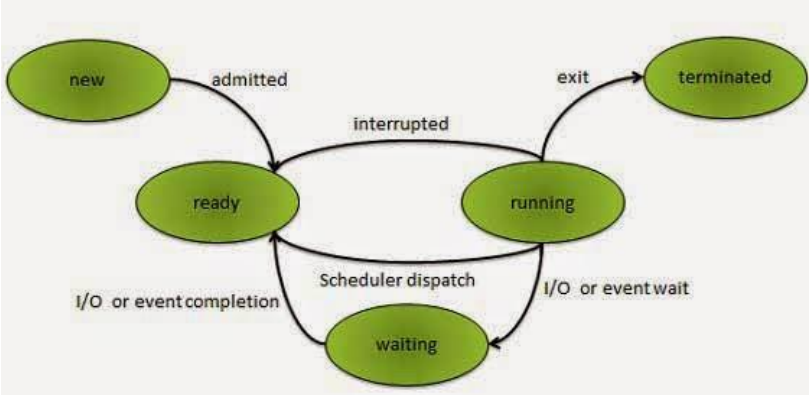


Figure 8 Process States

Sr. No	Component & Description
1	**New** The process is being created.
2	**Ready** The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.
3	**Running** Process instructions are being executed (i.e. The process that is currently being executed).

4	**Waiting** The process is waiting for some event to occur (such as the completion of an I/O operation).
5	**Terminated** The process has finished execution

Process Control Block (PCB)

Sr. No	Component & Description
1	**Pointer** Pointer points to another process control block. Pointer is used for maintaining the scheduling list.
2	**Process State** Process state may be new, ready, running, waiting and so on.
3	**Program Counter** Program Counter indicates the address of the next instruction to be executed for this process.
4	**CPU registers** CPU registers include general purpose register, stack pointers, index registers and accumulators etc. number of register and type of register totally depends upon the computer architecture.
5	**Memory management information** This information may include the value of base and limit registers, the page tables, or the segment tables depending on the memory system used by the operating system. This information is useful for de allocating the memory when the process terminates.
6	**Accounting information** This information includes the amount of CPU and real time used, time limits, job or process numbers, account numbers etc.

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

Process Creation: The OS creates a new process when a user initiates a new program or when an existing process creates a child process. The new process is assigned a unique process ID (PID) and allocated resources such as memory, CPU time, and input/output (I/O) devices.

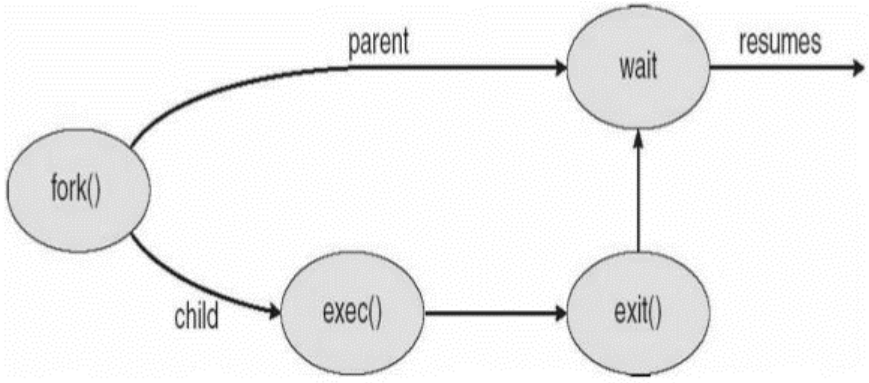


Figure 9 Process Creation

Process Termination: When a process completes its task or terminates abnormally, the OS releases the resources allocated to the process, including memory, CPU time, and I/O devices.

Process Termination: When a process completes its task or terminates abnormally, the OS releases the resources allocated to the process, including memory, CPU time, and I/O devices.

Scheduling Queues Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. This figure 2.3 shows the queuing diagram of process scheduling.

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

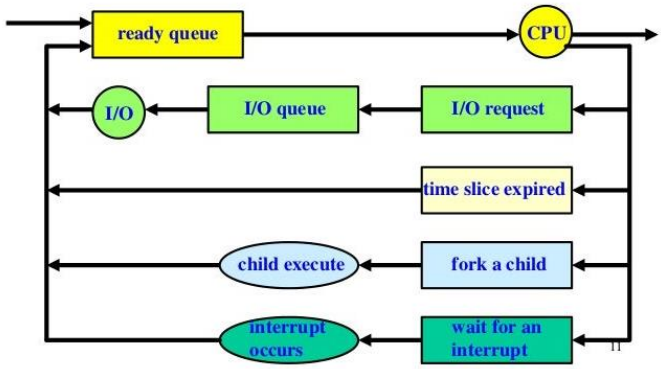


Figure 10 Scheduling Queues

- Job queue: set of all processes in the system ready/waiting.
- Ready queue: set of all processes residing in memory r/w to exe. A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.
- Device queue: device queue is a queue for which multiple processes are waiting for a particular I/O device each device has its own device queue.

Two State Process Model

Two state process models refer to running and non-running states which are described below.

Sr. No	Component & Description
1	**Running** When new process is created by Operating System that process enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process.

Process Communication: Processes can communicate with each other using inter process communication (IPC) mechanisms such as shared memory, pipes, and sockets. These mechanisms allow processes to exchange data and synchronize their activities.

Process Synchronization: When multiple processes share resources such as memory or I/O devices, the OS uses synchronization techniques such as semaphores, monitors, and locks to ensure that the processes access the shared resources in a coordinated manner.

Process Monitoring and Control: The OS monitors the performance of processes and can terminate or suspend them if they violate system policies or consume too many resources. The OS also provides tools for users to monitor and control processes, such as task managers and system monitors.

Effective process management is critical to ensure that the system resources are utilized efficiently and that processes run smoothly without interfering with each other. The OS provides a range of mechanisms and tools to manage processes effectively, allowing multiple processes to run concurrently and improving the overall performance of the system.

- The creation and deletion of both user and system processes.
- The suspension and resumption of processes.
- The provision of mechanisms for process synchronization.
- The provision of mechanisms for process communication.
- The provision of mechanisms for deadlock handling.

Process Scheduling: The OS schedules processes to run on the CPU based on different scheduling algorithms such as round-robin, priority-based, and shortest job first. The scheduler determines which process should run next, and how long it should run on the CPU. Schedulers are special system software's which handles process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types.

- Long Term Scheduler

- Short Term Scheduler
- Medium Term Scheduler

Long Term Scheduler

Select which processes should be brought into the ready queue. It is also called job scheduler. Long term scheduler determines which programs are admitted to the system for processing. Job scheduler selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling. The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. Processes can be described as either:

I/O-bound process: spends more time doing I/O than computations; many short CPU burst.

CPU-bound process: spends more time doing computation than I/O; few very long CPU burst

Short Term Scheduler

Select which process should be executed next and allocates CPU. It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them. Short term scheduler also known as dispatcher, execute most frequently and makes the fine-grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

Medium Term Scheduler

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

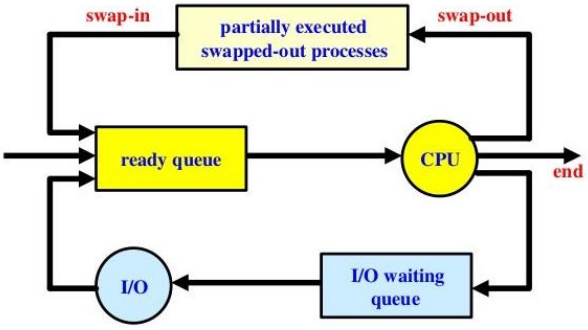


Figure 11 Medium Term Scheduler

The medium-term scheduler is Running process may become suspended if it makes an I/O request. Suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other process, the suspended process is moved to the secondary storage. This process is called swapping.

Comparison between Schedulers

Sr. No	Long Term Scheduler	Short Term Scheduler	Medium Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short- and long-term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time-sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch

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

Context switching is an essential part of a multitasking operating system features. When the scheduler switches the CPU from executing one process to execute another, the context switcher saves the content of all processor registers for the process being removed from the CPU, in its process descriptor. The context of a process is represented in the process control block of a process. Show in fig 12 Context switch time is pure overhead. Time dependent on hardware support like some hardware provide multiple set of registers per CPU.

Note: Context switch time is overhead mean, the system does not useful work while switching.

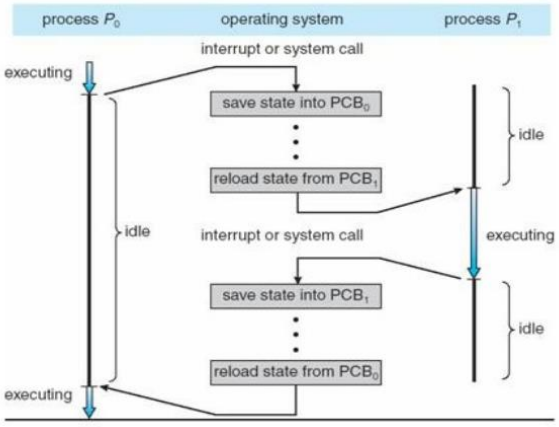


Figure 12 Context Switch

2.1.2. MEMORY MANAGEMENT

Memory management is an essential function of an operating system (OS). The OS is responsible for managing the memory resources of a computer system to ensure that processes can access the memory they require and that the system operates efficiently. The primary objective of memory management is to allocate memory to processes as needed and to ensure that memory is utilized efficiently. The OS must manage memory allocation and deallocation to prevent processes from accessing memory that has not been allocated to them, which could cause system instability or crashes.

Memory management techniques used by an OS include paging, segmentation, virtual memory, and memory mapping. The OS must also deal with fragmentation, which occurs when free memory blocks become scattered throughout the memory space. To reduce fragmentation, the OS uses techniques like compaction and defragmentation.

In addition to allocating and deallocating memory, the OS is responsible for managing shared memory and coordinating access to memory resources among multiple processes. The OS also manages the swapping of memory pages to and from the hard disk when physical memory becomes full, a process known as virtual memory.

memory management is a key function of an OS that allows multiple processes to run simultaneously on a computer system, improving the system's performance and efficiency.

- Keep track of which parts of memory are currently being used and by whom
- Decide which processes are to be loaded into memory when memory space becomes available
- Allocate and deallocate memory space as needed.

DEFINE VIRTUAL MEMORY

Virtual memory is a technique used by operating systems to simulate more physical memory (RAM) than is actually available on a computer. It works by temporarily transferring data from the RAM to the hard disk when the RAM becomes full, freeing up space for other programs to use. When that data is needed again, it is retrieved from the hard disk and loaded back into RAM.

Virtual memory allows the operating system to use the hard disk as a "swap space" for data that is not immediately needed. The size of the virtual memory is limited only by the available space on the hard disk. This means that programs can run even if the computer does not have enough physical memory to hold all the data they need at once.

Virtual memory also provides a level of security by isolating processes from each other, preventing one program from accessing the memory used by another program. This is achieved through a mechanism called memory mapping, which creates a virtual address space for each program.

EXPLAIN HOW THE OPERATING SYSTEM MANAGES THE RESOURCES OF A COMPUTER

The operating system (OS) is responsible for managing the resources of a computer system. This includes managing the hardware resources, such as the CPU, memory, and input/output devices, as well as managing the software resources, such as programs and data.

The OS manages the resources of a computer system in several ways:

- Process management
- Memory management
- Device management
- File management
- Security management

2.1.3. DISK AND STORAGE MANAGEMENT

The main purpose of a computer system is to execute programs. These programs, with the data they access, must be in main memory (primary storage) during execution. Because main memory is too small to accommodate all data and programs, and its data are lost when power is lost, the computer system must provide secondary storage to back up main memory. Most modern computer systems use disks as the principle on-line storage medium, for both programs and data. Most programs including compilers, assemblers, sort routines, editors, and formatters are stored on a disk until loaded into memory, and then use the disk as both the source and destination of their processing. Hence, the proper management of disk storage is of central importance to a computer system.

The operating system is responsible for the following activities in connection with disk management:

- Free-space management
- Storage allocation
- Disk scheduling

EXPLAIN DISK MANAGEMENT SYSTEM

Disk management is the process of organizing and managing the physical disks and volumes in a computer system. The disk management system is responsible for creating, formatting, and managing disk partitions, volumes, and file systems on a computer's hard disk drive or other storage devices. Disk management allows users to divide a single physical disk into multiple logical partitions or volumes. Each partition can be formatted with a specific file system, such as NTFS or FAT32, and can be assigned a drive letter or mount point for easy access.

Disk management also allows users to resize, extend, or shrink partitions as needed, depending on their storage requirements. In addition, disk management provides tools for checking and repairing disk errors, optimizing disk performance, and defragmenting disks to improve file access speeds.

Other features of disk management systems may include support for advanced disk features, such as RAID (redundant array of independent disks), disk mirroring, and disk encryption. These features can help to improve data protection and security for users and organizations that rely on computer storage systems.

2.1.4. FILE SYSTEM MANAGEMENT

File System management is one of the most visible components of an operating system. Computers can store information on several different types of physical media. Magnetic tape, magnetic disk, and optical disk are the most common media. Each of these media has its own characteristics and physical organization. Each medium is controlled by a device, such as a disk drive or tape drive, with its own unique characteristics. These properties include speed, capacity, data - transfer rate, and access method (sequential or random-access method).

File system management is a critical component of an operating system (OS) that is responsible for organizing and managing files on storage devices such as hard disks, solid-state drives, and network file systems. The file system is a logical structure that provides a way to organize, store, and retrieve data efficiently. The following are some key tasks that the OS performs in file system management:

File Creation: When a user creates a new file, the OS allocates space on the storage device to store the file and creates a file descriptor to track the file's metadata, such as its name, location, size, and access permissions.

File Naming and Directory Management: The OS provides a hierarchical structure of directories and subdirectories that allows users to organize their files logically. The OS manages the naming and organization of directories and files to prevent conflicts and ensure efficient storage and retrieval.

File Access and Protection: The OS controls access to files by enforcing permissions and security policies that determine which users or processes can read, write, or execute files. The OS also provides file locking mechanisms to

prevent conflicts when multiple processes attempt to access the same file simultaneously.

File System Monitoring and Repair: The OS monitors the file system for errors and inconsistencies that could cause data corruption or loss. The OS also provides tools for users to diagnose and repair file system errors, such as disk utilities and file system checkers.

File Backup and Recovery: The OS provides tools for backing up and restoring files to protect against data loss due to hardware failures, software errors, or malicious attacks.

Effective file system management is critical to ensure that data is stored and retrieved efficiently and securely. The OS provides a range of file system management tools and mechanisms that allow users to organize, access, and protect their data effectively.

- The creation and deletion of files
- The creation and deletion of directories.
- The support of primitives for manipulating files and directories
- The backup of files on stable (nonvolatile) storage media.

DESCRIBE HIERARCHICAL SYSTEM.

A hierarchical directory system, also known as a tree directory structure, is a way of organizing files and folders on a computer system by placing them in a hierarchical structure, similar to a tree. This system is used by most modern operating systems, including Windows, macOS, and Linux.

In this system, the top level of the hierarchy is called the root directory, which contains all other directories and files. Each directory or folder can contain other directories or files, creating a branching structure that resembles a tree.

The directory tree is usually displayed in a file manager or command-line interface, with each level of the hierarchy represented by a folder icon or text label. Users can navigate through the tree by clicking on the folder icons or entering commands to move up or down the hierarchy.

One of the benefits of a hierarchical directory system is that it allows for easy organization and retrieval of files. Users can group related files and folders together in a logical way, making it easier to find the information they need. Additionally, the system allows for easy backup and sharing of files, as they can be organized into discrete folders that can be easily transferred or copied.

DESCRIBE UNIX FILE SYSTEM

The UNIX file system is a hierarchical file system used by the UNIX operating system and its variants, such as Linux and macOS. It is organized into a tree-like structure, with the root directory "/" at the top of the hierarchy.

Each directory in the file system can contain files and subdirectories, which can in turn contain additional files and directories. Directories are represented as regular files that contain a list of filenames and pointers to the locations of the corresponding files and directories.

The UNIX file system supports several types of files, including regular files, directories, special files, and symbolic links. Regular files contain data in a specific format, such as text or binary, while directories contain pointers to other files and directories. Special files, such as device files, provide access to system devices such as printers or network interfaces. Symbolic links are special files that point to another file or directory in the file system.

One of the key features of the UNIX file system is its support for permissions and ownership. Each file and directory in the file system is associated with a set of permissions that specify who can read, write, or execute the file, as well as who can change those permissions.

2.2. OS SERVICES AND SYSTEM CALLS

Operating system services: An operating system provides an environment for the execution of programs. The operating system provides certain services to programs and to the users of those programs. The specific

services provided will, of course, differ from one operating system to another, but there are some common classes that we can identify. These operating-system services are provided for the convenience of the programmer, to make the programming task easier.

ENUMERATE OS SERVICES AND EXPLAIN THE SERVICE

Operating systems provide a variety of services to users and applications, such as:

Process management: The OS manages the creation, execution, and termination of processes, which are running instances of programs. The OS provides resources to processes, such as CPU time, memory, and input/output devices, and manages process scheduling to ensure that each process gets a fair share of the resources.

Memory management: The OS manages the allocation and deallocation of memory to processes, as well as virtual memory management to allow processes to use more memory than is physically available on the system. The OS also handles memory protection to prevent processes from accessing each other's memory areas.

File management: The OS provides a file system that allows applications to create, read, write, and delete files on the system. The file system manages file permissions, metadata, and storage locations.

Device management: The OS manages the input/output devices of the computer system, such as keyboards, mice, and printers. The OS provides a set of device drivers that allow applications to communicate with the devices, and manages device access and scheduling to ensure that multiple applications can use the devices simultaneously.

Network management: The OS manages network connections and communication, providing services such as IP addressing, routing, and security. The OS also provides network protocols and interfaces that allow applications to send and receive data over the network.

Security management: The OS provides security features such as authentication, access control, and encryption to protect the system from unauthorized access and malicious attacks. The OS also manages user accounts, passwords, and permissions to control access to system resources.

User interface: The OS provides a user interface, such as a graphical user interface (GUI) or command-line interface (CLI), that allows users to interact with the system and run applications.

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

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

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

The system must be able to load a program into memory and to run it. The program must be able to end its execution, either normally or abnormally (indicating error).

I/O operations: An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users. An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.

- Operating system provides the access to the required I/O device when required.

A running program may require I/O. This I/O may involve a file or an I/O device. For specific devices, special functions may be desired (such as rewind a tape drive, or blank the screen on a CRT). For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide some means to do I/O.

File-system manipulation: A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods. A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

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

The file system is of particular interest. It should be obvious that programs need to read and write files. They also need to create and delete files by name.

Communications: In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the

operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network. The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

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

There are many circumstances in which one process needs to exchange information with another process. There are two major ways in which such communication can occur. The first takes place between processes executing on the same computer; the second takes place between processes executing on different computer systems that are tied together by a computer network. Communications may be implemented via shared memory, or by the technique of message passing, in which packets of information are moved between processes by the operating system.

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

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

The operating system constantly needs to be aware of possible errors. Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices (such as a parity error on tape, a connection failure on a network, or lack of paper in the printer), or in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or a too great use of CPU time). For each type of error, the operating

system should take the appropriate action to ensure correct and consistent computing.

In addition, another set of operating-system functions exists not for helping the user, but rather for ensuring the efficient operation of the system itself. Systems with multiple users can gain efficiency by sharing the computer resources among the users.

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

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

Resource allocation is the process by which a computing system aims to meet the hardware requirements of an application run by it. When there are many jobs running at the same time, resources must be allocated to each of them. Many different types of resources are managed by the operating system. Some (such as CPU cycles, main memory, and file storage) may have special allocation code, whereas others (such as I/O devices) may have much more general request and release code. For instance, in determining how best to use the CPU, operating systems have CPU-scheduling routines that take into account the speed of the CPU, the jobs that must be executed, the number of registers available, and other factors. There might also be routines to allocate a tape drive for use by a job. One such routine locates an unused tape drive and marks an internal table to record the drive's new user. Another routine is used to clear that table. These routines may also be used to allocate plotters, modems, and other peripheral devices.

Accounting: We want to keep track of which users use how much and what kinds of computer resources. This record keeping may be for accounting (so that users can be billed) or simply for accumulating usage statistics. Usage statistics may be a valuable tool for researchers who wish to reconfigure the. system to improve computing services.

Protection: Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

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

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

The owners of information stored in a multiuser computer system may want to control its use. When several disjoint processes execute concurrently, it should not be possible for one process to interfere with the others, or with the operating system itself. Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important. Such security starts with each user having to authenticate himself or herself to the system, usually by means of a password, to be allowed access to the resources. It extends to defending external I/O devices, including modems and network adapters, from invalid access attempts, and to recording all such connections for detection of break-ins. If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

DEFINE SYSTEM CALLS

A system call is a programming interface provided by the operating system (OS) to allow applications to request services from the kernel, which is the core of the OS. System calls provide a way for applications to interact with the underlying hardware and software of the computer system, without having to directly access the system resources.

A system call is a function that a user program uses to ask the operating system for a particular service. User programmers can communicate with the operating system to request its services using the interface that is created by a system call.

System calls serve as the interface between an operating system and a process. System calls can typically be found as assembly language instructions. They are also covered in the manuals that the programmers working at the assembly level use. When a process in user mode needs access to a resource, system calls are typically generated. The resource is then requested from the kernel via a system call.

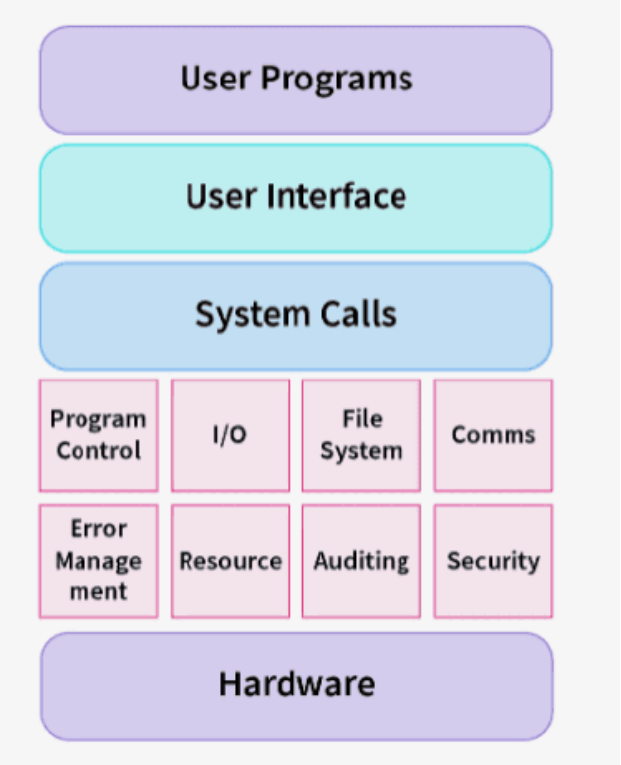


Figure 1 architecture of System Call

System call uses Application Programming Interfaces to provide operating system services to user programmes (API). It offers a point of contact between processes and the operating system so that user-level processes can ask for

the operating system's assistance. The only ways to access the kernel system are through system calls. System calls are required for all programmes that require resources.

The following is a diagram that illustrates what is system call in an operating system and how the system call is executed: As you can see in the below-given System Call example diagram.

Step 1) The processes executed in the user mode till the time a system call interrupts it.
Step 2) After that, the system call is executed in the kernel-mode on a priority basis.
Step 3) Once system call execution is over, control returns to the user mode.,
Step 4) The execution of user processes resumed in Kernel mode.

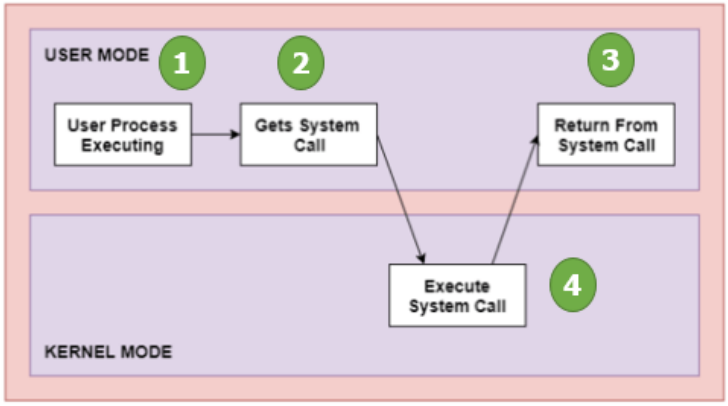


Figure 13 system calls process

Why do you need System Calls in OS?

Following are situations which need system calls in OS:

- Reading and writing from files demand system calls.
- If a file system wants to create or delete files, system calls are required.
- System calls are used for the creation and management of new processes.
- Network connections need system calls for sending and receiving packets.
- Access to hardware devices like scanner, printer, need a system call.

Types of System calls

Here are the five types of System Calls in OS:

Figure 14 Types of System Calls

Process Control

This system calls perform the task of process creation, process termination, etc.

- System calls under process control: end, abort, load, execute, create process, terminate process, get process attributes, set process attributes, wait for time, wait for event, signal event, allocate and free memory.

File Management

File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc.

- Some System calls under File management: create file, delete file, open close file, read, write, reposition get and set file attributes.

Device Management

Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc.

- Some System calls under Device management: request device, release device, read, write, reposition get device attributes, set device attributes, logically attach or de-attach devices.

Information Maintenance

It handles information and its transfer between the OS and the user program.

- Some System calls under Information maintenance: get time or date, set time or date, get system data, set system data, get and set process, file, or device attributes.

Communication:

These types of system calls are specially used for inter process communications.

- Some System calls under communication control: create, delete communication, connection send, User space, Application System Calls, Kernel receives messages transfer status information attach and de-attach remote devices.

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Figure 15 example of windows and Unix System Calls

2.3. SYSTEM PROGRAMS AND STRUCTURE

System Program

Another aspect of a modern system is the collection of system programs. Recall Figure 5, which depicted the logical computer hierarchy. At the lowest level is hardware, of course. Next is the operating system, then the system programs, and finally the application programs.

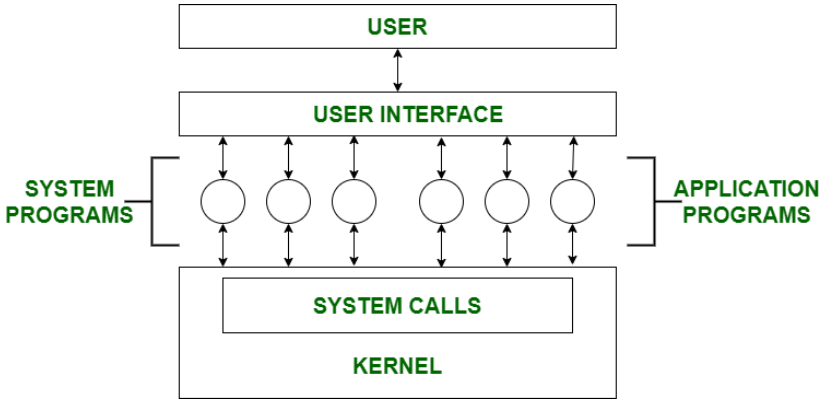


Figure 16 Modern System Program

System programs provide a more convenient environment for program development and execution. Some of them are simply user interfaces to system calls, whereas others are considerably more complex.

IDENTIFY SYSTEM PROGRAMS THAT COME WITH THE OPERATING SYSTEM

1. Command shell
2. File manager
3. Device drivers
4. Text editor
5. Backup and recovery tools
6. System monitoring tools
7. Antivirus and security software

DESCRIBE THE FUNCTIONS OF THESE SYSTEM PROGRAMS

System programs are software programs that are included with the operating system to provide essential functionality and services to the user and the system. Here are some examples of system programs that come with the operating system:

Command shell: A command shell is a program that provides a command-line interface for interacting with the operating system. The command shell allows users to execute commands and scripts, as well as to manage files and directories.

File manager: A file manager is a program that provides a graphical user interface for managing files and directories on the system. The file manager allows users to view, create, edit, and delete files and directories, as well as to perform other file-related tasks.

Device drivers: Device drivers are programs that allow the operating system to communicate with hardware devices, such as printers, scanners, and network cards. Device drivers provide a standard interface for applications to access the devices, and manage device-specific tasks, such as printing and scanning.

Text editor: A text editor is a program that allows users to create, edit, and save text files. Text editors provide basic text formatting and editing functionality, such as search and replace, undo and redo, and syntax highlighting.

Backup and recovery tools: Backup and recovery tools are programs that allow users to create backups of their files and data, and to restore them in case of data loss or system failure. Backup and recovery tools provide various backup strategies, such as full backups, incremental backups, and differential backups.

System monitoring tools: System monitoring tools are programs that allow users to monitor the performance and health of the system, and to diagnose and troubleshoot problems. System monitoring tools provide information

about system resources, such as CPU, memory, and disk usage, as well as about running processes and network activity.

Antivirus and security software: Antivirus and security software are programs that protect the system from viruses, malware, and other security threats. Antivirus and security software provide real-time scanning and detection of malicious software, as well as scheduled scans and updates.

These are just some examples of system programs that come with the operating system. The specific set of system programs may vary depending on the operating system and its version.

Most operating systems are supplied with programs that are useful to solve common problems, or to perform common operations. Such programs include web browsers, word processors and text formatters, spreadsheets, database systems, compiler compilers, plotting and statistical-analysis packages, and games. These programs are known as system utilities or application programs. Perhaps the most important system program for an operating system is the command interpreter, the main function of which is to get and execute the next user-specified command.

Many of the commands given at this level manipulate files: create, delete, list, print, copy, execute, and so on. There are two general ways in which these commands can be implemented. In one approach, the command interpreter itself contains the code to execute the command. For example, a command to delete a file may cause the command interpreter to jump to a section of its code that sets up the parameters and makes the appropriate system call. In this case, the number of commands that can be given determines the size of the command interpreter, since each command requires its own implementing code.

An alternative approach used by UNIX, among other operating systems, implements most commands by special systems programs. In this case, the command interpreter does not "understand" the command in any way; it merely uses the command to identify a file to be loaded into memory and executed. Thus, a command

delete G

would search for a file called delete, load the file into memory, and execute it with the parameter G. The function associated with the delete command would be defined

completely by the code in the file delete. In this way, programmers can add new commands to the system easily by creating new files of the proper name. The command-interpreter program, which can now be quite small, does not have to be changed for new commands to be added.

DESCRIBE THE GENERAL STRUCTURE AND ARCHITECTURE OF AN OPERATING SYSTEM

System Structure

Operating system can be implemented with the help of various structures. For efficient performance and implementation an OS should be partitioned into separate subsystems, each with carefully defined tasks, inputs, outputs, and performance characteristics. These subsystems can then be arranged in various architectural configurations:

Simple structure

Many operating systems do not have well-defined structures. MS-DOS is an example of such a system. When DOS was originally written its developers had no idea how big and important it would eventually become. It was written by a few programmers in a relatively short amount of time, without the benefit of modern software engineering techniques, and then gradually grew over time to exceed its original expectations. It does not break the system into subsystems, and has no distinction between user and kernel modes, allowing all programs direct access to the underlying hardware.

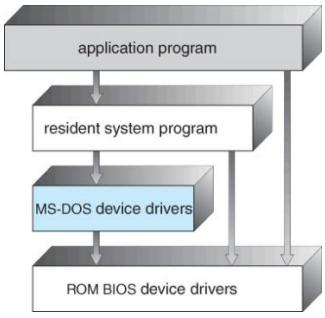


Figure 17 Simple Structure

MS-DOS layer structure

The original UNIX OS used a simple layered approach, but almost all the OS was in one big layer, not really breaking the OS down into layered subsystems:

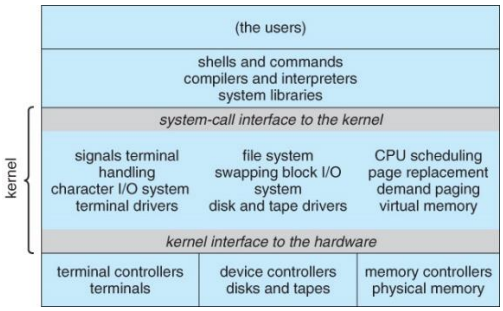


Figure 18 Traditional UNIX system structure

Layered Structure:

- Another approach is to break the OS into a number of smaller layers, each of which rests on the layer below it, and relies solely on the services provided by the next lower layer.
- This approach allows each layer to be developed and debugged independently, with the assumption that all lower layers have already been debugged and are trusted to deliver proper services.
- The problem is deciding what order in which to place the layers, as no layer can call upon the services of any higher layer.
- Layered approaches can also be less efficient, as a request for service from a higher layer has to filter through all lower layers before it reaches the Hardware, possibly with significant processing at each step.

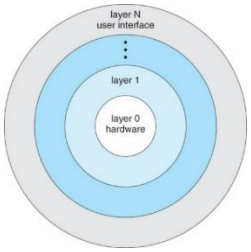


Figure 19 Layered Structure

Micro-kernel

- The basic idea behind micro kernels is to remove all non-essential services from the kernel, and implement them as system applications instead, thereby making the kernel as small and efficient as possible.
- Most microkernels provide basic process and memory management, and message passing between other services, and not much more.
- Security and protection can be enhanced, as most services are performed in user mode, not kernel mode.
- System expansion can also be easier, because it only involves adding more system applications, not rebuilding a new kernel.
- Mach was the first and most widely known microkernel, and now forms a major component of Mac OSX.
- Windows NT was originally microkernel, but suffered from performance problems relative to Windows 95. NT 4.0 improved performance by moving more services into the kernel, and now XP is back to being more monolithic.
- Another microkernel example is QNX, a real-time OS for embedded systems.

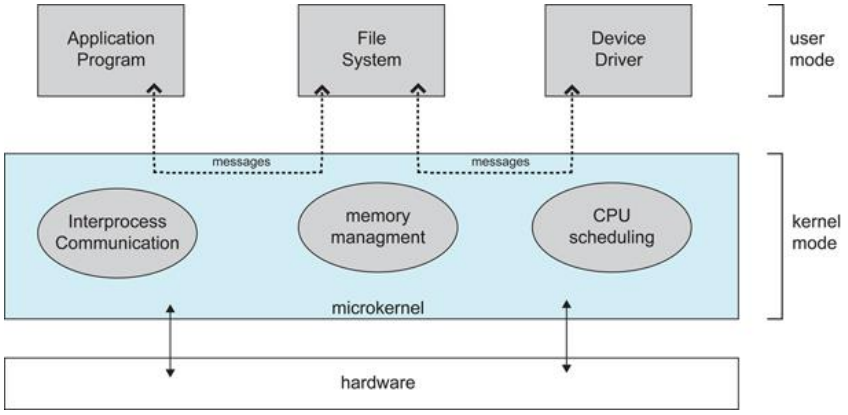


Figure 20 Architecture of a Typical Microkernel

Modular structure or approach:

It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time.

- Modern OS development is object-oriented, with a relatively small core kernel and a set of modules which can be linked in dynamically. See for example the Solaris structure, as shown in Figure 13 below.
- Modules are similar to layers in that each subsystem has clearly defined tasks and interfaces, but any module is free to contact any other module, eliminating the problems of going through multiple intermediary layers.
- The kernel is relatively small in this architecture, similar to microkernels, but the kernel does not have to implement message passing since modules are free to contact each other directly.

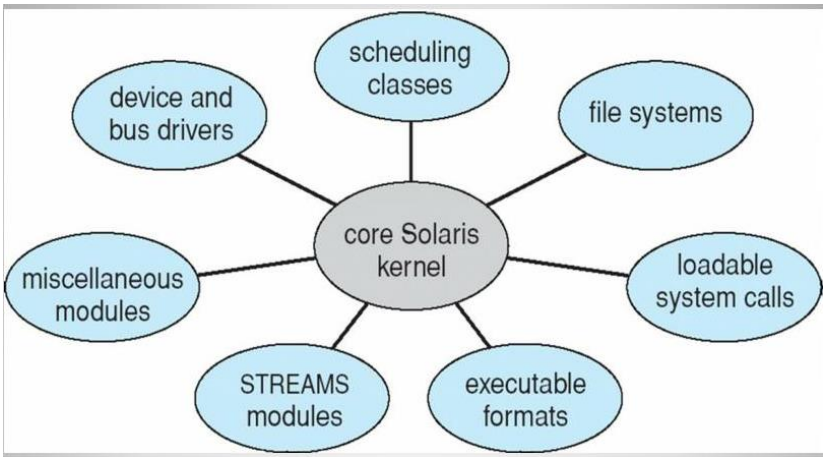


Figure 21 Modular Structure or Approach

Hybrid Systems

Most OSes today do not strictly adhere to one architecture, but are hybrids of several.

Mac OS X

The Max OSX architecture relies on the Mach microkernel for basic system management services, and the BSD kernel for additional services. Application services and dynamically loadable modules (kernel extensions) provide the rest of the OS functionality.

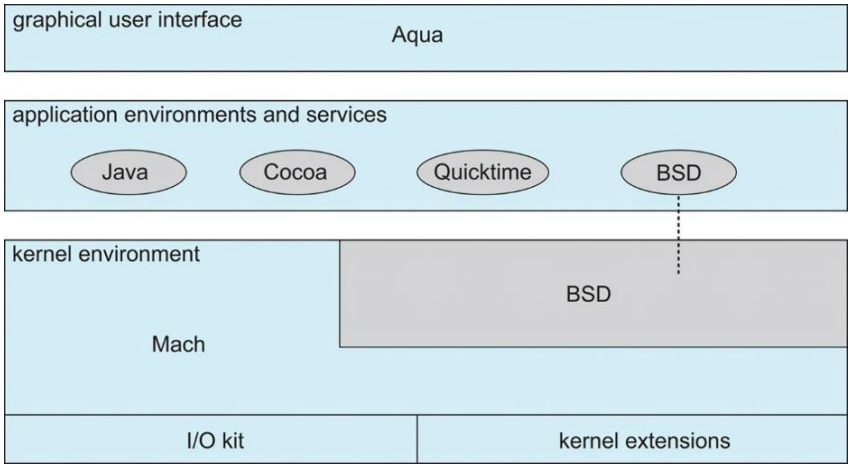


Figure 22 The Mac OS X structure

iOS

The iOS operating system was developed by Apple for iPhones and iPads. It runs with less memory and computing power needs than Max OS X, and supports touchscreen interface and graphics for small screens.

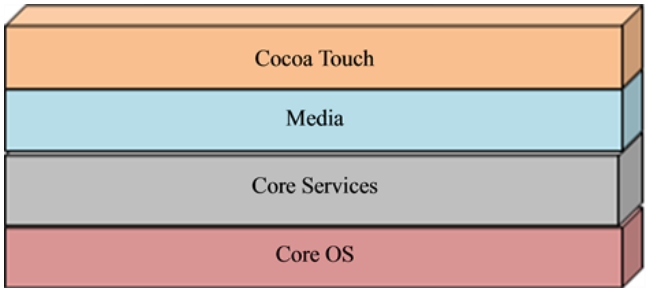


Figure 23 Architecture of Apple's iOS.

Android

- The Android OS was developed for Android smartphones and tablets by the Open Handset Alliance, primarily Google.
- Android is an open-source OS, as opposed to iOS, which has led to its popularity.
- Android includes versions of Linux and a Java virtual machine both optimized for small platforms.

- Android apps are developed using a special Java-for-Android development environment.

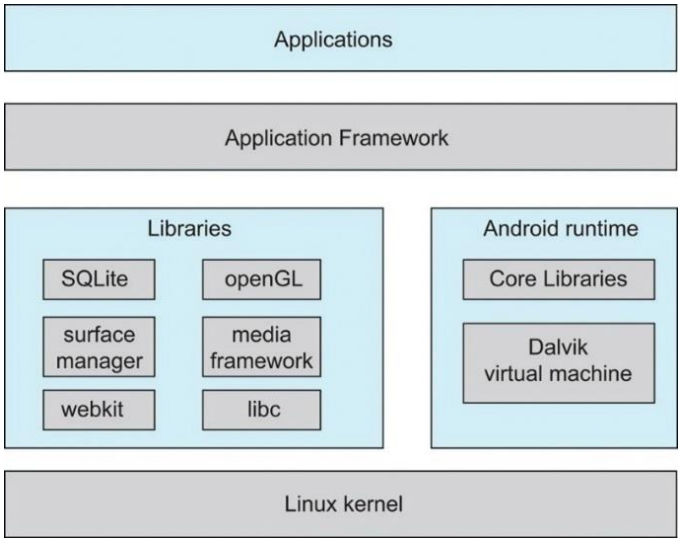


Figure 24 Architecture of Google's Android

ILLUSTRATE THE STRUCTURE BY DIAGRAMS

Here is a simplified diagram illustrating the general structure and architecture of an operating system.

User Interface
Application
System Libraries
Device Driver
Kernal

In this diagram, the user interface layer is at the top, representing the layer that allows users to interact with the operating system and the applications. The applications layer is below the user interface layer, representing the layer of programs that run on top of the operating system and use its services and resources.

The system libraries layer is below the applications layer, representing the layer of reusable code that provides common functionality to applications. The device drivers layer is below the system libraries layer, representing the layer of programs that allow the operating system to communicate with hardware devices. Finally, the kernel layer is at the bottom, representing the core component of the operating system that manages the system resources and provides a low-level interface for applications to access the system resources.

2.4. SYSTEM DESIGN AND IMPLEMENTATION

In this section, we discuss the problems of designing and implementing a system. There are, of course, no complete solutions to the design problems, but there are approaches that have been successful.

Design Goals:

The first problem in designing a system is to define the goals and specifications of the system. At the highest level, the design of the system will be affected by the choice of hardware and type of system: batch, time-shared, single-user, multiuser, distributed, real-time, or general purpose. Beyond this highest design level, the requirements may be much harder to specify. The requirements can be divided into two basic groups: user goals and system goals.

DISCUSS DESIGN GOALS OF THE OPERATING SYSTEM

The design goals of an operating system are the overarching objectives that guide the development of the system. These goals are influenced by the needs of the users, the hardware and software environment, and the system requirements. Here are some common design goals of an operating system:

Efficiency: An operating system should be designed to use system resources efficiently, including memory, CPU, and I/O devices. This includes optimizing

resource allocation and scheduling to ensure that system resources are used effectively and to minimize waste.

Reliability: An operating system should be designed to provide reliable and consistent performance, even in the face of errors or failures. This includes providing robust error handling and recovery mechanisms to ensure that the system remains operational and responsive.

Security: An operating system should be designed to provide secure access to system resources and protect against unauthorized access or malicious attacks. This includes implementing authentication and authorization mechanisms, encrypting data in transit and at rest, and providing protection against viruses and malware.

Compatibility: An operating system should be designed to support a wide range of hardware and software configurations, and to provide a stable and consistent interface for applications. This includes providing support for legacy hardware and software, and maintaining compatibility with industry standards and protocols.

Scalability: An operating system should be designed to support the growth and expansion of the system over time, including support for multiple users and multiple applications running simultaneously. This includes providing scalable resource management and scheduling, and optimizing system performance for high-load conditions.

Maintainability: An operating system should be designed to be easily maintainable and upgradable, including providing tools and utilities for system administration and maintenance. This includes providing documentation and support resources for system administrators and end-users.

These design goals are not mutually exclusive and often overlap with each other. For example, improving security may require changes to the efficiency and reliability of the system. Balancing these goals and prioritizing them according to the specific needs of the system is a critical part of operating system design.

Mechanisms and Policies

One important principle is the separation of policy from mechanism. Mechanisms determine how to do something. In contrast, policies decide what will be done. For example, a mechanism for ensuring CPU protection is the timer construct of how long the timer is set for a particular user, on the other hand, is a policy decision.

Mechanisms and policies are two important concepts in operating system design that work together to provide the necessary functionality and behavior of the system.

Mechanisms refer to the actual implementation of system features, such as memory allocation, process scheduling, and file management. They are the low-level mechanisms that provide the essential services of the operating system. For example, the mechanism of virtual memory allows the operating system to manage memory more efficiently by swapping data between RAM and disk as needed.

Policies, on the other hand, refer to the high-level rules and decision-making processes that determine how resources are allocated and how tasks are prioritized. Policies are typically defined in terms of goals, such as maximizing system throughput or minimizing response time. For example, the policy of process scheduling determines which process should be given access to the CPU at any given time, and for how long. Here are some examples of mechanisms and policies in an operating system:

1. **Memory Management:**

- **Mechanisms:** virtual memory, paging, segmentation
- **Policies:** page replacement algorithms, memory allocation policies.

2. **Process Scheduling:**

- **Mechanisms:** context switching, interrupt handling, dispatching

- **Policies:** scheduling algorithms (e.g., round-robin, priority-based), thread prioritization, preemption policies.

3. **File Management:**
- **Mechanisms:** file systems, disk management, file locking
- **Policies:** file access permissions, disk quota policies, caching policies.

4. **Network Management:**
- **Mechanisms:** network protocols, socket programming, packet handling.
- **Policies:** congestion control, routing policies, Quality of Service (QoS) policies.

The distinction between mechanisms and policies is important because it allows the operating system to be flexible and adaptable to different environments and use cases. By separating the implementation details (mechanisms) from the high-level decision-making (policies), the operating system can be customized to meet the specific needs of the system and its users.

Implementation

Once an operating system is designed, it must be implemented. Traditionally, operating systems have been written in assembly language. However, that is generally no longer true. Operating systems can now be written in higher-level languages. The implementation of an operating system involves several key steps that typically include:

Design: The first step in implementing an operating system is to define the design goals and requirements. This includes identifying the target hardware and software environment, defining the system architecture and components, and establishing performance metrics.

Kernel Implementation: The next step is to implement the kernel, which is the core of the operating system that provides low-level functionality such as process and memory management. The kernel is typically written in a low-

level programming language, such as C or assembly language, and requires careful attention to detail and performance optimization.

Device Driver Implementation: The device drivers are responsible for controlling and communicating with hardware devices such as the hard drive, keyboard, and mouse. These drivers are typically implemented as part of the kernel, and require detailed knowledge of the hardware architecture and interfaces.

System Services Implementation: The system services are higher-level functions that provide functionality such as file management, network communication, and user interface management. These services are typically implemented in a higher-level programming language, such as C++, and may rely on the kernel and device drivers for low-level functionality.

User Interface Implementation: The user interface is responsible for presenting information to the user and receiving input from the user. This can include graphical user interfaces (GUIs), command-line interfaces (CLIs), or a combination of both. The user interface is typically implemented using a combination of system services and application programming interfaces (APIs).

Testing and Debugging: Once the operating system is implemented, it must be thoroughly tested and debugged to ensure that it meets the design goals and requirements. This involves both automated testing and manual testing, as well as detailed performance analysis and profiling.

Deployment and Maintenance: Finally, the operating system must be deployed to the target hardware and software environment, and maintained over time to ensure that it remains stable, secure, and performant. This involves ongoing maintenance and bug fixes, as well as updates and upgrades to support new hardware and software. Today, operating systems are much smaller and easier to use. They have been designed to be modular so they can be customized by users or developers. There are many different types of operating systems:

- **Graphical user interfaces** (GUIs) like Microsoft Windows and Mac OS.

Figure 25 Example of GUI

- **Command line interfaces** like Linux or UNIX, DOS.

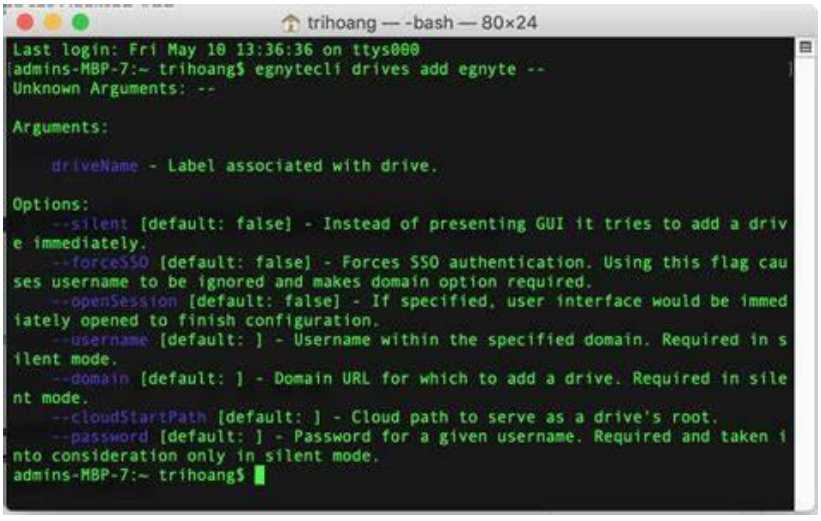


Figure 26 Example of CLI

- **Real-time operating systems** that control industrial and scientific equipment.

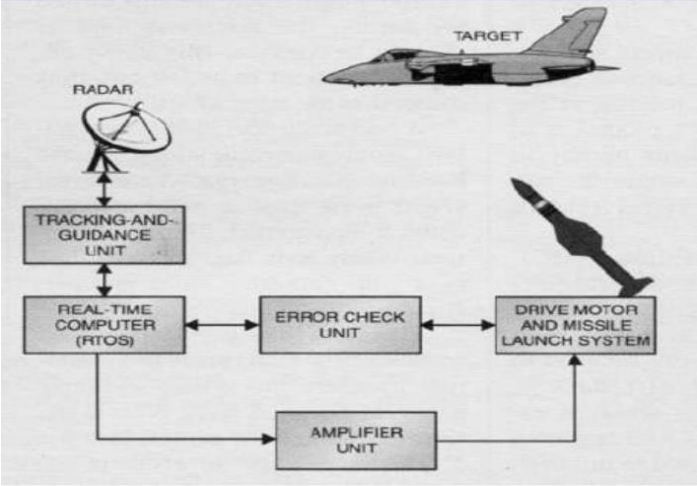


Figure 27 Example of Real Time Operating System

- **Embedded operating systems** are designed to run on a single computer system without needing an external display or keyboard.

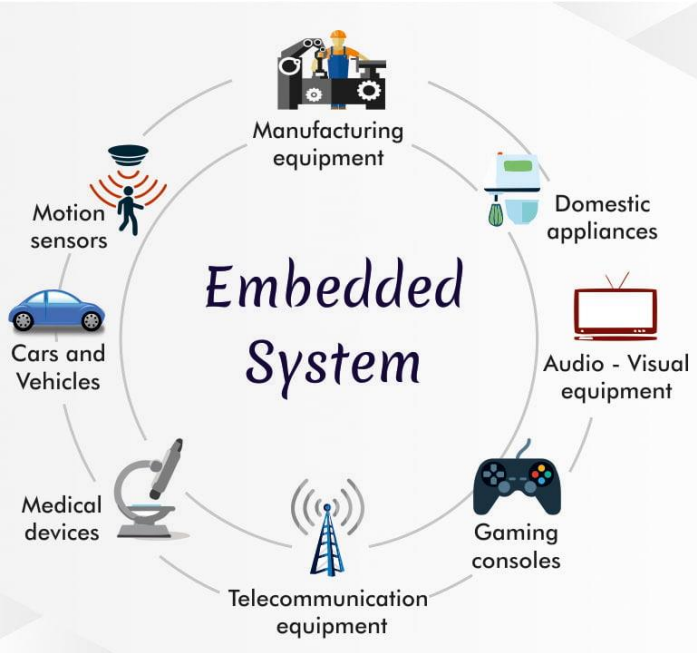


Figure 28 Example of Embedded System

Multiple Choice Questions

Q.1: operating system called a thousand systems are executed per?

(a) hour (b) day

(c) min (d) sec

Q.2: Choose one of the best options from the following. Dump of memory of the computer system is examined by the ___?

(a) programmer (b) debugger

(c) designer (d) engineer

Q.3: Indicate one of the best options from the following. CloseHandle () call function in windows operating system is a UNIX's function called for

(a) fork() (b) close ()

(c) read() (c) open ()

Q.4: In operating system, each process has its own _____

(a) open files (b) address space and global variables

(c) pending alarms, signals, and signal handlers

(d) All of above

Q.5: Which is not provided by the operating system is____

(a) networking (b) user interface

(c) error detection (d) program execution

Q.6: Choose one of the best options from the following. Chain of the program of a computer system is as strong as its ___

(a) strongest link (b) weakest link

(c) all links (d) mid link

Q.7: Which is Friendly user interface provided by operating system ____

(a) graphical user interface (b) command-line interface

(c) batch interface (d) device interface

Q.8: Which Bourne shell in command interpreter of the operating system is used by____

(a) windows (b) Windows and Macintosh

(c) Macintosh and LINUX (d) Linux and UNIX

Q.9: Programs of the operating system end ____.

(a) Interval (b) Interruptedly

(c) Normally (d) Erroneously

Q.10: Choose the best option from the following. The main functionality of command interpreter of the operating system is to_____.

(a) remove commands (b) execute commands

(c) fetch commands (d) Decode commands

Q.11: Power failure of a computer system is one of kinds of____

(a) error (b) outputs

(c) inputs (d) Interrupts

Q.12: X-windows operating system is a user's interface that is a common version of____?

(a) windows (b) MAC OS

(c) Linux (d) UNIX

Q.13: Operating system is mainly comprised of _____

(a) third party programs (b) user programs

(c) system programs (d) both B and C

Q.14: Operating system programs that are manipulated are asked by the_____?

(a) programmer (b) open source

(c) user (d) operating system

Q.15: System calls of operating system is done by___?

(a) programmer (b) debugger

(c) Caller (d) engineer

Q.16: For allocating resources Central Processing Unit has individual _____?

(a) routines (b) devices

(c) programs (d) Processes

Q.17: Indicate best option. In graphical user interface users uses____

(a) file-based interface (b) mouse-based Pointing

(c) command line interface (d) voice-based interface

ANSWER KEY

Q.1 (d)	Q.2 (b)	Q.3 (b)	Q.4 (d)	Q.5 (a)
Q.6 (b)	Q.7 (a)	Q.8 (d)	Q.9 (c)	Q.10 (b)
Q.11 (a)	Q.12 (d)	Q.13 (d)	Q.14 (c)	Q.15 (c)
Q.16 (a)	Q.17(b)			

Short Questions

1. Enlist the functions of an operating system?
2. Describe the functions of an operating system?
3. Explain memory management techniques??
4. Describe virtual memory?
5. Describe hierarchical directory system?
6. Describe UNIX file system?
7. List five services provided by an operating system?
8. Explain how the operating system manages the resources of a computer?
9. Explain disk management system?
10. Define system calls and also describe What is the purpose of system calls?
11. Write down the steps and its diagram for system call process?
12. Why do you need system calls in operating system?
13. Describe Error Detection in operating system?
14. Enumerate OS services and explain the service?
15. Identify system programs that come with the operating system?
16. Describe the functions of these system programs?
17. What is resource Allocation in operating system?
18. Which rules for passing parameters for system call?
19. Define file manipulation?
20. Difference between Real time and Embedded operating system?
21. Difference between GUI and CUI?
22. Define program loading and execution?
23. Illustrate the structure by diagrams?
24. Enlist the design goals of the operating system?

Long Questions

1. Explain system components and also describe its modern goals of system components?
2. What is system calls and also describe its type of system calls?
3. Write a detail note on Operating system services?
4. Explain the general structure and architecture of an operating system?
5. Explain system design and implementation in operating system?

Bibliography

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar

CHAPTER 03 Unix/Linux Implementation

Objectives
After completion of this chapter students will be able to:

3.1 Logging in and Logging Out to the System
3.2 Configuring the Environment and Managing the Password
3.3 Unix/Linux Manual System
3.4 Unix/Linux File System and File System Organization
3.5 File Types, Names and Directories
3.6 Managing Directories: File and Directory Permissions

DISCUSS THE DEVELOPMENT OF UNIX/LINUX DEVELOPMENT

What is UNIX?

UNIX is an operating system which was first developed in the 1960s. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multitasking system for servers, desktops and laptops. It is only command based.

A UNIX is a multi-user, multi-tasking environment. Unlike personal computers, UNIX systems are inherently designed to allow simultaneous access to multiple users. UNIX on a large, multi-user system or have a dedicated UNIX-based workstation on your desk, the multi-user, multi-tasking architecture of the operating system influences the way you will work with the system and the requirements it will place on you as a user and a system administrator.

History of UNIX

UNIX has been a popular OS for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities.

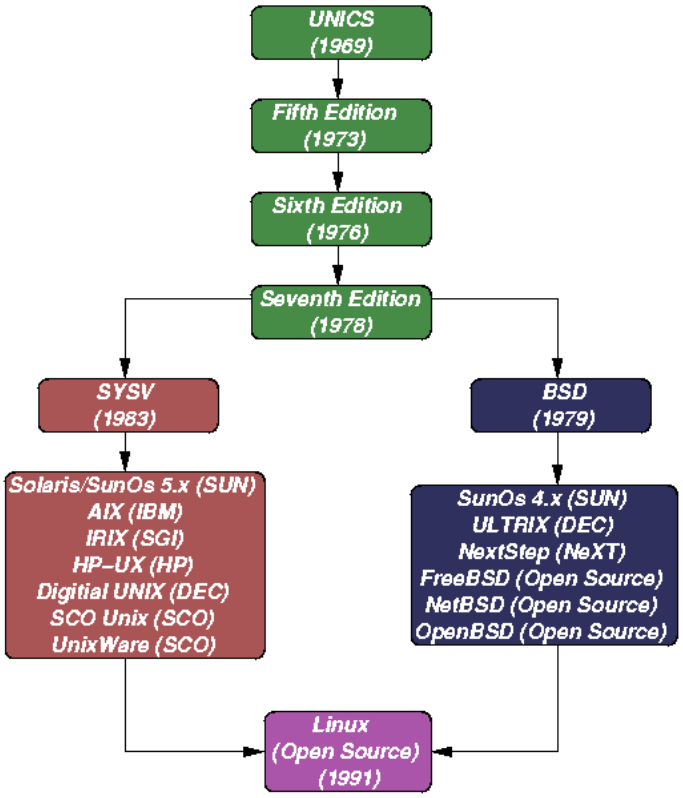


Figure 1. Simplified UNIX Family Tree

In the late 1960s, researchers from General Electric, MIT and Bell Labs launched a joint project to develop an ambitious multi-user, multi-tasking OS for mainframe computers known as MULTICS (Multiplexed Information and Computing System). MULTICS failed (for some MULTICS 74 enthusiasts "failed" is perhaps too strong a word to use here), but Ken Thompson writing a simpler operating system himself, known as UNICS (Uniplexed Information and Computing System).

Those days CPU power and memory is short, so command of this OS is suitable for those computers, this OS use short command, less space, time etc. hence the tradition of short UNIX commands we use today, e.g., ls, cp, rm, mv etc. Ken Thompson the author of the first C compiler in 1973. They rewrote the UNIX kernel in C - and released the Fifth Edition of UNIX to universities in 1974. UNIX development into two main branches: SYSV (System 5) and BSD (Berkeley Software Distribution).

Architecture of the Linux OS

Linux has all of the components of a typical OS.

Kernel

A 'kernel' is the central or Core component of most computer operating systems; it acts as a bridge between application and the data processing performed at the hardware level. The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).

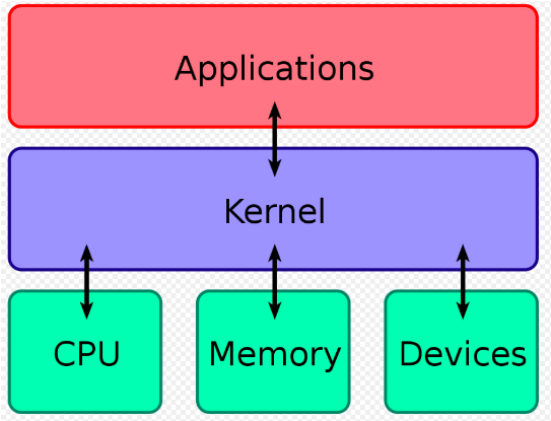


Figure 2. Kernel

A computer kernel interfaces between the three major computer hardware components, providing services between the application/user interface and the CPU, memory and other hardware I/O devices. The kernel is responsible for:

- Process management for application execution
- Memory management, allocation and I/O
- Device management through the use of device drivers
- System calls control, which is essential for the execution of kernel services.

When an OS is loaded into memory, the kernel loads first and remains in memory until the OS is shut down again.

Shells and GUIs

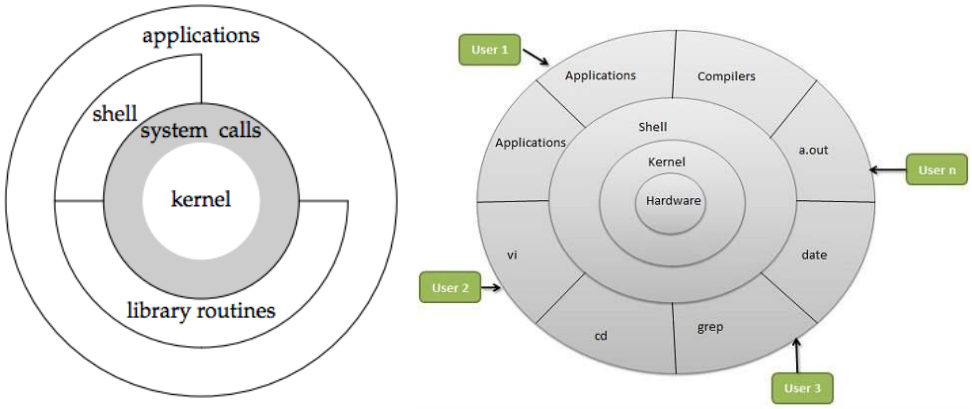
A shell is software that provides an interface for an operating system's users to provide access to the kernel's services. OR A shell is a user interface for access to an operating system service. On Unix-based or Linux-based operating systems, a shell can be invoked through the shell command in the command line interface (CLI), allowing users to direct operations through computer commands, text or script. UNIX shells are divided into four categories:

- Bourne-like shells
- C shell-like shells
- Nontraditional shells
- Historical shells

System Utilities A program that performs a very specific task, usually related to managing system resources. Operating system contains a number of utilities for managing disk drivers, printers, and other devices.

Application programs

An application software or group of programs that is designed for end user. It is used to perform various applications on computer, also known as application package. Ms Word, Access Photoshop is the example of Application software.



DESIGN PRINCIPLE OF UNIX/LINUX

Linux is a Multiuser Multitasking system with a full set of UNIX-compatible tools. Network compatible Main design goals are speed, efficiency, flexibility and standardization.

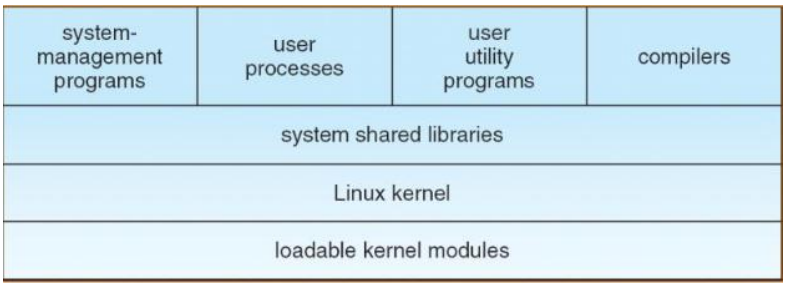


Figure 3 Design principle of Unix/Linux

3.1. LOGGING IN AND LOGGING OUT TO THE SYSTEM

Log in and log out in the system

The process of establishing the communications session and identifying yourself is known as "logging in." Several people can be using a UNIX-based computer at the same time. In order for the system to know who you are and

what resources you can use, you must identify yourself. In addition, since UNIX expects to communicate with you over a terminal (or a PC running terminal-emulation software). UNIX When you first connect to a UNIX system, you usually see a prompt such as the following.

```
login:
```

To log in

- Have your user id (user identification) and password ready. Contact your system administrator if you don't have these yet.
- Type your user id at the login prompt, and then press ENTER. Your user id is case sensitive, so be sure you type it exactly as your system administrator instructed.
- Type your password at the password prompt, then press ENTER. Your password is also case-sensitive.
- If you provided correct user id and password then you would be allowed to enter into the system. Read the information and messages that come up on the screen something as below.

```
login : amrood amrood's

password:

last login: Sun Jun 14 09:32:32 2009 from 62.61.164.73

$
```

You would be provided with a command prompt (sometime called $ prompt) where you would type your all the commands. For example, to check calendar you need to type Cal command as follows.

```
$ cal

 June 2009

Su Mo Tu We Th Fr Sa

 1 2 3 4 5 6
```

```
7 8 9 10 11 12 13

14 15 16 17 18 19 20 21

22 23 24 25 26 27

28 29 30

$
```

Logging Out

When you are done using the system, you should log out. This will prevent other people from accidentally or intentionally getting access to your files. The normal way to log out from almost any shell is to type **exit**. This causes your shell to exit, or stop running. When you exit from your login shell, you log out. If you are using **csh**, you can also type logout; if you are in a login shell, then **csh** will log out. When you finish your session, you need to log out of the system to ensure that nobody else accesses your files while masquerading as you.

To logout

Just type logout command at command prompt, and the system will clean up everything and break the connection.

3.2. CONFIGURING THE ENVIRONMENT AND MANAGING THE PASSWORD

CONFIGURE THE USER OWN ENVIRONMENT

In order to make using the shell easier and more flexible, UNIX uses the concept of an environment. Your environment is a set of values. You can change these values, add new values, or remove existing ones. These values are called environment variables--environment because they describe or define your environment, and variables because they can change.

CHANGE AND MANAGE PASSWORD

All UNIX systems require passwords to help ensure that your files and data remain your own and that the system itself is secure from hackers and crackers. Here are the steps to change your password

- To start, type passwd at command prompt as shown below.
- Enter your old password the one you're currently using.
- Type in your new password. Always keep your password complex enough so that nobody can guess it. But make sure, you remember it.
- You would need to verify the password by typing it again.

```
$ passwd
Changing password for amrood
(current) Unix password: ******
New UNIX password: *******
Retype new UNIX password: *******
passwd: all authentication tokens updated successfully $
```

I have put stars (*) just to show you the location where you would need to enter the current and new passwords otherwise at your system, it would not show you any character when you would type.

3.3. UNIX/LINUX MANUAL SYSTEM

One of the most important things to know about UNIX or any computer system is how to get help when you don't know how to use a command. Many commands will give you a usage message if you incorrectly enter the command. This message shows you the correct syntax for the command. This can be a quick reminder of the arguments and their order.

USE UNIX/LINUX HELP SYSTEM (MANUAL SYSTEM)

The UNIX command man is a powerful tool that gives you complete online access to the UNIX manuals. In its simplest form, the man command takes one

argument, the name of the command or manual entry on which you need information.
In LINUX/UNIX whenever you need help with a command type “man” followed by the command name.

3.4. UNIX/LINUX FILE SYSTEM AND FILE SYSTEM ORGANIZATION

File System

All data in UNIX is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system. When you work with UNIX, one way or another you spend most of your time working **with** files. This tutorial would teach you how to create and remove files, copy and rename them, create links to them etc.

CHARACTERIZE UNIX/LINUX FILE SYSTEM AND ITS ORGANIZATION

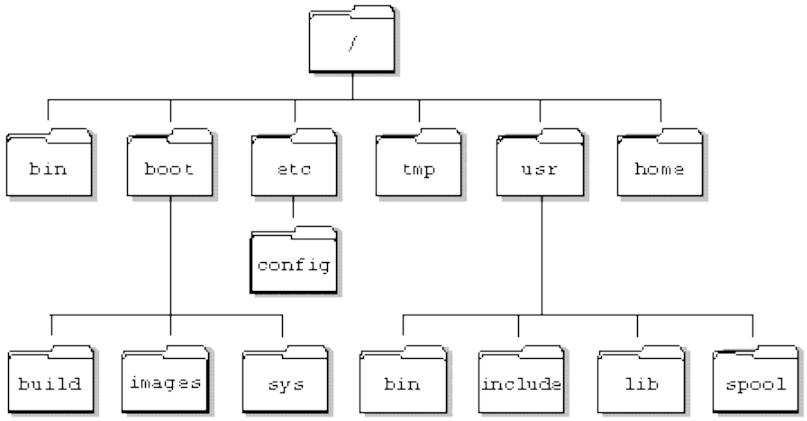
A file can reside on different media. A file can also be a permanent file on disk, a temporary file in the memory, or a file displaying or accepting data from the terminal. If the file is a permanent file, you might be able to view it--but if the file is temporary, you might not know it exists. The functions usually performed on a file are as follows:

- Opening a file for processing
- Reading data from a file for processing
- Writing data to a file after processing
- Closing a file after all the necessary processing has been done

File System Organization

UNIX has provided the directory as a way of organizing files. The directory is a special file under which you can have files or more directories (also referred to as subdirectories). You can visualize the UNIX file structure as a bottom-up tree with the root at the top. Thus, the top-level directory is called the root

directory and is denoted by a single / (forward slash). All the directories and files belong to the root directory. You can also visualize the UNIX file system as a file cabinet in which the file cabinet is the root directory, the individual cabinets are various directories under the root directory, the file folders are the subdirectories, and the files in the individual folders are the files under the directories or subdirectories. Figure shows a typical directory tree structure.



3.5. FILE TYPES, NAMES AND DIRECTORIES

There are various file types available in UNIX. You might be familiar with some of these types of files, such as text documents and source files.

- Ordinary Files − an ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- Directories files − Directories store both special and ordinary files. For users familiar with Windows or Mac OS, UNIX directories are equivalent to folders.
- Special Files − some special files provide access to hardware such as hard drives, CDROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Name correctly file and directory

Each file is identified by a name, which is a sequence of characters. The older versions of UNIX had limitations on the numbers of characters that could be used in a filename. All the newer versions of UNIX have removed this limitation. You should be careful when naming the files, though. Although UNIX allows most characters to be used as part of the filename, some of the characters have special meaning in UNIX and can pose some problems.

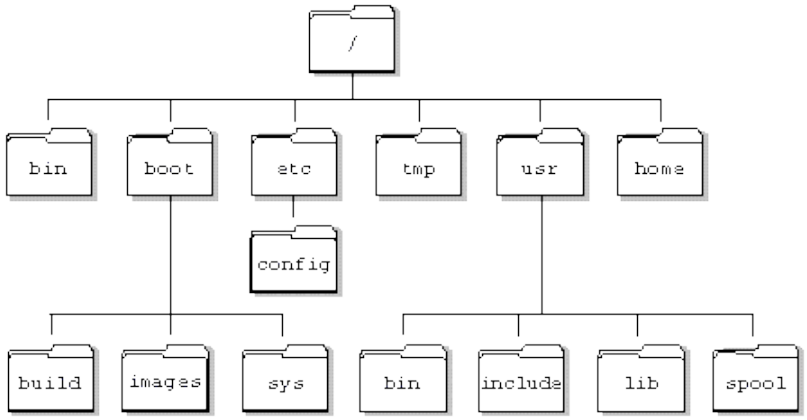


Figure 4 Directory structure and Names

Important Directories

1. **/:** The root directory, the top-level directory in the FHS (File system Hierarchy) or Directory structure. All other directories are sub directories of root. Every single file and directory start from the root directory. Only root user has privilege under this directory.
2. **/bin:** user binaries This contains files that are essential for correct operation of the system. These are available for use by all users.
   - Contains binary executables.
   - Common Linux commands you need to use in single-user mode are located under this directory.
   - Commands used by all user of the system are located here. For example: Ps, ls, ping, grep, cp.
3. **/sbin:** system binaries Sbin mean's system binaries which are only expected to be used by the super user.

- Also Contains executables
- Linux commands located under this directory are used typically by system administrator.
- For example: iptable, reboot, fdisk, ipconfig etc.

4. **/mnt:** mount Directory Provides a location for mounting devices such as remote file system and removable media.
5. **/home:** home directories This is where user home directories are stored.
   - Home directories for almost all user to store their personal files
   - For example: /home/ABC
6. **/var:** Variable files This directory is used to store files which change frequently, and must be available to be written to:
   - Var stands for variable file
   - This includes system log file(/var/log);
   - packages and database files(/var/lib);
   - Email (/var/email); print queues (/var/spool).
7. **/tmp:** temporary files Directory that contains temporary files created by system and users, files under this directory are deleted when system is rebooted.
8. **/etc:** configuration files Various system configuration files are stored here.
   - This also contain startup and shutdown shell scripts used to start/stop individual programs.
   - For example: /ete/resotv.conf /etc/logrotate.conf
9. **/dev:** device file This contains various devices as file, e.g. hard disk, CD ROM drive etc.
   - Contains device files
   - These include terminal device, USB, or any device attached to the system.
   - For example: /dev/tty /dev/usbmon0.
10. **/boot:** boot loader files Includes Linux startup files, including the Linux kernel.
11. **/usr:** user programs Contain user documentation, games, graphical files, libraries(lib) etc.

12. **/proc:** process information This is a pseudo file system contains information about running process. (E.g., proc/pid) it contains information about system process.
13. **/lib:** system libraries Contains libraries files that supports the binaries located under /bin and /sbin. Library file name are either id* or lib*.so*
14. **/svr:** service data Contains server specific services related data, e.g., /srv/cvs
15. **/opt:** optional add-on applications Opt stands for optional, contains add-on applications from individual vendors. Add-on applications should be installed under either /opt or /opt/sub-directory.
16. **/media:** Removable media device Temporary mount directory for removable device. For example: media/cdrom for CDROM Media/floppy for FLOPPY DRIVE Media/cdrecoder for CDWriter.

3.6. MANAGING DIRECTORIES: FILE AND DIRECTORY PERMISSIONS

CREATING AND DELETING FILE AND DIRECTORY

Creating Files:

You can use vi editor to create ordinary files on any Unix system. You simply need to give following command.

```
$ vi filename
```

Above command would open a file with the given filename. You would need to press key I to come into edit mode. Once you are in edit mode you can start writing your content in the file as below.

```
This is Unix file.... I created it for the first time....

I'm going to save this content in this file.
```

Once you are done, do the following steps

Press key esc to come out of edit mode.

Press two keys Shift + ZZ together to come out of the file completely.

Now you would have a file created with filename in the current directory.

```
$ vi filename
$
```

Deleting File To delete an existing file use the rm command. Its basic syntax is.

```
$ rm filename
```

It may be dangerous to delete a file because it may contain useful information. So be careful while using this command. It is recommended to use -i option along with **rm** command.

Following is the example which would completely remove existing file filename:

```
$ rm filename
$
```

You can remove multiple files at a tile as follows.

```
$ rm filename1 filename2 filename3
$
```

Creating Directories

Directories are created by the following command

```
$mkdir dirname
```

Here, directory is the absolute or relative pathname of the directory you want to create. For example, the command

```
$mkdir mydir

$
```

Creates the directory mydir in the current directory. Here is another example –

```
$mkdir /tmp/test-dir

$
```

This command creates the directory test-dir in the /tmp directory. The mkdir command produces no output if it successfully creates the requested directory. If you give more than one directory on the command line, mkdir creates each of the directories. For example

```
$mkdir docs pub

$
```

Creates the directories docs and pub under the current directory.

Removing Directories

Directories can be deleted using the rmdir command as follows

```
$rmdir dirname

$
```

Note − To remove a directory make sure it is empty which means there should not be any file or sub-directory inside this directory. You can create multiple directories at a time as follows

```
$rmdir dirname1 dirname2 dirname3

$
```

Above command removes the directories dirname1, dirname2, and dirname2 if they are empty. The rmdir command produces no output if it is successful.

Changing Directories

You can use the cd command to do more than change to a home directory: You can use it to change to any directory by specifying a valid absolute or relative path. The syntax is as follows

```
$cd dirname

$
```

Here, dirname is the name of the directory that you want to change to. For example, the command

```
$cd /usr/local/bin

$
```

Changes to the directory /usr/local/bin. From this directory you can cd to the directory /usr/home/amrood using the following relative path

```
$cd ../../home/amrood

$
```

Renaming Directories

The mv (move) command can also be used to rename a directory. The syntax is as follows

```
$mv olddir newdir

$
```

You can rename a directory mydir to yourdir as follows

```
$mv mydir yourdir

$
```

SET PERMISSION ON FILE AND DIRECTORY

File ownership is an important component of UNIX that provides a secure method for storing files. Every file in UNIX has the following attributes

- **Owner permissions**: the owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions**: the group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions**: the permissions for others indicate what action all other users can perform on the file.

The Permission Indicators

While using ls -l command it displays various information related to file permission as follows

```
$ls -l /home/amrood

-rwxr-xr-- 1 amrood users 1024 Nov 2 00:10 myfile

drwxr-xr--- 1 amrood users 1024 Nov 2 00:10 mydir
```

Here first column represents different access mode ie. permission associated with a file or directory. The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x)

- The first three characters (2-4) represent the permissions for the file's owner. For example -rwxr-xr-- represents that owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example -rwxr-xr-- represents that group has read (r) and execute (x) permission but no write permission.

- The last group of three characters (8-10) represents the permissions for everyone else. For example -rwxr-xr-- represents that other world has read (r) only permission.

File Access Modes

The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the read, write, and execute permissions, which are described below.

- **Read:** Grants the capability to read ie. view the contents of the file.
- **Write:** Grants the capability to modify, or remove the content of the file.
- **Execute:** User with execute permissions can run a file as a program.

Directory Access Modes

Directory access modes are listed and organized in the same manner as any other file. There are a few differences that need to be mentioned:

- **Read:** Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.
- **Write:** Access means that the user can add or delete files to the contents of the directory.
- **Execute:** Executing a directory doesn't really make a lot of sense so think of this as a traverse permission.

A user must have executed access to the bin directory in order to execute ls or cd command.

Changing Permissions

To change file or directory permissions, you use the chmod (change mode) command. There are two ways to use chmod: symbolic mode and absolute mode.

Using chmod in Symbolic Mode

The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table.

Chmod operator	Description
+	Adds the designated permission(s) to a file or directory.
-	Removes the designated permission(s) from a file or directory.
=	Sets the designated permission(s).

Here's an example using testfile. Running ls -1 on testfile shows that the file's permissions are as follows

```
$ls -l testfile

-rwxrwxr-- 1 amrood users 1024 Nov 2 00:10 testfile
```

Then each example chmod command from the preceding table is run on testfile, followed by ls -l so you can see the permission changes.

```
$chmod o+wx testfile
$ls -l testfile
-rwxrwxrwx 1 amrood users 1024 Nov 2 00:10 testfile
$chmod u-x testfile
$ls -l testfile
-rw-rwxrwx 1 amrood users 1024 Nov 2 00:10 testfile
$chmod g=rx testfile
$ls -l testfile
-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

Here's how you could combine these commands on a single line:

```
$chmod o+wx, u-x, g=rx testfile
$ls -l testfile
-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile
```

Using chmod with Absolute Permissions

The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set. Here's an example using testfile. Running ls -1 on testfile shows that the file's permissions are as

Follows.

```
$ls -l testfile
-rwxrwxr-- 1 amrood users 1024 Nov 2 00:10 testfile
```

Then each example chmod command from the preceding table is run on testfile, followed by ls -l so you can see the permission changes.

```
$ chmod 755 testfile
$ls -l testfile
```

```
-rwxr-xr-x 1 amrood users 1024 Nov 2 00:10 testfile
$chmod 743 testfile
$ls -l testfile
-rwxr---wx 1 amrood users 1024 Nov 2 00:10 testfile
$chmod 043 testfile
$ls -l testfile
----r---wx 1 amrood users 1024 Nov 2 00:10 testfile
```

Changing Owners and Groups

While creating an account on Unix, it assigns a owner ID and a group ID to each user. All the permissions mentioned above are also assigned based on Owner and Groups. Two commands are available to change the owner and the group of files.

- chown: The chown command stands for "change owner" and is used to change the owner of a file.
- chgrp: The chgrp command stands for "change group" and is used to change the group of a file.

Changing Ownership

The chown command changes the ownership of a file. The basic syntax is as follows

```
$ chown user filelist
```

The value of user can be either the name of a user on the system or the user id (uid) of a user on the system. Following example

```
$ chown amrood testfile

$
```

Changes the owner of the given file to the user amrood. NOTE: The super user, root, has the unrestricted capability to change the ownership of an any file but normal users can change only the owner of files they own.

Changing Group Ownership

The chrgp command changes the group ownership of a file. The basic syntax is as follows −

```
$ chgrp group filelist
```

The value of group can be the name of a group on the system or the group ID (GID) of a group on the system. Following example:

```
$ chgrp special testfile
$
```

Multiple Choice Questions

Q.1: Which of the following best describes Linux/Unix?

(a) An operating system designed for mobile devices

(b) An open-source operating system based on Unix

(c) A proprietary operating system developed by Microsoft

(d) A programming language used for web development

Q.2: What is the shell in Linux/Unix?

(a) A graphical user interface (GUI) for Linux/Unix

(b) A command interpreter that provides an interface to the operating system

(c) A virtualization software used to run multiple operating systems simultaneously

(d) A file system used to organize and store data in Linux/Unix

Q.3: Which command is used to display the current working directory in Linux/Unix?

(a) ls (b) mkdir

(c) cd (d) pwd

Q.4: What is the purpose of the chmod command in Linux/Unix?

(a) To create a new directory

(b) To change the ownership of a file or directory

(c) To change the permissions of a file or directory

(d) To remove a file or directory

Q.5: Which command is used to list the contents of a directory in Linux/Unix?

(a) ls (b) cp

(c) mv (d) rm

Q.6: Maximum number of significant characters checked for password.?

(a) 6 (b) 7

(c) 8 (d) 9

Q.7: The process of establishing the communications session and identifying is _____?

(a) Logging-out (b) Logging-in

(c) Sockets (d) Environment

Q.8: MKDIR is used for _____ Directory?

(a) Creating (b) Removing

(c) Renaming (c) Copy

Q.9: RMDIR is used for _____ Directory?

(a) Creating (b) Removing

(c) Renaming (c) Copy

Q.10: A _____is a file created to do inter-process communication?

(a) Regular File (b) Directory File

(c) Named Pipe (d) Character File

ANSWER KEY

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

Short Questions

1. How to Login to a UNIX system?
2. How to Logout of a UNIX system?
3. What type of function can be performed in file?
4. Draw directory tree of files?
5. List down the file's types in UNIX?
6. List down Characters used as part of the UNIX filenames?
7. Write Command to create directory?
8. Write Command to remove directory?
9. How many types of permission are there? Write them down?
10. What is the purpose of Directory name /bin?

Long Questions

1. How to configure environment in UNIX?
2. Explain File system in UNIX.
3. Write a detail note on file Types.
4. What do you know about naming files and directories?
5. Write a note on file system organization?

Bibliography

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar

CHAPTER 04 UNIX/LINUX BASIC COMMANDS

Objectives
After completion of this chapter students will be able to:

4.1 User-Related, Locating and Search
4.2 Usage Determination and Process-related Commands
4.3 File and Directory Manipulation
4.4 File Content and File Content Search
4.5 Printing and Scheduling
4.6 Storage and Status
4.7 Miscellaneous Commands

INTRODUCTION

UNIX has commands that are native to UNIX, and commands that you (or someone at your installation) can write. Depending on the UNIX version you are running and the shell you are in, the commands differ. In this chapter, commands that are available in most UNIX versions are discussed. Each command can have several arguments and several flags associated with it. The general form of a command is

```
command [flags] [argument1] [argument2] ...
```

Flags are preceded by a hyphen. Several flags can be specified together with only one hyphen. For example, the following two commands are equivalent:

```
ls -a -l
```

```
ls -al
```

Depending on the command, the arguments can be optional or mandatory. All commands accept inputs from the standard input, display output on standard output and display error message on standard error. You can use UNIX redirection capabilities to redirect one or more of these. Standard input is where UNIX gets the input for a command, standard output is where UNIX displays output from a command, and standard error is where UNIX displays any errors as a result of the execution of a command.

All commands, when executed successfully, return a zero-return code. However, if the commands are unsuccessful or partially successful, they return non-zero return codes. The return codes can be used as part of control logic in shell scripts.

4.1. USER-RELATED, LOCATING AND SEARCH

The commands related to logging in and out of the system are discussed in the following sections. User Related commands are those that are used to log in and out of your UNIX system. They might differ slightly from system to system, but the commands discussed here are common to most systems.

Login

When you start working on a UNIX system, the first thing you need is a login ID. Each user in a UNIX system has a unique login ID that identifies the user and any characteristics associated with the user. When you first connect to a UNIX system, you get the login prompt. The login prompt usually asks for the following information:

```
login:
password:
```

You will be allowed into a system, if, and only if, you enter the login ID and password correctly. When the login is successful, you get information, such as

last unsuccessful login, last successful login, whether you have any mail, messages from the system administrator, and more. Following is an example of a successful login:

```
**************************************************************
* *
* You are now logged on to the host1 computer *
* *
**************************************************************
Last unsuccessful login: Thu Nov 7 22:32:41 1996 on tty1
Last login: Fri Nov 8 01:17:04 1996 on tty1
/u/testuser:>>
```

While logging in, there are shell scripts that are executed. The script that gets executed depends on the shell you are running. For example, in Bourne shell, the. profile file in your home directory is execute, in Korn shell, the file pointed to by the environment variable ENV is executed. You should put in as part of the startup file all the commands that you want to be executed before login. For example, you can set the search path, the terminal type, and various environment variables or run special programs, depending on your requirements.

You can invoke the login command from the command line also. The user is optional. If user is not provided, the system prompts for login id followed by the password prompt, if needed. If user is provided, the system prompts for a password, if needed.

INVOKE CORRECTLY UNIX/LINUX COMMANDS USED IN USER AND USER-DETERMINATION AND FILE SEARCH

Rlogin

UNIX provides you with the rlogin command to let you move between various computers in a network. The rlogin command is similar to the telnet command described in this section.

To be allowed access to a remote host, you must satisfy the following conditions:

1. The local host is included in the ***/etc/hosts.equiv*** file of the remote host, the local user is not the root user and the -l User option is not specified.
2. The local host and user name are included in the ***$HOME/.rhosts*** file in the remote user account.

If the user on the remote host is set up without a password, rlogin will allow you to log in without using any password, provided the above defined conditions are satisfied. However, it is not advisable to set up users without passwords. When you exit from the remote host, you are back on the local host.

Telnet

If you are in an environment where you work with multiple UNIX computers networked together, you will need to work on different machines from time to time. The telnet command provides you with a facility to login to other computers from your current system without logging out of your current environment. The telnet command is similar to the rlogin command described earlier in this section.

The hostname argument of telnet is optional. If you do not use the host computer name as part of the command, you will be placed at the telnet prompt, usually, ***telnet>.*** There are a number of sub-commands available to you when you are at the ***telnet>*** prompt. Some of these sub-commands are as follows:

- **exit** to close the current connection and return to the **telnet>** prompt if sub-command open was used to connect to the remote host. If, however, telnet was issued with the host-name argument, the connection is closed and you are returned to where you invoked the telnet command.
- **display** to display operating arguments.
- **open** to open a connection to a host. The argument can be a host computer name or address. telnet will respond with an error message if you provide an incorrect name or address.
- **quit** to exit telnet.
- **set** to set operating arguments.
- **status** to print status information.

- **toggle** to toggle operating arguments (toggle ? for more).
- **?** to print help information.

Passwd

As you have seen, every time you try to log in to a computer system, you are asked for your user ID and password.

Although users can be set up without passwords, most users will have a password, which they will must use when logging in to a computer system.

When you first get you user ID set up in a computer, the system or security administrator will assign you a temporary password using the root user ID. The first time you try to log in, the system will ask you to change your password; you will use the new password for all subsequent logins.

However, you can change your password if, for example, you think that somebody else has come to know it. As a security precaution, it is a good practice to modify your password frequently. You can use the ***passwd*** command to change your password.

When you issue the ***passwd*** command, you are first prompted for your current password, followed by two prompts to enter your new password. The new password must match on both entries. This is a precaution to ensure that you do not type in something you did not intend and are subsequently unable to login. Your new password cannot be the same as you current password. When you are typing your password (both old and new), UNIX does not display them. Whenever you change your password, the new password must follow the rules set up at your installation. Some of the rules that govern passwords are as follows:

- The minimum number of alphabetic characters.
- The maximum number of times a single character can be used in a password.
- The minimum number of weeks that must elapse before a password can be changed.
- The maximum number of weeks after which the password must be changed. The system will prompt you to modify the password when this happens.

Some of the UNIX system have additional flags for the ***passwd*** command. You should be careful about choosing passwords. If you choose passwords that other people can make an educated guess about, your login will not be secure. Following are some guidelines for choosing passwords:

- Avoid using proper nouns like name of spouse, children, city, place of work, and so on.
- Avoid using strings of characters followed by numbers, like xyz01.
- Use both uppercase and lowercase letters mixed with numbers.
- The length should be at least 7 characters.
- You should never write down your password.
- You should be able to type it quickly.

Exit

When you log in to a UNIX system, you are always placed in a shell. The shell may be the Bourne shell, C shell, Korn shell, or other. As we have seen, we can log in to other system with rlogin or telnet commands. The exit command allows you to exit the current shell.

You can also exit your current shell by typing CTRL-d. (Hold down the Ctrl key and the d key together.) If you are on the command line and press Ctrl-d, you will be logged off.

Locating Commands

When you try to execute a command, UNIX has to locate the command before it can execute it. UNIX uses the concept of search path to locate the commands you are trying to execute. The search path is a list of directories in the order to be searched for locating commands.

The default search paths set by the installation usually have the standard directories like ***/bin, /usr/bin***, and other installation specific directories. You can modify the search path for your environment as follows:

- Modify the PATH statement in the .profile file (Korn shell and Bourne shell).
- Modify the set path=(....) in the .cshrc or .login file (C shell)

Which Command

This command can be used to find whether a particular command exists in you search path. If it does exist, which tells you which directory contains that command.

Whence Command

The whence command is a more verbose form of the ***which*** command in Korn shell. It has a flag ***-v*** associated with it to produce output in a verbose form. For a native command that has not been aliased, it generates output similar to ***which***.

Where Command

The where command is used to obtain the full pathname of one or more files or directories. There are no flags associated with the where command. When the pathname is displayed, it is prefixed by the hostname, and if there are no arguments, the full pathname of the current directory is displayed.

4.2. USAGE DETERMINATION AND PROCESS-RELATED COMMANDS

Usage Determination Command

UNIX provides you with online help to learn about various commands and their options and flags. You may be familiar with the most often used commands and their various options, but to find out about less popular commands or command usage, you can use the online help provided by UNIX.

Man Command

The man command is used to display the online UNIX manual pages, which include commands, files, sub-routines, and so on. You have to provide the man command with the name of the object you are looking for. If you do not know the full name, you can use the UNIX wildcard to specify the object name. You

can even find out more about the ***man*** command itself by using the ***man*** command.

The following are some of the flags and arguments that can be used for the man command:

- **-k** keyword for a list of summary information of manual sections in the keyword database for the specified keyword.
- **-f** command for details associated with the command. The root user must set up the file **/usr/man/whatis** before you can use this option.
- **-M** path to specify the search path for the man command.

The following is a list of sections that can be specified:

1. Commands
2. System calls
3. Subroutines
4. File formats
5. Miscellaneous
6. Special files
7. Maintenance

Process Related Commands

In UNIX, a process is a program that has its own address space. Usually, a command or a script that you can execute consists of one or more processes. Simple commands like unmask have only one process associated with them, while a string of commands connected by pipes has multiple processes associated.

The processes can be categorized into the following broad groups:

- Interactive processes, which are those executed at the terminal. The interactive processes can execute either in foreground or in background. In a foreground process, the input is accepted from standard input, output is displayed to standard output, and error messages to standard error. While executing a process in the background, the terminal is detached from the process so that it can

be used for executing other commands. It is possible to move a process from foreground to background and vice versa.

- Batch processes are not submitted from terminals. They are submitted to job queues to be executed sequentially.
- Daemons are never-ending processes that wait to service requests from other processes.

In UNIX, each process has a number of attributes associated with it. The following is a list of some of these attributes:

- Process ID is a unique identifier assigned to each process by UNIX. You can identify a process during its life cycle by using process ID.
- Real User ID is the user ID of the user who initiated the process.
- Effective User ID is the user ID associated with each process. It determines the process's access to system resources. Under normal circumstances, the Real User ID and Effective User ID are one and the same. But it is possible for the effective user ID to be different from the real user ID by setting the Set User ID flag on the executable program. This can be done if you want a program to be executed with special privilege without actually granting the user special privilege.
- Real Group ID is the group ID of the user who initiated the process.
- Effective Group ID is the group ID that determines the access rights. The effective group ID is similar to the effective user ID.
- Priority (Nice Number) is the priority associated with a process relative to the other processes executing in the system.

INVOKE CORRECTLY UNIX/LINUX COMMANDS USED IN ADMINISTRATION OF OWN ENVIRONMENT AND PROCESS-RELATED JOB

Kill Command

The kill command is used to send signals to an executing process. The process must be a non-foreground process for you to be able to send a signal to it using this command.
The flags associated with the kill commands are as follows:

- **-l** to obtain a list of all the signal numbers and their names that are supported by the system.
- **-signal number** is the signal number to be sent to the process. You can also use a signal name in place of the number. The strongest signal you can send to a process is 9 or kill.

Nice Command

As you have seen, a relative priority is associated with each process. The relative priority governs the resources allocated to it by the operating system. The ***nice*** command lets you modify the priority of one or more processes so that you can assign them higher or lower priority. You can increase the priority only if you have root privileges though.

A negative number signifies a higher priority than a positive number. The value is usually in the range of -20 to 20. If you do not specify an increment, the nice command takes a value equal to or higher than the current process, provided you have appropriate privileges. If you do not have sufficient privileges, the priority is not affected at all. Usually, you should use the nice command to lower the priority of background or batch processes for which you do not need fast turnaround.

PS Command

The **ps** command is used to find out which processes are currently running. Depending on the options, you can find all processes or only those initiated by you user ID. This command provides you with details about the various background and batch processes running in the system. It can provide information only about the active processes.

When the **ps** command is executed without any flags or arguments, it lists all processes (if any) initiated from the current terminal.

Following is a list of some of the flags that determine what processes are listed by the **ps** command:

- **-A** to list details of all processes running in the system.
- **-e** to list details of all processes, except kernel processes.
- **-k** to list all the UNIX kernel processes.

- **-p** list to list details of all processes specified in the list.
- **-t** list to list details of all processes initiated from the terminals specified in the list.
- **-U** list (-u list) to list details of all processes initiated by the users specified in the list.
- **a** to list details of all processes that have terminals associated wit them.
- **g** to list details of all processes.
- **x** to list details of all processes that do not have any terminal associated with them.

The following is a list of some of the flags for the ps command that determine which details are displayed for each process listed:

- **-l** to generate the listing in a long form
- **-f** to generate a full listing
- **F** format to generate a formatted output

The following details are displayed if formatting flags are not used:

- Process ID of the process.
- Terminal ID associated with the process. Hyphen (-), if no terminals are associated.
- CPU time consumed by the process.
- Command being executed as part of the process.

By using the formatting command, some of the details that can be obtained are as follows:

- USER ID of the user who initiated the process.
- Process ID of the process.
- Parent Process ID of the process.
- CPU utilization of the process.
- Start time of the process. If the process has been started on the same day, it shows the time; otherwise, it shows only the date.
- Terminal ID associated with the process. It display a hyphen (-) if there are no terminals associated with the process.

- CPU time consumed by the process.
- Commands being processed as part of the process.

Jobs Command

In UNIX, there is a subtle difference between processes and jobs. A job typically is one command line of commands, which can a single command, a shell script, or a chain of piped commands. In a chain of piped commands, each command has a unique process ID, but all have the same job ID.

The C shell and some versions of Korn and Bourne shell offer the jobs command. You can use the jobs command to find out the details about active jobs. Once you have the job ID, you can start using it to do primitive job controls.

You can use ***%*** (percent sign) in front of the job number to indicate that the number is a job number rather than a process ID.

Wait Command

You can use the wait command to wait for completion of jobs. This command takes one or more process IDs as arguments. This is useful while doing shell programming when you want a process to be finished before the next process is invoked. If you do not specify a process ID, UNIX will find out all the processes running for the current environment and wait for termination of all of them.

Nohup Command

When you are executing processes under UNIX, they can be running in foreground or background. In a foreground process, you are waiting at the terminal for the process to finish. Under such circumstances, you cannot use the terminal until the process is finished. You can put the foreground process into background as follows:

```
Ctrl-z
bg
```

The processes in UNIX will be terminated when you logout of the system or exit the current shell whether they are running in foreground or background. The only way to ensure that the process currently running is not terminated when you exit is to use the ***nohup*** command.

Sleep Command

If you want to wait for a certain period of time between executions of commands, use the sleep command. This can be used in cases where you want to check for, say, the presence of a file, every 15 minutes. The argument is specified in seconds.

4.3. FILE AND DIRECTORY MANIPULATION

File Manipulation Commands

Several commands that can be used to manipulate various attributes of one or more files, as well as to copy and move files from one location to another. The various attributes that can be manipulated include modification time, permission, and more.

USE CORRECTLY UNIX/LINUX COMMANDS TO MANIPULATE FILES AND DIRECTORIES

Touch Command

The touch command can be used for a number of purposes depending on whether a file already exists. If a file does not exist, the touch command will create it if you have written access to the directory. If a file is already present, the ***touch*** command modifies the last modification time of the file.

Chmod Command

You may need to modify the permission of a directory or files either to secure them or to make them accessible to others. You can use the chmod command to modify the permission or files and directories. The permission in UNIX is specified in octal number (0 thorough 7). Permission for a file or directory can be specified for the following:

- Owner--The user who created the file
- Group--The group to which the owner belongs

- World or others--Users other than the owner and users in the group to which the owner belongs

For each of these, one octal number is specified to designate the permission. The permission for the owner, group, and world is derived on the basis of three bits associated with read, write, and execute authority for the file. That is, the bit for read will have a value of one if read permission is to be granted, the bit for write will have a value of one if write permission is to be granted, and the bit for execute will have a value of one if execute permission is to be granted.

Chgrp Command

If you want to change the group to which the file belongs, use the ***chgrp*** command. The group must be one of the groups to which the owner belongs. That is, the group must be either the primary group or one of the secondary groups of the owner.

Chown Command

In case you want to change the owner of a file or directory, use the ***chown*** command.

Rm Command

Once you are done using a file and you do not want to use it any more, then you would like to remove the file so as to regain the space utilized by the file. The ***rm*** command lets you do this by removing files permanently from the disk. If an entry is the last link to a file, the file is deleted. To remove a file from a directory, you do not need either read or write permission to the file, but you do need write permission to the directory containing the file. The rm command is usually used to remove files, but it provides a special flag -r to remove files in a directory recursively, including the directory and its sub-directories.

Following is a list of some of the flags that can be used with the rm command:

- **-i** to interactively remove the files.
- **-f** to remove the files without any messages. This will not generate any messages for cases where a file does not exist or you do not have permission to remove one or more files.
- **-r** to remove files within a directory and directories themselves recursively.

Mv Command

If you are not satisfied with a filename, you may wish to name the file differently. The mv command will let you do that. In addition, it allows you to move files from one directory to another retaining the original filename, which is equivalent to copying the files from the source directory to the destination directory and then removing the file from the source directory. You may do that if you are reorganizing your files. While moving files or directories, the target file or directory gets the permission of the source file or directory, irrespective of whether the target file or directory already exists or not.
Following is a list of some of the flags that may be used with the ***mv*** command:

- **-i** to move or rename files interactively.
- **-f** to move or rename files without any messages. Use of this flag will suppress messages when you are trying to rename a nonexistent file or you do not have permission to rename a file.

The ***mv*** commands takes two arguments. The first argument is the source file or directory name, and the second argument is the destination file or directory. However, the behavior of the mv command depends on whether the destination file or directory name exists. If you move files within the same file system, all links to other files are retained. But if you move the files across file systems, the links are not retained.

Cp Command

The **cp** command can be used to make a copy of the contents of one or more source files as specified target files. If the target file already exists, it is overwritten with the contents of the source file. The cp command behavior varies depending on whether the source and the target are files or directories. The following is a list of some of the flags that can be used with the ***cp*** command:

- **-p** to retain the modification date and time as well as permission modes of the source file.
- **-i** to execute the copy command in an interactive mode so that it asks

for confirmation if the target file exists.

- **-h** to follow the symbolic links.
- **-r** to copy files under the specified directories and their sub-directories.

Directory Manipulation Commands

When you are set up as a user in a UNIX operating system, you usually are set up with the directory /u/username as your home directory. To organize your files, you must set up directories of your liking. Here we will present the commands to create and remove directories.

Mkdir Command

To create a directory, use the ***mkdir*** command. The ***mkdir*** command accepts multiple directory names for creation at the same time. As you did with the files, use relative pathname or absolute pathname to create a directory. To create a directory, you must have write permission for its parent directory, and UNIX uses the current permission setting (refer to the ***umask*** command) to set the permission for the directory.

The following is a list of the flags that can be used with the ***mkdir*** command:

- -p to create all the directories in the part name of the specified directory if they do not exist.
- -m permission to specify permission for the directory to be created.

Rmdir Command

Once you are done with a directory or you run out of space and want to remove a directory, use the ***rmdir*** command. You can remove a directory only if it is empty, that is, all the files and directories in it have been removed. You can specify multiple directory names as arguments to rmdir command. To remove a directory, you must have written permission to the parent directory.

The following is a flag that can be used with **rmdir** command:

- -p to remove all the directories in the specified pathname.

4.4. File Content and File Content Search

File Content Commands

Commands that can be used to look at the contents of the file or parts of it. You can use these commands to look at the top or bottom of a file, search for strings in the file.

USE CORRECTLY UNIX/LINUX COMMANDS TO SEARCH AND VIEW AT CONTENTS OF FILES

More Command

The more command can be used to display the contents of a file one screen at a time. By default, the more command displays one screen worth of data at a time. However, the number of lines displayed can be modified. The more command pauses at the end of display of each page. To continue, press a space bar so that the next page is displayed or press the Return or Enter key to display the next line. Mostly the more command is used where output from other commands is piped into the more command for display. Following is a list of flags that can be used with the more command:

- **-d** for more to prompt to quit, continue, or get help.
- **-f** to count logical lines in the file.
- **-number** to set the size of the display window to number.

Less Command

The less command is one more in the family of commands to view the contents of a file. This may not be available by default on all UNIX systems. It behaves similarly to the more command. The less command allows you to go backward as well as forward in the file by default.

The following is a list of sub-commands that can be used once you are in the less command:

- **h** to display a list of the sub-commands that can be used.
- **spacebar or Ctrl-v or Ctrl-f** or f to go forward in the file one screen. If preceded by a number, moves forward by the specified number of lines.

- **return** key or Ctrl-n or Ctrl-e or e or j to move forward by 1 line. If preceded by a number, moves forward by the specified number of lines.

Tail Command

You can use the tail command to display, on standard output, a file starting from a specified point from the start or bottom of the file. Whether it starts from the top of the file or end of the file depends on the parameter and flags used. One of the flags, -f, can be used to look at the bottom of a file continuously as it grows in size. By default, tail displays the last 10 lines of the file. The following is a list of flags that can be used with the tail command:

- -c number to start from the specified character position number.
- -b number to start from the specified 512-byte block position number.
- -k number to start from the specified 1024-byte block position number.
- -n number to start display of the file in the specified line number.
- -r number to display lines from the file in reverse order.
- -f to display the end of the file continuously as it grows in size.

With all these flags, the number you can specify may be a number prefixed by a + (plus sign) or a - (minus sign). If you specify a + the tail command starts processing from the start of the file. If you specify a - or do not specify any sign, tail starts processing from the bottom of the file.

Head Command

The head command displays a file on the standard output. The head command starts from the top of the file and displays the specified number of bytes or lines from the start of the file. By default, head displays 10 lines.
Following are the flags that can be used with the head command:

- -c number to display the number of bytes from the top of the file.
- -n number to display the number of lines from the top of the file.

Wc Command

The wc command counts the number of bytes, words, and lines in specified files. A word is a number of characters stringed together delimited either by a space or a newline character. Following is a list of flags that can be used with the wc command:

- **-l** to count only the number of lines in the file.
- **-w** to count only the number of words in the file.
- **-c** to count only the number of bytes in the file.

Read Command

The read command is used in shell scripts to read each field from a file and assign them to shell variables. A field is a string of bytes that are separated by a space or newline character. If the number of fields read is less than the number of variables specified, the rest of the fields are unassigned.
The following is a list of flags that can be used with read command:

- -r to treat a \(backslash) as part of the input record and not as a control character.

Tee Command

If you want to execute a command and want its output redirected to multiple files in addition to the standard output, use the tee command. The tee command accepts input from the standard input, so it is possible to pipe another command to the tee command.
Following is an optional flag that can be used with the tee command:

- ***-a*** to append to the end of the specified file. The default of the tee command is to overwrite the specified file.

File Content Search Commands

We can use the find command to search for filenames in a directory. For searching for a pattern in one or more files, use the ***grep*** series of commands. The ***grep*** commands search for a string in the specified files and display the output on standard output.

Egrep Command

The ***egrep*** command is an extended version of grep command. This command searches for a specified pattern in one or more files and displays the output to standard output. The pattern can be a regular expression where you can specify special characters to have special meaning, some of which are as follows:

- **.** to match any single character.

- ***** to match one or more single characters that precede the asterisk.
- **^** to match the regular expression at the beginning of a line.
- **$** to match the regular expression at the end of a line.
- **+** to match one or more occurrences of a preceding regular expression.
- **?** to match zero or more occurrences of a preceding regular expression.
- **[ ]** to match any of the characters specified within the brackets.

Fgrep Command

Like ***egrep*** and ***grep, fgrep*** also searches one or more files for a specified string and displays output on standard output. The ***fgrep*** command is supposed to be the faster version of the ***grep*** command but in reality may not be. Please notice that the ***fgrep*** command is used to search for a specified string and not pattern (regular expression where special characters can be used to indicate special meaning). Following is a list of flags that can be used with the fgrep command:

- **-b** to display the block number at the start of each line found.
- **-c** to display the count of lines in which the pattern was found without displaying the lines.
- **-f** filename to specify filename that contains the patterns to be match.

Grep Command

The ***grep*** command can be used to search for a specified pattern in one or more files, and displays the matching output on standard output. Following is a list of flags that can be used with ***grep*** command:

- **-b** to display the block number at the start of each line found.
- **-c** to display the count of lines in which the pattern was found without displaying the lines.

Strings Command

The strings command can be used to search for strings in executable files where a string consists of four or more printable characters terminated by a null or newline. Following is a list of some of the flags that can be used with the strings command:

- **-a** or - to search the entire file, not just the data section.

- **-o** to list each string preceded by its offset in the file (in octal).
- **-Number** to specify minimum string length other than the default of 4.

4.5. PRINTING AND SCHEDULING

Printing Commands

You may have several documents that you want to print, and you may have several printers attached to your computer where you can print. Here we will discuss some of the commands that direct printing of specified documents to specified printers and find out the status of the printers. We will also discuss commands to cancel specified printing jobs.
In a UNIX system, you can have multiple printers but only one of the printers can be set up as the default printer to which all the print requests are sent if a printer name is not specified.

PRINT FILES USING UNIX/LINUX COMMANDS

Cancel Command
If you have earlier queued up requests to print one or more documents using the ***lp*** command and you wish to cancel these requests, use the cancel command. Using the cancel command, you can either cancel a specified job or all queued requests to a specified printer queue. If you are an ordinary user, cancel jobs that have your user ID only. You can either specify one or more job ID, or a printer name with the cancel command.

Lp Command

To print one or more files to a specified printer, use the ***lp*** command. By default, the ***lp*** command accepts input from the standard input. If more than one file is specified, the files are printed in order of their appearance in the command.
The files you are printing should exist until they are printed because the lp command does not make copies of the file while printing (unless you use the -c flag). Following is a list of a commands that can be used with the ***lp*** command:

- ***-c*** to make a copy of the file so that the file can be deleted or modified while the printing is still going on.
- ***-dprintqueue*** to specify the print queue where the print request is to be directed.
- ***-m*** to notify the requesting user upon successful completion of the print request by mail.
- ***-ncopies*** to specify the number of copies to be printed.
- ***-tittle*** to print the specified title on the banner page.

Pr Commands

The ***pr*** command accepts input from the standard input and generates output on the standard output by default. This command formats the output into pages with name of file, date, and time and page numbers. If the line length is larger than the page width, it is truncated. As the ***pr*** command formats and paginates the output, you can pipe the output of the ***pr*** command to a print command like ***lp*** to print the output. Following is a list of some of the flags that can be used with the ***pr*** command:

- ***-d*** generates the output with double spacing.
- ***-f*** or ***-F*** to use a form-feed to a new page instead of a sequence of line-feed characters.
- ***-h*** "heading" to print heading instead of the filename as header on each page.

Lpstat Command

You can use the ***lpstat*** command to display the current status of all line printers. If the ***lpstat*** command is executed without any flags, it displays the status of each printer with the entries queued by the ***lp*** command.
Following is a list of some of the flags that can be used with the ***lpstat*** command:

- **-d** to display the default line printer information.
- **-r** to display status and job information of all queues.
- **-s** to display summary information on all queues.

- **-t** to display detailed status information on all queues.
- **-uusername** to display status of print requests started by specified username.
- **-vprintername** to display a list of specified printer name.

USE CORRECTLY UNIX/LINUX SCHEDULING COMMANDS

UNIX gives you the ability to schedule scripts and commands for execution at a later point in time. You can specify the exact time when the command should be run. UNIX also provides a way of reporting on the scheduled jobs and removing them if you do not want to execute them.

At Command

The ***at*** command allows you to:

- Schedule a command for execution at a specified time.
- Display a list of scheduled jobs.
- Remove jobs from the scheduled jobs list

Following is a list of some of the flags that can be used with the ***at*** command:

- **-l** to display a list of jobs scheduled by you.
- **-m** to mail a report of successful execution of the job.
- **-t** date to schedule a job to be executed at the specified date and time.
- **-r** joblist to remove the jobs specified in the job list from the queue.

Atq Command

The ***atq*** command can be used to list the jobs scheduled at a later time. The jobs are displayed in the order of the time scheduled with earlier scheduled jobs displayed first.

Following is list of flags that can be used with the ***atq*** command:

- **-*c*** to display a list of jobs in order of time at which the ***at*** command was executed to schedule the jobs.
- **-*n*** to display the number of scheduled jobs.

Crontab Command

UNIX systems have a ***deamon*** running all the time that can run jobs at regularly scheduled intervals. You can specify the jobs that the ***crontab*** command will

execute in a file, and the ***cron deamon*** will check it when the ***cron deamon*** is initialized or when additions or modifications are made to the file.
The entries you can make in the ***crontab*** file consist of the following fields (separated by spaces or tab character):

- Minute
- Hour
- Day (of the month)
- Year
- Day of the week
- Command

Following is a list of flags that can be used with the ***crontab*** command:

- **-l** to list your **crontab** file.
- **-e** to edit or create the **crontab** file.
- **-r** to remove your **crontab** file.
- **-v** to list the status of the **crontab** jobs.

4.6. STORAGE AND STATUS

Storage Commands

A number of commands that can be used for file management, that is, to backup files to different media, to restore files from different media, to compress files to save disk space, to un-compress files to restore, and so on.

USE CORRECTLY UNIX/LINUX STORAGE AND STATUS COMMANDS

Compress Command

You can use the compress command to reduce the size of a file. A file created by the compress command has **a. Z** appended to its name. The compressed file retains the permission and time attributes of the original file. Following is a list of flags that can be used with the compress command:

- **-d** to force the compress command to act as a un-compress command.
- **-c** to compress the file to standard output (which can be redirected to another file) so that the original file is intact.
- **-f or -F** to compress the file and overwrite the compressed file if it already exists.
- **-v** to display the compression percentage.
- **-V** to display the current version and compile options.

Cpio Command

You can use the ***cpio*** command to copy files to archival medium from disk or to restore from archival medium to disk. There are three major forms of the ***cpio*** command:

- ***cpio -o*** to read standard input for path names and copy them to standard output.
- ***cpio -i*** to read from standard input archival files and create disk files.
- ***cpio -p*** to read standard input for path name and copy to the specified directory.

Dd Command

The ***dd*** command can be used to read data from standard input and copy it to the standard output after converting the data according to specified conversion parameters. Along with the data conversion, the physical attributes, such as block size, can also be modified by specifying appropriate parameters.

Pcat Command

The ***pcat*** command can be used to un-compress a file to the standard output. This command does not have any flags.

Pack Command

If you want to save space, use the pack command to compress a file in a way similar to the compress command. The ***pack*** command compresses a file and generates a new file with. z appended to the filename. The original file is removed. The amount of space saved depends on the contents of the file. Usually, you can get about 30 percent to 50 percent compression for text files.

By default, the pack command will not compress if it cannot reduce the size of the file.

Uncompress Command

The ***uncompress*** command can be used to ***uncompress*** a file that has earlier been compressed using the compress command. By default, the ***uncompress*** command un-compresses a file in place; that is, the compressed file is deleted and the uncompressed file without the ***.Z*** suffix is created in its place. The uncompressed file retains the permission and modification time attributes of the compressed file, but the user and the group of the file are changed to that of the user uncompressing the file. Following is a list of some of the flags that can be used with the ***uncompress*** command:

- ***-f or -F*** to force the uncompress even though a file by the name of the uncompressed file may already exist.
- ***-c*** to uncompress the specified by file to the standard output retaining the original compressed file.

Unpack Command

The ***unpack*** command can be used to uncompress files that have been compressed using the pack command and have the. z extension. The uncompressed file is created at the same place as the compressed file and the compressed file is removed. The uncompressed file retains attributes, such as the user, group, permission, and access and modification time of the compressed file. The unpack command will not un-compress the file if a file by the name of the uncompressed file already exists.

Status Commands

Commands that display the status of various parts of the system. These commands can be used to monitor the system status at any point in time.

Date Command

You can use the date command to display the current date and time in a specified format. If you are root user, use the date command to set the system date. To display the date and time, you must specify a + (plus) sign followed by

the format. The format can be as follows:

- **%A** to display date complete with weekday name.
- **%b or %h** to display short month name.
- **%B** to display complete month name.
- **%c** to display default date and time representation.
- **%d** to display the day of the month as a number from 1 through 31.

Env Command

The ***env*** command can be used to display the current environment or change one or more of the environment variables and run a specified command. The changes are effective only during the execution of the command. Following is an optional flag that you can use with the ***env*** command:

- ***-i*** to indicate that only the variables setup as part of the ***env*** command are used for the specified command and all the current variable setups are ignored.

Iostat Command

The ***iostat*** command can be used to obtain statistics about CPU, disks and TTY for a system. The first time you run ***iostat*** after the most recent booting of the system, the ***iostat*** provides the statistics since then. After that, ***iostat*** provides statistics since the last execution of the ***iostat*** command.
The ***iostat*** command displays the following details:

- TTY and CPU header
- TTY and CPU statistics detail
- Physical volume header
- One line for each physical volume

Sar Command

You can use the ***sar*** command to report on system information. The ***sar*** command allows you to save the information and report on it. By default, the ***sar*** command generates the CPU utilization reports, but you can use various flags to collect information about other system activities. Following is a list of some of the flags that can be used with the ***sar*** command:

- ***-A*** to report data on all system activities.
- ***-a*** to report data on the usage of file system access routine.
- ***-b*** to report buffer activities.

Uptime Command

The ***uptime*** command displays the following information:

- The current time
- The length of time the system has been up
- The number of users currently logged one
- the number of jobs executing in the system.

Vmstat Command

The ***vmstat*** command can be used to get information about the processes, virtual memory, physical volumes and CPU activity. The information includes the utilization of CPU, virtual memory, and physical volume, which can be used to monitor the load on the system.

The first invocation of ***vmstat*** displays the statistics since the system startup, and subsequent invocations display statistics since the last invocation. You can specify a count and an interval parameter to control the number of reports generated and interval between the reports. The details displayed by ***vmstat*** are as follows:

- Processes
- Virtual memory
- Page
- Faults
- CPU

4.7. MISCELLANEOUS COMMANDS

In this section we will discuss some of the commands available to do miscellaneous operations in UNIX.

USE OTHER MISCELLANEOUS COMMANDS AVAILABLE TO UNIX/LINUX

Banner Command

You can use the banner command to print one or more characters in large size.

Bc Command

If you want to perform simple arithmetic expression in UNIX, use the ***bc*** command. By default, all the numbers are assumed to be decimal numbers, but you can perform operations on octal or hexadecimal numbers. You can also scale the decimal numbers. The ***bc*** command accepts input first from the specified file followed by standard input. You can, however, use input redirection to accept input only from a file.

The arguments that can be used with the ***bc*** commands are as follows:

- Variable name (one letter)
- Variable array name (letter[expression])
- A literal like scale

Some of the other operands that can be used are as follows:

- + For adding
- - For subtracting
- / For division
- * For multiplication
- % For percentage

Cal Command

You can use the ***cal*** command to display the calendar for one or more months on standard output. If you do not specify any arguments, ***cal*** displays the calendar for the current month. You can specify the month and year for which you want to display the calendar. If you specify only one argument, ***cal*** will display a calendar for all 12 months of the specified year.

Calendar Command

You can use the calendar command to get reminders from messages stored in a special file named calendar in the current directory. The messages are stored in the following format:

- date message
- message date

Clear Command

You can use the ***clear*** command to clear the screen of your workstation. This command checks the terminal type to find out the terminal type to determine how to clear the screen.

Time Command

You can use the time command to obtain the execution time of a script, command, or program. The execution time is displayed with the following three times:

- real
- user
- system

Multiple Choice Questions

Q.1: Which of the following is User Related Command?

(a) Whence (b) Kill

(c) bc (d) exit

Q.2: Which of the following is Locating Command?

(a) Whence (b) Kill

(c) bc (d) exit

Q.3: Which of the following is process related command?

(a) Whence (b) Kill

(c) bc (d) exit

Q.4: Which of the following in directory manipulation command?

(a) chmod (b) fgrep

(c) tail (c) Mkdir

Q.5: Which of the following is file manipulation command?

(a) chmod (b) fgrep

(c) tail (c) Mkdir

Q.6: Which of the following is file content locating command?

(a) chmod (b) fgrep

(c) tail (c) Mkdir

Q.7: Which of the following is printing Command?

(a) Cpio (b) Crontab

(c) lpstat (d) fgrep

Q.8: Which of the following is Scheduling Command?

(a) Cpio (b) Crontab

(c) lpstat (d) fgrep

Q.9: Which of the following in Status Command?

(a) Cpio (b) Crontab

(c) lpstat (d) fgrep

Q.10: Which of the following in Status Command?

(a) Cpio (b) Crontab

(c) lpstat (d) fgrep

Q.11: Calendar is _____ type of command?

(a) Locating (b) File Search

(c) Printing (d) Miscellaneous

Q.12: Nice is _____ type of command?

(a) Process-Related (b) File Search

(c) Locating (d) Printing

Q.13: Chown is _____ type of command?

(a) Process-Related (b) File Search

(c) Locating (d) File-manipulation

Q.14: Read is _____ type of command?

(a) Locating (b) File Search

(c) File Content (d) Printing

Q.15: 'Pr' is _____ type of command?

(a) Locating (b) File Search

(c) Printing (d) File Content

Q.16: 'Atq' is _____ type of command?

(a) Status (b) Scheduling

(c) Printing (d) Status

Q.17: Compress is _____ type of command?

(a) Locating (b) Scheduling

(c) Printing (d) Status

Q.18: 'Env' is _____ type of command?

(a) Locating (b) Storage

(c) Printing (d) Status

Q.19: 'Pcat' is _____ type of command?

(a) Locating (b) File Search

(c) Status (d) Miscellaneous

Q.20: 'Pcat' is _____ type of command?

(a) User-Related (b) File Search

(c) Printing (d) Process-Related

ANSWER KEY

Q.1 (d)	Q.2 (a)	Q.3 (b)	Q.4 (a)	Q.5 (d)
Q.6 (b)	Q.7 (c)	Q.8 (b)	Q.9 (a)	Q.10 (c)
Q.11 (d)	Q.12 (a)	Q.13 (d)	Q.14 (c)	Q.15 (c)
Q.16 (b)	Q.17(d)	Q.18 (d)	Q.19(c)	Q.20 (a)

Short Questions

1. List down the User Related Commands?
2. Briefly explain login & rlogin command?
3. What do you know about telnet command?
4. Explain passwd & exit command briefly?
5. List down Locating Commands?
6. Explain which Command?
7. Write a note on whence Command?
8. What do you know about where Command?
9. Write a note on man command?
10. List down section of man commands?
11. List down the Process –Related commands?
12. Explain kill Command briefly?
13. Explain nice Command Briefly?
14. Explain ps command briefly?
15. Write a note on jobs Command?
16. What do you know about wait Command?
17. What is the purpose of nohup command?
18. Write the use of sleep Command?
19. List down file manipulation command?
20. Write a note on touch Command?
21. Explain chmod Command?

22. Explain chgrp Command?
23. Explain chown Command?
24. Explain rm Command?
25. Explain mv Command?
26. Explain cp Command?
27. List down Directory manipulation commands?
28. Explain mkdir Command?
29. Explain mkdir Command?
30. List down File Content Commands?
31. Explain more Command?
32. Explain less Command?
33. Explain tail Command?
34. Explain head Command?
35. Explain wc Command?
36. Explain read Command?
37. Explain tee Command?
38. List down File Content Search Commands?
39. Explain fgrep Command?
40. Explain fgrep Command?
41. Explain strings Command?
42. List down Printing Commands?
43. Explain cancel Command?
44. Explain lp Command?
45. Explain pr Command?
46. Explain lpstat Command?
47. List down Scheduling Command?
48. Explain at Command?
49. Explain atq Command?
50. Explain crontab Command?
51. List down Status Commands?
52. Explain compress Command?
53. Explain cpio Command?
54. Explain dd Command?
55. Explain pcat Command?

56. Explain pack Command?
57. Explain uncompress Command?
58. Explain unpack Command?
59. List down Storage Commands?
60. Explain date Command?
61. Explain env Command?
62. Explain iostat Command?
63. Explain sar Command?
64. Explain uptime Command?
65. Explain vmstat Command?

Long Questions

1. What do you know about user related commands? Explain any three.
2. Explain Locating commands with four examples.
3. Write a note on process related commands with examples.
4. Explain directory manipulation commands with examples.
5. Explain file manipulation commands with examples.
6. Explain File Content Search Commands with three examples
7. What do you know about printing commands? Explain with three examples.
8. What do you know about Scheduling commands? Explain with three examples.
9. What do you know about Storage commands? Explain with three examples.
10. Write a note on any two of the following commands
    I. Bc Command
    II. Clear Command
    III. Calendar Command
    IV. Time Command

Bibliography

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar

CHAPTER 05 Text Processing

Objectives
After completion of this chapter students will be able to:

5.1. Invoke vi editor
5.2. Compose text using vi editor
5.3. Describe different mode of vi editor

Text Processing

Text processing UNIX commands are those affecting text and text files. UNIX provides several commands to process the contents of a text file.

Cut Command
You can use the ***cut*** command to extract data from each line of a text file. This command can be used from a file that contains data records so that each line consists of one or more fields separated by tab characters. Following is a list of some of the flags that can be used with the cut command:

- **-ccharacterlist** to specify a list of characters to be cut from each line.
- **-ffieldlist** to specify a list of fields to be cut from each line. You can additionally specify a flag **- dcharacter** to override the character to be interpreted as the field.

Ex Command
The ***ex-command*** invokes the ***ex-editor*** to edit one or more files. Following is a list of some of the flags that can be used with the ***ex*** command:

-c subcommand to perform the specified subcommand on the specified file before invoking the ***ex*** command.

- **-R** to disallow updating the file.
- **-wsize** to set the window to number of lines equal to size
- **-v** to invoke the **vi editor**.
- **-r** file to do the recovery on the specified file.

Once you are in the ***ex editor***, you can use the following subcommands to move around in the file and edit the file:

- **z** to invoke full screen mode.
- **u** to undo the last change made.
- **n** to move to the next file if you have invoked ex editor with multiple files.
- **/pattern/** to find a pattern in the file.
- **d** to delete one or more lines.
- **a** to append.

The ex operates in the following modes:

Command mode: When the **ex editor** starts, it is in command mode which is a: (colon) prompt where you can enter a sub-command.

Text input mode: In this mode, you can add or change text in the file. You can enter text using the ***a, i or c*** sub-commands. The use of these sub-commands will allow you to enter text in the buffer. You can return to the command mode by entering a. (a single period) as the first character of the text buffer.

Fmt Command

The ***fmt*** command can be used to format files to a 72-character line by default. The ***fmt*** command preserves the blank lines in the input file as well as the spacing between words. You can modify the line length using the ***-Width*** flag.

Fold Command

The ***fold*** command can be used to generate multiple lines from a single line by splitting the line at the specified position. By default, the line length is 80 bytes. A newline character is inserted at the end of the line. Following is list of flags that can be used with the fold command:

- ***-b*** to specify the position in bytes.
- ***-s*** to split a line after the last space at a position that is less than or equal to the specified width.
- ***-w*** width to specify the line width.

Join Command

The ***join*** command can be used to merge two files (one can be standard input) to create a third file (can be standard output). Each line in the file is merged on the basis of a field that has the same value in both input files to create one line in the output file. The fields in each file are separated by either a space or tab character.

Following is a list of flags that can be used with the ***join*** command:

- **-1 field or -j1** field to specify that the join should be made on the basis of the field in the first file.
- **-2 field or -j2** field to specify that the join should be made on the basis of the field in the second file.
- **-e** string to specify that blank fields in the output file be replaced by the specified string.
- **-o fileid.fieldnumber** to specify that the output should consist of the specified fields. You can specify multiple fields separating them by commas.
- **-t** character to modify the field separator character from the default value of space.

Paste Command

The paste command can be used to paste lines from one or more files (one of them can be a standard input) to the standard output, which can be redirected

to a file. The paste command concatenates the line from each input file to the output file separating them by the tab character (default). Following is a list of flags that can be used with the paste command:

- **-dlist** to specify characters that will be used to separate corresponding lines from the input files in the output file. You can specify multiple characters if you have multiple input files.
- **-s** to merge subsequent lines from input file for each input file, one at a time, separated by the specified delimiter character.

Sort Command

The ***sort*** command is used to sort one or more files in the specified order by the specified key. It can also be used to merge files that have already been sorted. When more than one file is used, the sort command concatenates these files before sorting according to specifications. Following is a list of some of the flags that can be used with the sort command:

- -**kkey** to specify the key on which to sort. The specification for the key includes the starting field and column position and end field and column position.
- **-A** to specify that sorting be done according to ASCII collating sequence.
- **-c** to check whether the specified files are sorted according to the specified key and order.
- **-d** to sort according to dictionary order.
- **-f** to change all letters to uppercase before the sort.
- **-i** to ignore nondisplayable characters for comparison.
- **-m** to merge pre-sorted input files.
- **-n** to sort according to numeric value.
- **-ofile** to redirect the output to the specified file instead of standard output.
- **-r** to sort the output in the reverse order of the specified order.
- **-u** to create only one line in the output for lines that sort identically.

Tr Command

You can use the tr command to translate or delete characters from standard input to generate standard output. Following is the detail of the main functions of the tr command:

- Translate characters specified input to a new specified character in the output.
- Delete specified characters input from the input to generate the output.
- To delete all but the first occurrence of the specified characters.

Following is a list of some of the flags that can be used with the ***tr*** command:

- ***-c*** to translate all but the specified characters by the specified new character.
- ***-d*** to delete the specified characters.
- ***-s*** to delete all but the first occurrence of the specified characters.

Uniq Command

The ***uniq*** command can be used to eliminate duplicate adjacent lines from a file or from standard input to generate standard output or another file. This is the default operation. However, it is possible to compare only part of a line for comparison by using certain flags.

Following is a list of some of the flags that can be used with the uniq command:

- **-c** to precede each line with a number while displaying the output (the number specifies the number of occurrences of the line in the input file).
- **-d** to display only the lines that occur multiple times adjacent to each other in the input file.
- **-u** to display only the lines that appear only once in the input file.
- **-s numberofcharacter or +numberofcharacters** to specify the number of characters from the start of a line that will be ignored while comparing adjacent lines.

Sed Command

You can use the ***sed*** command to edit a file using a script. In the script, you can specify commands to edit one or more lines according to rules specified as part of one or more commands. Following is a list of some of the flags that can be used with the sed command:

- ***-e*** command to use the specified ***sed*** command to edit the file.
- ***-f*** filename to use the filename as the editing script to edit the file.
- ***-n*** to suppress messages from sed.

The ***sed*** command uses two different areas while performing editing:

- pattern area to hold selected lines for editing.
- hold area to temporarily hold the lines.

5.1. INVOKE VI EDITOR

The VI editor is the most popular and classic text editor in the Linux family. Below, are some reasons which make it a widely used editor.

- It is available in almost all Linux Distributions.
- It works the same across different platforms and Distributions
- It is user-friendly. Hence, millions of Linux users love it and use it for their editing needs

How to Use Vi Editor

To launch the VI Editor -Open the Terminal (CLI) and type

```
vi <filename_NEW> or <filename_EXISTING>
```

Open a file with *vi*. Type: *vi myfile.txt*

- If ***myfile.txt*** does not exist, a screen will appear with just a cursor at the top followed by tildes (~) in the first column.

- If ***myfile.txt*** does exist, the first few line of the file will appear.
- The status line at the bottom of your screen shows error messages and provides information and feedback, including the name of the file

Vi Editing commands

Following flags are used with ***vi editor*** for editing

- i – Insert at cursor (goes into insert mode)
- a – Write after cursor (goes into insert mode)
- A – Write at the end of line (goes into insert mode)
- ESC – Terminate insert mode
- u – Undo last change
- U – Undo all changes to the entire line
- o – Open a new line (goes into insert mode)
- dd – Delete line
- 3dd – Delete 3 lines.
- D – Delete contents of line after the cursor
- C – Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
- dw – Delete word
- 4dw – Delete 4 words
- cw – Change word
- x – Delete character at the cursor
- r – Replace character
- R – Overwrite characters from cursor onward
- s – Substitute one character under cursor continue to insert
- S – Substitute entire line and begin to insert at the beginning of the line
- ~ – Change case of individual character

5.2. COMPOSE TEXT USING VI EDITOR

Basic Cursor Movement

Cursor movement in ***vi editor*** are done using following keys

- k Up one line
- j Down one line
- h Left one character
- Right one character (or use <Spacebar>)
- w Right one word
- b Left one word

Entering, Deleting, and Changing Text

Entering, deleting and changing text in ***vi editor*** are done using following keys

- i Enter text entry mode.
- x Delete a character.
- dd Delete a line.
- r Replace a character.
- R Overwrite text, press <Esc> to end.

Setting Basic Options in vi

Displaying Line Numbers

- : set nu Display line numbers
- : set nonu Hide line numbers

Setting Right Margin

- : set wm= number Set Wrap Margin number of spaces from right edge of screen
- : set wm=10 Set Wrap Margin 10 spaces from right edge of screen
- : set wm=0 Turn off Wrap Margin

Exiting vi

To exit you must be in command mode-press <Esc> if you are not in command mode

You must press <Return> after commands that begin with a: (colon)

- ZZ Write (if there were changes), then quit
- :wq Write, then quit
- :q Quit (will only work if file has not been changed
- :q! Quit without saving changes to file

5.3. DESCRIBE DIFFERENT MODE OF VI EDITOR

There are two modes in **vi editor.**

- Command Mode
- Insert Mode

Command Mode

- The vi editor opens in this mode, and it only understands commands
- In this mode, you can, move the cursor and cut, copy, paste the text
- This mode also saves the changes you have made to the file
- Command mode is the mode in which commands are given to move around in the file, to make changes, and to leave the file
- Commands are case sensitive: j not the same as J
- Most commands do not appear on the screen as you type them. Some commands will appear on the last line: : / ?

Insert (or Text) Mode

- The mode in which text is created. (You must press <Return> at the end of each line unless you've set wrap margin.)
- There is more than one way to get into insert mode but only one way to leave: return to command mode by pressing <Esc>
- When in doubt about which mode you are in, press <Esc>

Multiple Choice Questions

Q.1: What is text preprocessing in the context of operating systems?

(a) Parsing and analyzing text data in an operating system

(b) Cleaning and preparing text data before it is processed by the operating system

(c) Converting text data into machine-readable format

(d) Generating natural language outputs from text data in an operating system

Q.2: Which of the following is NOT a step-in text preprocessing in operating systems?

(a) Tokenization (b) Lemmatization

(c) Syntax analysis (d) Stop word removal

Q.3: Which of the following is an example of text cleaning in operating systems?

(a) Removing HTML tags from text

(b) Removing stopwords from text

(c) Converting text to lowercase

(d) All of the above

Q.4: How many modes are there in Vi editor.?

(a) One (b) Two

(c) Three (d) Four

Q.5: Which of the following is a mode of Vi editor?

(a) Insert (b) Eject

(c) Command (d) a&c

Q.6: Following command is used to display line number?

(a): set nu (b): set no

(c): set numb (d): display

Q.7: _____ Key is used to move one line up?

(a) u (b) d

(c) k (c) l

Q.8: _____ Key is used to delete character?

(a) x (b) y

(c) z (c) c

Q.9: _____ Key is used to undo last change?

(a) Q (b) d

(c) u (c) m

Q.10: Which technique is used to handle misspelled words in text data for operating systems?

(a) Tokenization (b) Spell checking and correction

(c) Stemming (d) Lemmatization

ANSWER KEY

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

Short Questions

1. What is the purpose of text preprocessing in operating systems?
2. What is the vi editor and how is it used for text preprocessing?
3. How do you open a file in the vi editor for text preprocessing?
4. What are the different modes in the vi editor, and how do they relate to text preprocessing?
5. How can you navigate within a file using the vi editor?
6. List down any five text processing commands?
7. How many modes are in vi editor?
8. Write a note on Cut command?
9. Write a note on Fmt Command?
10. Write a note on Fold Command?
11. Write a note on Join Command?
12. Write a note on Sort Command?
13. How to open file in Vi editor?
14. List down the keys used for cursor movement?
15. How to exit vi editor?

Long Questions

1. Explain Vi editor mode in Unix/Linux?
2. How to compose text in Vi editor in Linux/Unix?
3. Explain Uniq command with all Flags?
4. Explain sort command with all flags?

Bibliography

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar

CHAPTER 06 BOURNE SHELLS

Objectives
After completion of this chapter students will be able to:

The Bourne shell (sh) is a command-line interpreter that is commonly used in Unix and Unix-like operating systems. It was written by Stephen Bourne at Bell Labs in the early 1970s and is one of the oldest and most widely used shells.

The Bourne shell provides a command-line interface to the operating system and allows users to execute commands, manipulate files and directories, and control the behavior of the system. It supports basic programming constructs such as variables, control flow statements, and loops, making it useful for writing scripts and automation tasks.

WHAT IS COMMAND AND ITS STRUCTURE

In an operating system, a command is a specific instruction given by a user to a computer program or shell to perform a particular task. The structure of a command in an operating system can vary depending on the shell or command interpreter being used, but the basic components remain the same.

The structure of a command in an operating system typically includes the following components:

```
CSS                                                    Copy code

command [options] [arguments]
```

- The **command** is the name of the program or operation that you want to execute.
- The **options** modify the behavior of the command, such as enabling or disabling certain features or specifying configuration settings.
- The **arguments** are the inputs that the command needs to perform its task, such as file names or search terms.

WHAT IS COMMAND LINE

In an operating system, the command line is a text-based interface that allows users to interact with the system by entering commands and receiving text-based output. It provides an alternative to graphical user interfaces (GUIs) and allows users to perform a wide range of tasks, such as managing files, running programs, and configuring system settings.

The command line interface typically consists of a command prompt, which displays the user's current working directory and a cursor, waiting for the user to enter a command. The user can then enter commands, which are executed by the operating system or shell.

The commands entered on the command line are typically composed of the command name, followed by any required or optional parameters, such as filenames or options that modify the command's behavior. The command line syntax can vary depending on the operating system and shell being used.

WHAT IS SHELL IN OPERATING SYSTEM

In operating systems, a shell is a command-line interface that provides a user with access to the operating system's services and utilities. It is a program that acts as an intermediary between the user and the operating system, allowing the user to enter commands and receive output in response.

A shell typically provides a command-line prompt, where the user can enter commands to perform tasks such as managing files, running programs, and configuring system settings. The shell interprets the user's commands and executes them, either directly or by invoking other programs as needed.

There are several different **types of shells**, including the Bourne shell (sh), the C shell (csh), the Korn shell (ksh), and the Bourne-Again shell (bash), which is the default shell for most Linux distributions. Each shell has its own syntax, built-in commands, and scripting language, although they share many common features.

6.1. INTRODUCTION TO BOURNE SHELLS

The Bourne shell is a command-line interpreter used in Unix and Unix-like operating systems. It is named after its creator, Stephen Bourne, who wrote it in the early 1970s while working at Bell Labs. The Bourne shell is one of the oldest and most widely used shells in Unix-based operating systems.

The primary function of a shell is to provide a command-line interface to the operating system. A user interacts with the system by entering commands into the shell, which then executes them. The Bourne shell supports basic programming constructs, such as variables, loops, and conditional statements, allowing users to write scripts to automate tasks.

One of the advantages of the Bourne shell is its portability. It is available on most Unix and Unix-like systems, including Linux, macOS, and the various versions of Unix itself. The shell is also lightweight and efficient, making it well-suited for use in resource-constrained environments.

While the Bourne shell has been around for several decades, it has been surpassed in popularity by other shells, such as the Bourne-Again shell (bash) and the Z shell (zsh). However, the Bourne shell is still widely used, particularly in environments where its lightweight and efficient nature is an advantage.

6.2. BOURNE SHELL BASICS

The Bourne shell is a command-line interpreter that provides a simple and lightweight interface for interacting with the Unix/Linux operating system. Here are some basic concepts and commands to get started with using the Bourne shell:

1. **Command syntax:** The Bourne shell uses a simple syntax for entering commands, with commands typically consisting of a command name followed by any options or arguments. For example, to list the contents of the current directory, you can use the "ls" command:

```
bash

ls
```

2. **Command options:** Many commands have options that can modify their behavior. Options are usually specified using a single dash (-) followed by a letter or a word. For example, to list the contents of a directory in long format, you can use the "-l" option with the "ls" command:

```
bash

ls -l
```

3. **Command arguments:** Commands can also take arguments, which are usually specified after the command name and any options. Arguments can be filenames, directories, or other parameters that affect the behavior of the command. For example, to display the contents of a file using the "cat" command, you can specify the filename as an argument:

```
bash

cat myfile.txt
```

4. **Variables:** The Bourne shell supports variables, which can be used to store and manipulate data. Variables are typically set using the "=" operator, and can be referenced using the "$" symbol. For example, to set a variable named "myvar" to the value "hello", you can use the following command:

```
makefile

myvar="hello"
```

To reference the value of the variable, you can use the "$" symbol:

```
bash

echo $myvar
```

This will print the value of the variable to the terminal.

5. **Environment variables:** The Bourne shell also supports environment variables, which are variables that affect the behavior of the shell and other programs. Environment variables are typically set using the "export" command, and can be referenced using the "$" symbol. For example, to add a directory to the system's search path, you can set the "PATH" environment variable:

```
ruby

export PATH=$PATH:/my/new/directory
```

This will add the directory "/my/new/directory" to the system's search path. these are some basic concepts and commands to get started with using the Bourne shell in Unix/Linux. The Bourne shell is a powerful tool for interacting with the operating system, and can be used for a wide range of tasks, from running simple commands to writing complex shell scripts.

Define Unix/Linux shells

In Unix and Linux operating systems, a shell is a program that provides a command-line interface for interacting with the operating system. It is

essentially a command-line interpreter that reads user input and executes the corresponding commands. There are several different shells available in Unix and Linux, each with its own syntax and features. Some of the most commonly used shells include:

1. Bourne shell (sh): The original Unix shell, created by Stephen Bourne in the 1970s. It is a simple and lightweight shell that is still used in some environments today.
2. Bourne-Again shell (bash): A more advanced and widely used shell that is compatible with the Bourne shell. It includes many additional features, such as command history and tab completion.
3. C shell (csh): A shell that includes features for interactive use, such as command history and job control. It has a syntax similar to the C programming language.
4. Korn shell (ksh): A shell that is similar to the Bourne shell, but includes many additional features such as advanced scripting capabilities.
5. Z shell (zsh): A highly customizable shell that includes advanced features such as programmable completion and spelling correction.

Shells are an essential component of Unix and Linux systems, and they are used extensively by developers, system administrators, and power users to interact with the operating system and automate tasks.

Explain the functions of shell

In an operating system, a shell is a program that acts as an interface between the user and the operating system. It provides a command-line interface that allows users to interact with the system by entering commands and executing them. Some of the primary functions of a shell in an operating system include:

Command execution: A shell can execute commands entered by the user and display the results. These commands can be system commands, shell scripts, or other executable programs.

Input/output redirection: A shell can redirect input and output from/to files or other programs. This allows users to perform tasks such as piping the output of one command to another command or redirecting output to a file.

Scripting: A shell can interpret and execute shell scripts, which are files containing a series of commands. This allows users to automate repetitive tasks or create complex scripts for performing more advanced tasks.

Environment management: A shell can manage the environment variables that affect the behavior of the system and user applications. This includes setting variables, exporting variables to child processes, and modifying the PATH variable to include directories containing executable programs.

Job control: A shell can manage and control the execution of multiple processes and jobs, allowing users to run multiple tasks simultaneously or in the background.

Shells can be used to manage network connections and perform networking tasks, such as pinging other computers, connecting to remote servers via SSH, and transferring files using protocols such as FTP and SCP. This can be useful for managing remote servers and networks from the command line.

the shell is a powerful tool for interacting with an operating system and performing a wide range of tasks. It provides a flexible and customizable interface that allows users to tailor their experience to their specific needs and workflows.

Describe Bourne Shell of Unix/Linux

The Bourne shell (sh) is one of the oldest and most widely used shells in Unix and Unix-like operating systems. It was created by Stephen Bourne in the 1970s while working at Bell Labs and was included in the first versions of the Unix operating system. The Bourne shell is a command-line interpreter that provides a simple and lightweight interface for interacting with the operating system. It supports basic programming constructs such as variables, loops, and conditional statements, making it useful for writing scripts and automating tasks.

Some of the key features of the Bourne shell include:

Simple syntax: The Bourne shell has a simple and easy-to-learn syntax, which makes it a popular choice for beginners and for writing quick scripts.

Portability: The Bourne shell is available on most Unix and Unix-like systems, making it a portable choice for writing scripts that need to run on multiple platforms.

Efficiency: The Bourne shell is lightweight and efficient, making it well-suited for use in resource-constrained environments.

Compatibility: The Bourne shell is the standard shell for POSIX-compliant operating systems and is still widely used in some environments today.

One of the main limitations of the Bourne shell is its lack of advanced features, such as command history, tab completion, and programmable completion, which are available in more modern shells such as the Bourne-Again shell (bash) and the Z shell (zsh). However, the Bourne shell remains a popular choice for writing scripts and performing basic command-line tasks.

Use Bourne shell in Unix/Linux in operating system

To use the Bourne shell in Unix/Linux, you first need to open a terminal window or console. Once you have a command-line interface, you can start entering commands to interact with the system.

Here are some basic examples of how to use the Bourne shell in Unix/Linux:

1. Running a command: To run a command, simply type the command name followed by any options or arguments. For example, to list the contents of the current directory, you can use the "ls" command:

```
Bash

ls
```

2. Creating a shell script: To create a shell script, you can use a text editor such as "vi" or "nano" to write a series of commands in a file with a

".sh" extension. For example, to create a script that prints the current date and time, you can use the following commands:

```
nano myscript.sh
```

Then, enter the following commands in the text editor:

```
Bash

#! /bin/sh

Date
```

Save the file and exit the text editor.

3. Running a shell script: To run a shell script, you need to make the file executable using the "chmod" command, and then execute the script using its filename. For example, to run the script created in the previous step, you can use the following commands:

```
Bash

chmod +x myscript.sh

./myscript.sh
```

This will run the script and print the current date and time to the terminal. the Bourne shell provides a simple and efficient interface for interacting with the Unix/Linux operating system, and can be used for a wide range of tasks, from running simple commands to writing complex shell scripts.

STEPS INVOLVED IN WRITING A BOURNE SHELL SCRIPT

To write a Bourne Shell script in Unix/Linux, follow these steps:

1. Open a text editor and create a new file with a .sh extension. For example, "myscript.sh".
2. Add the shebang line at the top of the file. This line specifies the path to the shell interpreter, and is usually "#!/bin/sh".

3. Write your shell commands in the file, including variables, control structures, and other shell commands.
4. Save the file and exit the text editor.

DIFFERENT TYPES OF QUOTES IN BOURNE SHELL

Answer: There are three types of quotes in Bourne Shell scripts in Unix/Linux:

1. Single quotes: variables inside single quotes are not expanded. For example: 'Hello, $name' will output "Hello, $name".
2. Double quotes: variables inside double quotes are expanded. For example: "Hello, $name" will output "Hello, John".
3. Backticks: variables inside backticks are expanded as a command substitution. For example: date will output the current date and time.

Multiple Choice Questions

Q.1: What is the primary function of a shell?

(a) Manage files and directories (b) Run shell scripts

(c) Execute commands entered by the user

(d) Manage environment variables

Q.2: What is a shell script?

(a) A program written in a shell language that can be executed in the shell

(b) A program written in Python that can be executed in the shell

(c) A program that can only be executed in a graphical user interface.

(d) A program that can only be executed in a terminal emulator

Q.3: What is input/output redirection in shells?

(a) Allowing users to redirect input and output to and from files

(b) Allowing users to redirect input and output to and from the Internet

(c) Allowing users to redirect input and output to and from other processes

(d) Allowing users to redirect input and output to and from the command history

Q.4: What are environment variables in shells?

(a) Variables that hold information about the system or user environment

(b) Variables that hold information about network connections

(c) Variables that hold information about shell commands
(d) Variables that hold information about user preferences

Q.5: What is job control in shells?

(a) Allowing shells to manage network connections

(b) Allowing shells to manage jobs and processes

(c) Allowing shells to manage user preferences

(d) Allowing shells to manage text processing tasks

Q.6: Which shell is the default shell in most Linux distributions?

(a) Bash (b) Zsh

(c) Ksh (c) Tcsh

Q.7: Which command is used to display the current working directory in the shell?

(a) dir (b) pwd

(c) ls (d) cd

Q.8: Which shell command is used to create a new directory?

(a) mkdir (b) touch

(c) rm (d) cat

Q.9: Which symbol is used in shell commands to represent the home directory?

(a) $ (b) *

(c) # (d) ~

Q.10: Which command is used to list the files and directories in a directory?

(a) cd (b) ls

(c) cp (d) mv

ANSWER KEY

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

Short Questions

1. Who developed the Bourne Shell in Unix/Linux?
2. What is the file name of the Bourne Shell executable in Unix/Linux?
3. What is the default prompt of the Bourne Shell in Unix/Linux?
4. What is a Unix/Linux shell??
5. What is the difference between a shell and a terminal?
6. What is the primary function of a shell??
7. What is a shell script?
8. How do shells manage files and directories?
9. What is input/output redirection in shells?
10. What is job control in shells?
11. How can you execute a Bourne Shell script in Unix/Linux?
12. What is the difference between single quotes and double quotes in the Bourne Shell?
13. How can shells be used for networking tasks?
14. What is a shell script?
15. What is the purpose of the "case" statement in the Bourne Shell?

Long Questions

1. Explain the functions of shell?
2. What are the different types of quotes in Bourne Shell scripts in Unix/Linux, and how do they affect variable expansion?
3. What is the Bourne Shell in Unix/Linux, and what are its features?

4. What are the steps involved in writing a Bourne Shell script in Unix/Linux, and how can you execute it?

Bibliography

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar.

CHAPTER 07 SYSTEM ADMINISTRATION

Objectives
After completion of this chapter students will be able to:

7.1. System Administration Tasks
7.2. Unix/Linux Installation Basics
7.3. Resource and User Administration
7.4. File System and Disk Administration
7.5. System Accounting and Performance Monitoring
7.6. Device and Mail Administration
7.7. UUCP and FTP Services Administration
7.8. Backing Up and Restoring the System

System administration is the process of managing computer systems, networks, servers, and software in order to ensure optimal performance and security. It involves a wide range of tasks, including setting up and configuring hardware and software, managing ser accounts and permissions, monitoring system performance and resource usage, troubleshooting technical issues, and implementing security measures to protect the system from unauthorized access and other threats.

A system administrator is responsible for the day-to-day operation of a computer system and must be knowledgeable in various areas such as system administration tools, network protocols, operating systems, and security best practices. They must also be able to communicate effectively with users, other administrators, and management in order to ensure that the system meets the needs of the organization.

DESCRIBE UNIX/LINUX SYSTEM ADMINISTRATION

Unix/Linux system administration involves the management and maintenance of Unix or Linux-based operating systems. It encompasses a wide range of tasks and responsibilities, including installation, configuration, monitoring, maintenance, security, and troubleshooting.

7.1. SYSTEM ADMINISTRATION TASKS

Linux/Unix system administration involves managing the operating system and the applications running on a Linux/Unix server. System administrators are responsible for configuring, installing, securing, and maintaining the server, ensuring that it runs smoothly and efficiently.

Some of the key tasks involved in Linux/Unix system administration include:

1. Installing and configuring the operating system and applications.
2. Managing user accounts, file systems, and permissions.
3. Monitoring system performance and troubleshooting issues.
4. Creating and implementing backup and disaster recovery plans.
5. Managing network services, such as DNS, DHCP, and email.
6. Implementing security measures to protect the server from threats.
7. Applying software patches and updates to ensure system security and stability.

Linux/Unix system administrators need to have a strong understanding of the operating system and its components, as well as experience with scripting languages and command-line interfaces. They must also have excellent problem-solving skills and be able to work under pressure to resolve issues quickly and efficiently.

Linux/Unix system administration tasks can vary depending on the specific needs of an organization or individual users, but some common tasks include:

1. Installing and configuring the operating system and software: This involves setting up the server with the necessary software and applications required for the organization's needs.
2. User account management: Creating and managing user accounts, setting permissions, and managing access to files and directories.
3. System monitoring: Monitoring system performance, network activity, and identifying and resolving any issues or errors.
4. Backups and Disaster Recovery: Creating and maintaining backups of critical data and systems, and having a plan in place for disaster recovery in case of a failure.
5. Security management: Implementing and maintaining security measures to protect the system from cyber-attacks and other security threats.
6. Networking services: Configuring and managing network services such as DNS, DHCP, and routing protocols.
7. Software updates and maintenance: Installing and managing software updates and patches to keep the system secure and up-to-date.
8. Scripting and automation: Creating scripts and automating routine tasks to streamline system administration tasks and improve efficiency.
9. Performance optimization: Analyzing system performance and making optimizations to improve speed, stability, and reliability.

System Administration Tasks

- Installing and configuring hardware and software
- Managing user accounts and permissions.
- Configuring and managing network infrastructure.

- Monitoring system performance and resource usage.
- Performing system backups and disaster recovery procedures.
- Troubleshooting technical issues and resolving system errors.
- Ensuring system security and implementing security measures.
- Planning and implementing system upgrades and migrations.
- Maintaining documentation and providing user support and training.

Linux system administration involves managing and maintaining a Linux-based computer system. It is an important and challenging field that requires a broad range of skills and knowledge. The Linux operating system is highly customizable and flexible, allowing system administrators to configure and manage it in a variety of ways to suit their needs. Linux is widely used in enterprise environments, data centers, and web servers

DESCRIBE TASKS OF UNIX/LINUX SYSTEM ADMINISTRATION

Some of the key tasks involved in Unix/Linux system administration include:

- **Installation and configuration of the operating system and related software:** This includes selecting the appropriate distribution or version of Unix/Linux, setting up user accounts, configuring networking and security settings, and installing software packages.
- **Monitoring system performance and health:** This involves keeping an eye on system resources like CPU usage, memory utilization, disk I/O, and network traffic to ensure optimal performance and availability.
- **Maintenance and upgrades:** Regular maintenance tasks include applying security patches and updates, cleaning up temporary files, and performing regular backups. Upgrades involve updating the operating system or software to a newer version.
- **Security management:** This involves implementing and maintaining security measures such as firewalls, intrusion detection/prevention systems, access controls, and encryption to protect the system from unauthorized access or attacks.

- **User management:** System administrators are responsible for managing user accounts, setting permissions, and enforcing password policies to ensure the security and integrity of the system.
- **Troubleshooting:** When problems arise, system administrators must be able to quickly diagnose and resolve issues to minimize downtime and prevent data loss.
- **Networking:** Unix/Linux system administrators are responsible for configuring and managing networking settings, including IP addresses, DNS, and routing. They also need to troubleshoot network-related issues and ensure that network security is maintained.
- **Automation and scripting:** Automating tasks and creating scripts to simplify routine tasks is an important part of Unix/Linux system administration. This can include automating software updates, backups, and other routine tasks to save time and improve efficiency.

Unix/Linux system administration requires a broad set of technical skills, including knowledge of the operating system, networking protocols, security practices, scripting and automation, and familiarity with command-line interfaces.

IDENTIFY HARDWARE REQUIREMENTS OF UNIX/LINUX SYSTEM

The hardware requirements for Unix/Linux system depend on the specific distribution or version of the operating system, as well as the intended use of the system. However, here are some general hardware requirements for a basic Unix/Linux system:

Processor: A 64-bit processor is recommended for most modern Unix/Linux distributions. The exact processor speed and number of cores will depend on the specific workload and the number of users that will be accessing the system.

Memory (RAM): The minimum recommended amount of memory is 2 GB, although more memory will be required for more demanding applications or for systems with more users. As a general rule, it is recommended to have at least 4 GB of memory for a basic Unix/Linux system.

Storage: The amount of storage required will depend on the specific use case and the amount of data that needs to be stored. However, a minimum of 20 GB of storage is recommended for a basic Unix/Linux installation. Additionally, it is recommended to have a separate partition for the operating system and for user data to improve system performance.

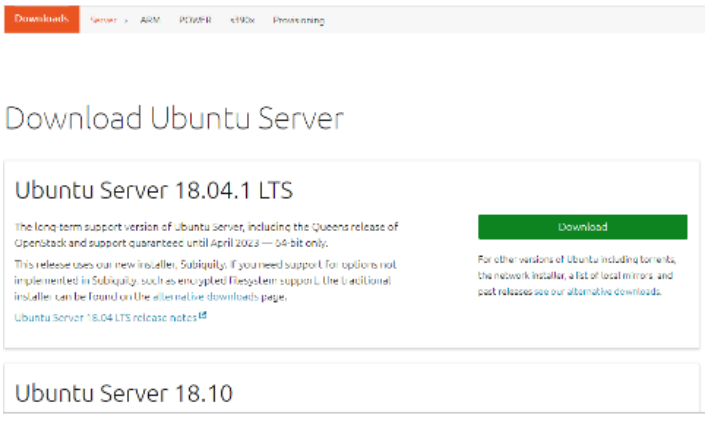
Network interface: A network interface is required to connect to a network, and most modern systems include built-in Ethernet adapters. The specific type of network interface will depend on the network configuration and the hardware available.

Peripherals: Peripherals such as a keyboard, mouse, and monitor may be required for desktop or workstation installations, but are not required for server installations.

7.2. LINUX/UNIX INSTALLATION BASICS

Install Linux Using CD-ROM or USB

Download .iso or the ISO files on a computer from the internet and store it in the CD-ROM or USB stick after making it bootable using Pen Drive Linux and UNetBootin

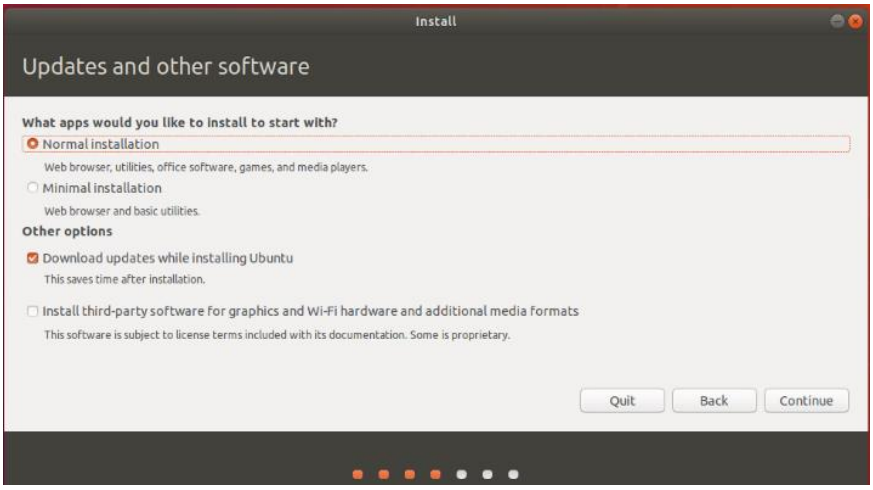


1. Boot into the USB Stick

You need to restart your computer after attaching CD –ROM or pen drive into the computer. Press enters at the time of boot, here select the CD-ROM or pen drive option to start the further boot process. Try for a manual boot setting by holding F12 key to start the boot process. This will allow you to select from various boot options before starting the system. All the options either it is USB or CD ROM or number of operating systems you will get a list from which you need to select one.

Note:
You will see a new screen when your computer boots up called “GNU GRUB”, a boot loader that handles installations for Linux. This screen will only appear in case there is more than one operating system.

- Set the keyboard layout.
- Now you will be asked What apps would you like to install to start with Linux? The two options are ‘Normal installation’ and ‘Minimal installation’.



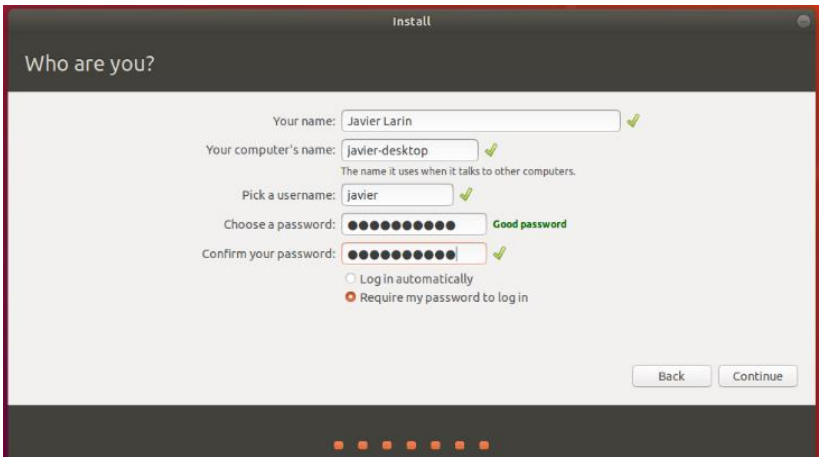
2. Drive Selection

Select the drive for installation of OS to be completed. Select "Erase Disk and install Ubuntu" in case you want to replace the existing OS otherwise select "Something else" option and click INSTALL NOW.

3. Start Installation

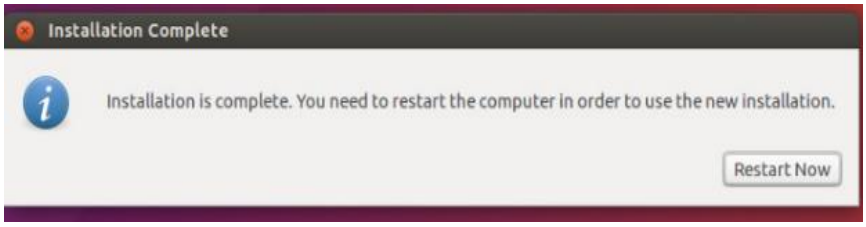
A small panel will ask for confirmation. Click Continue in case you don't want to change any information provided. Select your location on the map and install Linux.

- Provide the login details.



4. Complete the installation process

After the installation is complete you will see a prompt to restart the computer.



You can also download drivers of your choice through the System Settings menu. By following these steps:

Additional Drivers > select the graphics driver from the list.

Many useful drivers will be available in the list, such as Wi-Fi drivers. There are many other options also available to use and install Linux

B. Install Linux Using Virtual Box VMWARE

In this way, nothing will affect your Windows operating system.

What Are Requirements?

Good internet connection

At least 4GB RAM

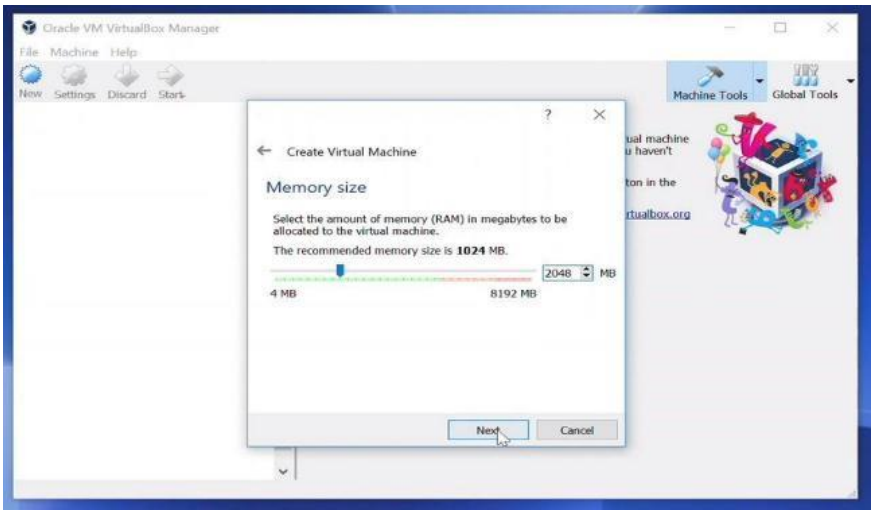
At least 12GB of free space

Steps:

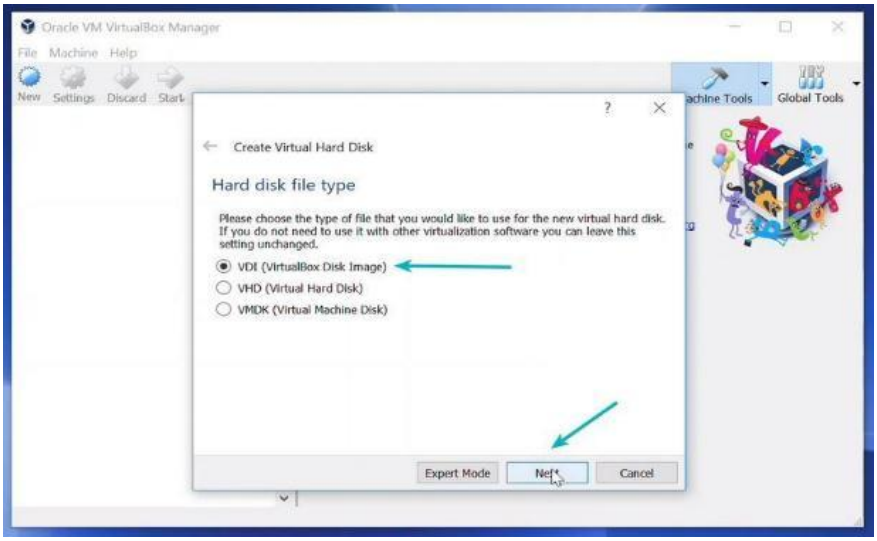
1. Download the VIRTUAL BOX from original ORACLE VIRTUAL BOX site. You can refer below link https://www.virtualbox.org/

2. Install Linux Using Virtual Box

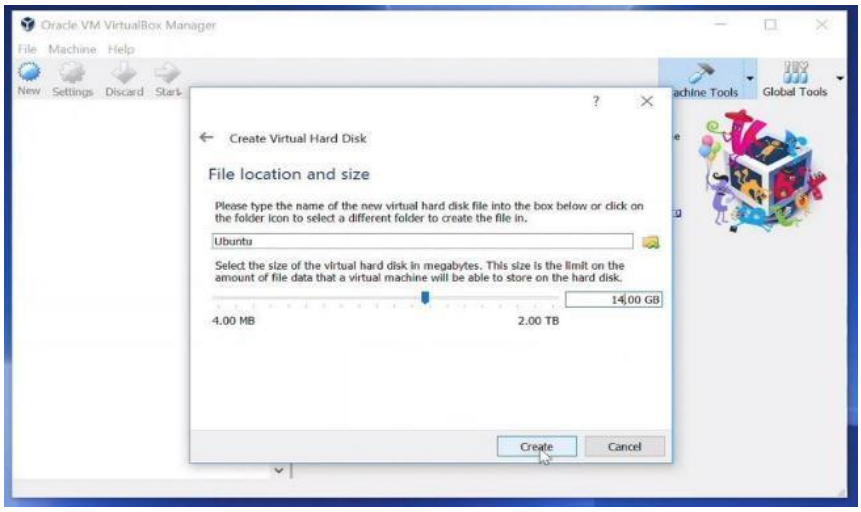
Use the .iso file or ISO file that can be downloaded from the internet and start the virtual box. Here we need to allocate RAM to virtual OS. It should be 2 GB as per minimum requirement.



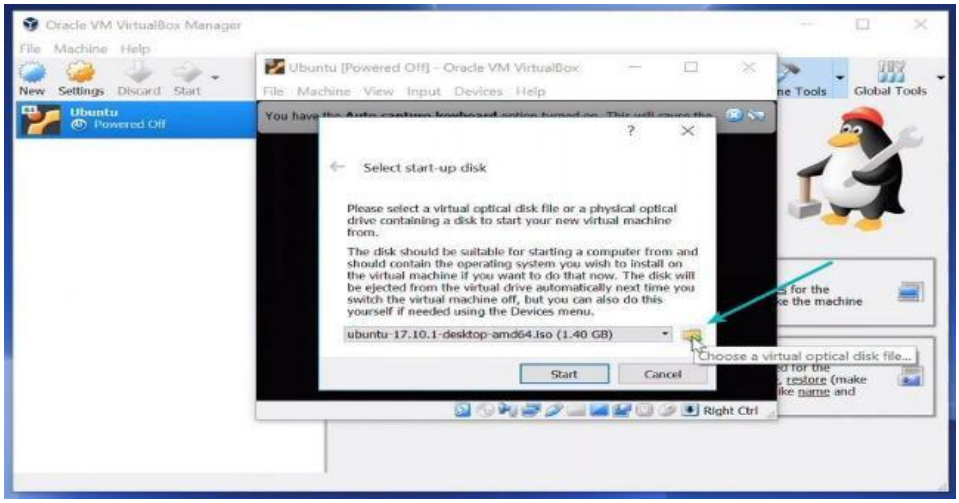
Choose an option under Create a virtual disk.



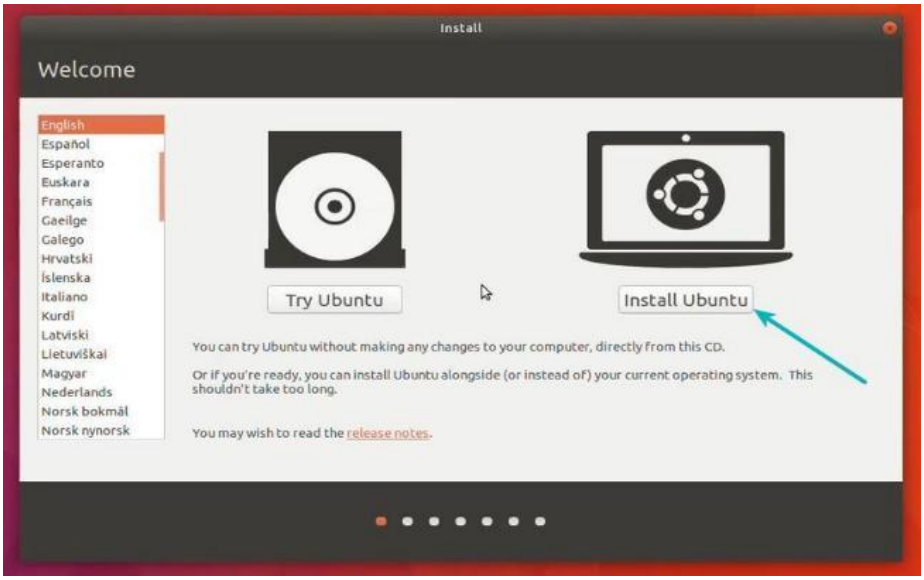
Choose a type of storage on physical hard disk. And choose the disk size (min 12 GB as per requirement)

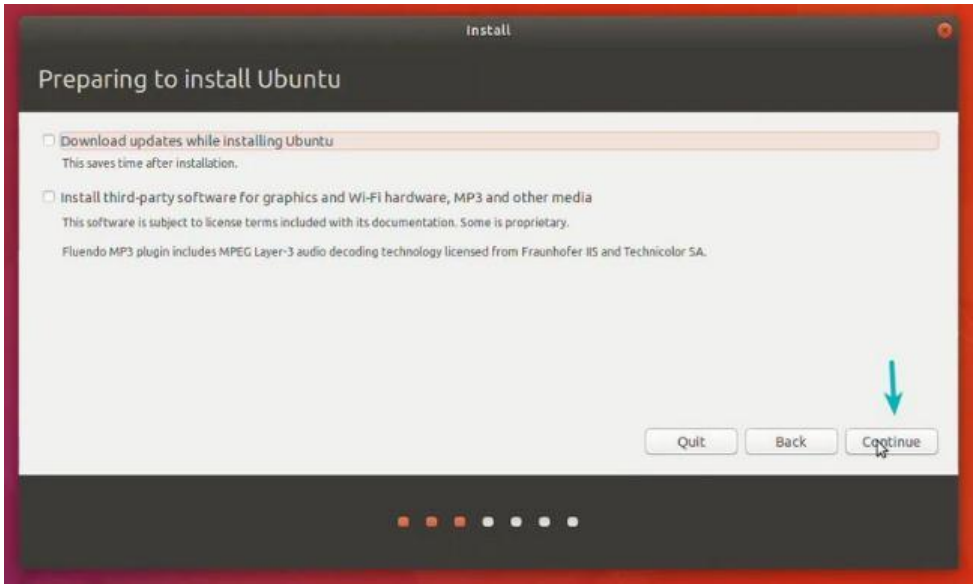


Click on create option and then click on the START button to start the virtual box and browse to the location of the .iso file of the OS.

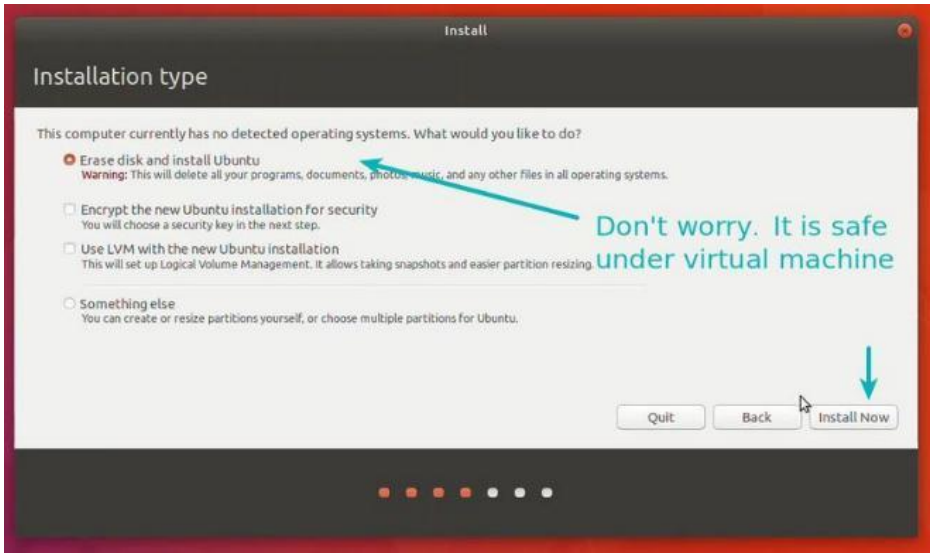


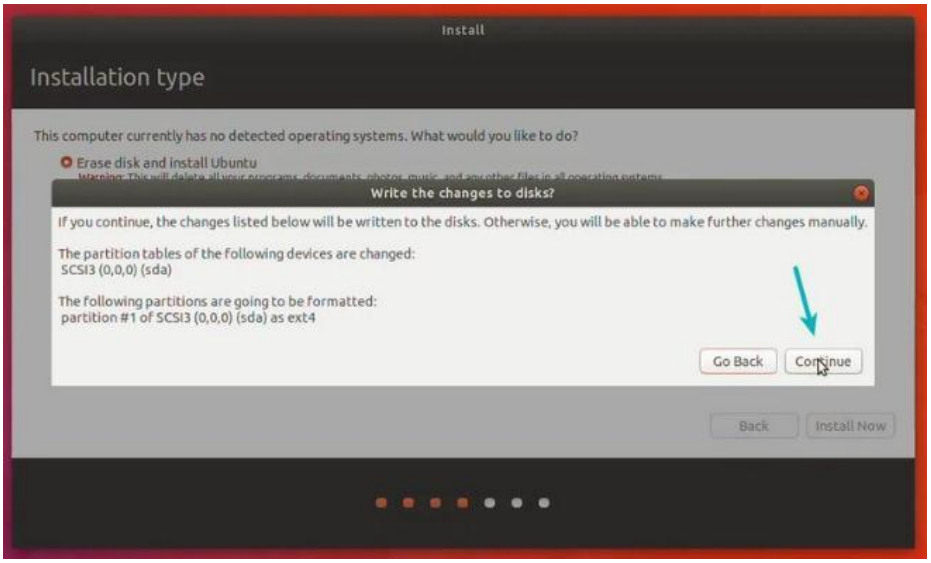
Now Linux OS will start, click on install option.

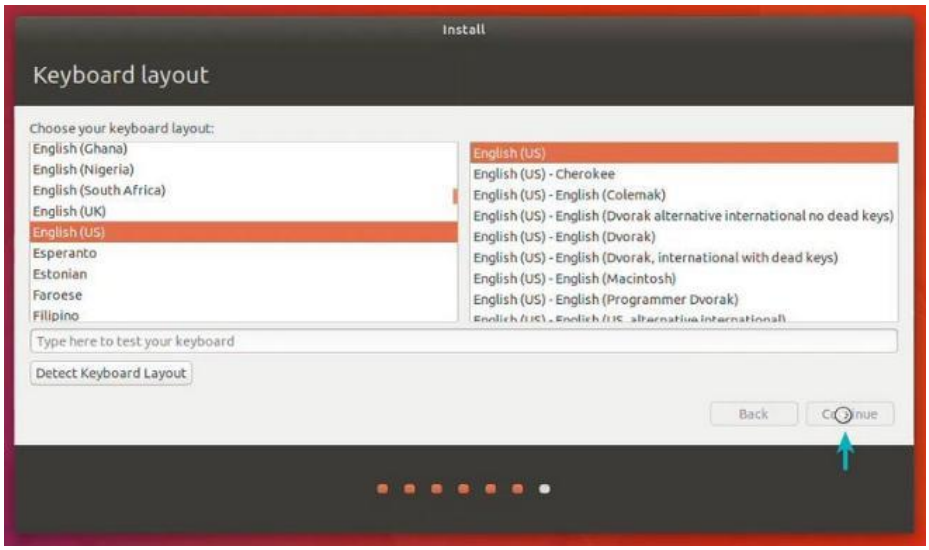




Select the drive for completing the OS installation. Select “Erase Disk and install Ubuntu” in case you want to replace the existing OS otherwise select “Something else” option and click INSTALL NOW.

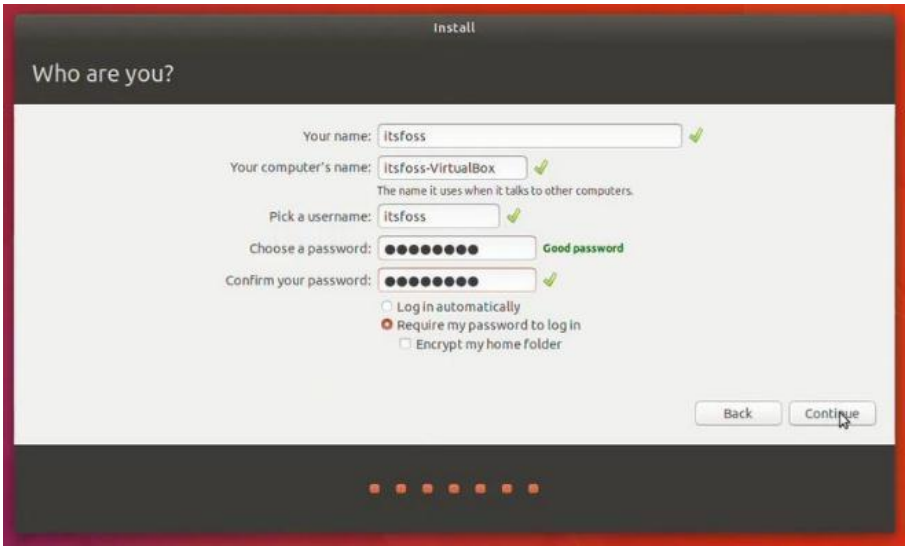




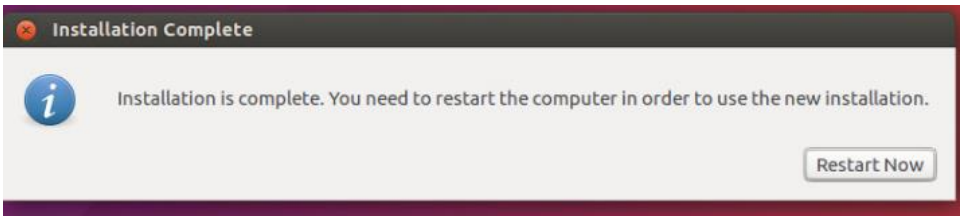


Click on Continue.

Choose a username and password.



You are almost done. It should take 10-15 minutes to complete the installation. Once the installation finishes, restart the system.



NOTE: In case of any issue close and again start the virtual box.

UNIX/LINUX INSTALLATION BASICS

Installing a Unix/Linux operating system requires some basic knowledge of computer hardware and software. Here are the general steps for installing Unix/Linux:

PLAN AND INSTALL UNIX/LINUX WITH DIAGRAM

Here is an overview of the planning and installation process for a Unix/Linux system.

Identify the purpose and requirements: Determine the intended purpose of the system and the hardware and software requirements needed to support the workload.

Choose a Unix/Linux distribution: Select a Unix/Linux distribution that is appropriate for the intended workload and hardware configuration.

Prepare the hardware: Ensure that the hardware meets the minimum requirements for the selected Unix/Linux distribution.

Create installation media: Download the installation media for the selected Unix/Linux distribution and create a bootable installation media such as a DVD or USB drive.

Boot from the installation media: Insert the bootable installation media and boot the system from it. Follow the prompts to begin the installation process.

Partition the disk: Partition the disk according to the recommended partitioning scheme for the selected Unix/Linux distribution.

Install the Unix/Linux operating system: Follow the prompts to install the Unix/Linux operating system, selecting the appropriate options such as language, time zone, and user accounts.

Configure the system: After the installation is complete, configure the system settings such as networking, security, and software packages.

Update the system: After the system is installed and configured, update it to the latest software packages and security patches.

Test the system: Test the system to ensure that it is functioning correctly and meets the intended requirements.

START UP AND SHUTDOWN UNIX/LINUX

Starting up and shutting down Unix/Linux involves a series of steps that are essential for proper system operation. Here are the basic steps for starting up and shutting down a Unix/Linux system:

Startup:

1. Power on the computer: Turn on the power to the computer.
2. Boot the system: The BIOS will run some checks and then boot the operating system.
3. Log in: Once the operating system has loaded, you will be prompted to log in with your username and password.
4. Start X Window System: If the system is configured to use the X Window System, you can start it by entering the command "startx".

Shutdown:

1. Save work: Save any open documents or files and close all running programs.
2. Log out: Log out of the system by entering the command "logout" or by clicking on the logout button in the graphical interface.
3. Shut down the system: Once logged out, you can shut down the system by entering the command "shutdown -h now". This will shut down the system immediately.
4. Restart the system: To restart the system, enter the command "shutdown -r now". This will shut down and then restart the system.

7.3. RESOURCES AND USER ADMINISTRATION

Linux/Unix provides powerful tools for managing resources and user administration. Here are some of the key features and commands:

1. **File system management**: Linux/Unix provides a hierarchical file system, which allows users to organize files and directories into a logical structure. The "ls" command is used to list files and directories, while "mkdir" is used to create new directories, and "rm" is used to remove files and directories.
2. **Disk management:** The "df" command is used to display disk usage, while "du" is used to display disk usage for specific directories.
3. **Memory management:** The "free" command is used to display memory usage, while "top" is used to display real-time system statistics, including memory usage and CPU usage.
4. **User administration:** Linux/Unix provides powerful tools for managing users and groups, including the "useradd" command for creating new users, the "usermod" command for modifying existing users, and the "userdel" command for deleting users. The "passwd" command is used to set or change a user's password.

Linux User Administration denotes how to manage a user account or group accounts. It deals with creating the user account, adding the user to the group, modifying it as well as deleting the account.

However, these all can be managed through GUI easily. Here, we would learn the commands that allow us to perform user administration. Before we begin, we should be aware of the common terminologies in the Linux User Administration.

Basic Terminologies

You might be aware of these terms but let's again look at these quickly. Some of the basic terminologies are:

- **User:** We are the "user" who has privileges to access the system's resources. Therefore, a user is an object with a unique identification number known as User ID(UID).

- **Group**: The group of users who have the privilege to access the same resources is called the group. In Linux, the users in a particular group can access all the resources assigned to that group. Group has also a unique identification number known as Group ID or GID.

Types of Users

There are three types of users in Linux:

- **SuperUser or Root User**: The administrator of the Linux system who has all the rights. The root account belongs to the superuser. The root user doesn't need permission to run any command.

System User: The users created by the software or applications. For example, we installed Apache Kafka in the system, then it will create the user account named "Apache". These are known as System Users created at the time of installing any application.

Normal User: Such accounts created by the root user are called Normal User. For example, the root user created an account named John, JournalDev and so on. The name can be anything. The root user can create it as well as has the privilege to delete the account.

User Configuration Files

There are 4 configuration files that store the information regarding user password, group information and so on. These configuration files are located in /etc directory. Let's discuss more about this.

/etc/passwd: The passwd file located in the /etc directory holds the user account information.

```
root@ubuntu:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologi
n
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/
sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nol
```

- **/etc/shadow:** This file saves the passwords for the users in an encrypted format, along with the days since the password last changed.

```
root@ubuntu:~# cat /etc/shadow
root:$6$N3fZcsFCcWyK93P6$/LSJkWnfj.eJ1ZJvqDYZyklD2ygWxtA9MHe8DST1pDyKq6r./kqtJ1p
M2Nv8O3vd8SQelB0imzpzaZgMdOi5L1:18341:0:99999:7:::
daemon:*:18295:0:99999:7:::
bin:*:18295:0:99999:7:::
sys:*:18295:0:99999:7:::
sync:*:18295:0:99999:7:::
games:*:18295:0:99999:7:::
man:*:18295:0:99999:7:::
lp:*:18295:0:99999:7:::
mail:*:18295:0:99999:7:::
news:*:18295:0:99999:7:::
uucp:*:18295:0:99999:7:::
proxy:*:18295:0:99999:7:::
www-data:*:18295:0:99999:7:::
backup:*:18295:0:99999:7:::
list:*:18295:0:99999:7:::
irc:*:18295:0:99999:7:::
gnats:*:18295:0:99999:7:::
nobody:*:18295:0:99999:7:::
systemd-network:*:18295:0:99999:7:::
systemd-resolve:*:18295:0:99999:7:::
syslog:*:18295:0:99999:7:::
```

- **/etc/group:** It holds the group account information. Whenever you create the group, the details are stored in /etc/group file.

```
root@ubuntu:~# cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:linuxdevices
sys:x:3:
adm:x:4:syslog
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:
floppy:x:25:
tape:x:26:
sudo:x:27:
audio:x:29:
```

- **/etc/gshadow:** This file is similar to the /etc/shadow file which stores the group account passwords. Since group accounts don't usually have a password, you'll see an asterisk for the password field.

```
root@ubuntu:~# cat /etc/gshadow
root:*::
daemon:*::
bin:*::linuxdevices
sys:*::
adm:*::syslog
tty:*::
disk:*::
lp:*::
mail:*::
news:*::
uucp:*::
man:*::
proxy:*::
kmem:*::
dialout:*::
fax:*::
voice:*::
cdrom:*::
floppy:*::
tape:*::
sudo:*::
audio:*::
```

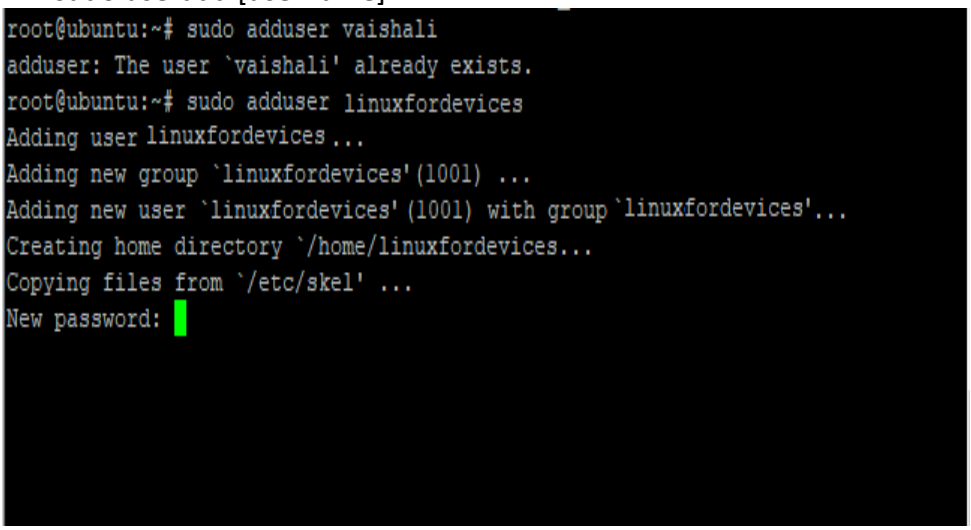
You can check all the files using cat command as shown above.

How to create a user account?

The first step in Linux user administration is knowing how to create user accounts. Earlier, we learned about the basic terminologies related to Linux user administration. Let's move further and discuss how to add or create a new user account.

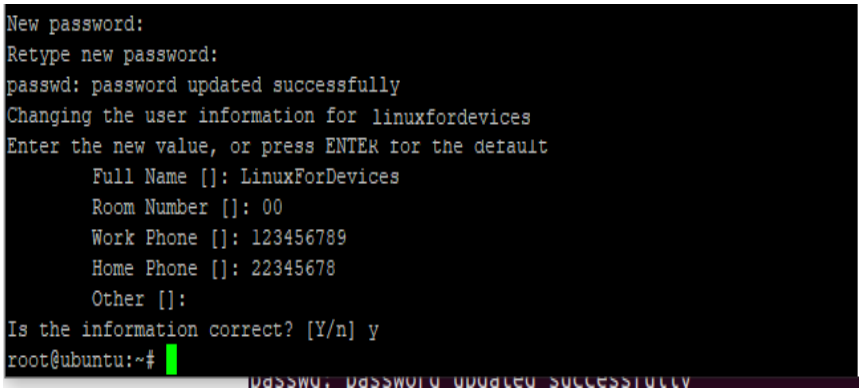
We can use either useradd or adduser command followed by the username. The username shouldn't start with uppercase letters. Let's look at the command below:

1 sudo adduser [username]
2 Or
3 sudo useradd [username]



User Add

You can see that while creating the user named "vaishali", It showed an error. Therefore, users with the same name cannot exists in the system. Further, it would ask to enter the details such as password, full name, contact details as shown in the image given below:



User Details

Hence, the new user account has been successfully created. Some of the operations are performed while creating the account such as:

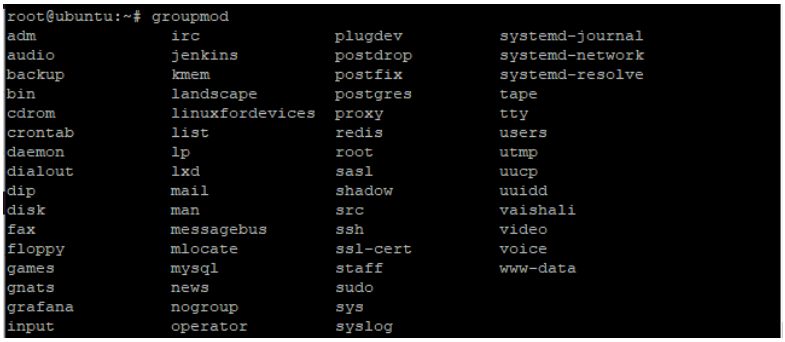
1. When we create a new user account, It automatically amends the configuration files (discussed above).
2. The home directory is assigned to the new user.
3. There are 3 hidden files mainly, .bash_logout, .bash_profile, .bashrc which are copied to the home directory of the new user.
4. Set the permission and ownership for the home directory.
5. The group with the same name as the user account is created.

Add the user to multiple groups

Earlier we discussed how to create the user. The next step in Linux user administration is understanding how to add users to groups. We learned that the group is created having the same name as of user account. What if, we want to add the user to the multiple groups already created. **"-G"** option is used to add the user to the multiple groups.

Each group name is separated by the comma. To list the group name use **groupmod followed by the tab key twice command.** Let's have a look at the command below:

```
groupmod [press tab key twice]
```



Check List of group

Add the user to the group you want using the following command given below:

useradd -G group1, group2, group3 [username]

Options used with Useradd command

There are some of the common options used with useradd command. The list of options is described below stating their use. Let's first look at the syntax:

useradd [options] LOGIN
OR
useradd -D [options]

Option	Description
-b, –base-dir, BASE_DIR	gives the base directory for the home directory of the new account.
-d, –home-dir HOME_DIR	denotes the home directory of the new account
-h, –help	prints the help message
-m, –create-home	Create the new home directory but if it doesn't exist
-p, –password PASSWORD	specifies the encrypted password of the user account
-u, –uid UID	specifies the user ID of the new user account

Delete the user account

The next step in Linux user administration is to learn how to delete the user account. We'll use the userdel command to delete the account. You can choose to get the password expired with the command below so the user cannot login. Let's have a look at the command below:

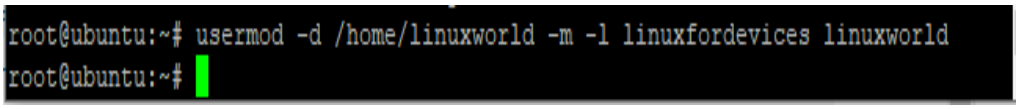
sudo passwd -l 'username'

Now use the userdel command to delete the account as shown below:

sudo userdel -r 'username'

To create an account, we use useradd command whereas to modify an already existing account, we use usermod command. This command is used to modify the attributes of an existing user account. When performing Linux user administration, it's necessary to know how to modify existing attributes of a user. The attributes can be changing a user's home directory, login name, password expiry date and so on. Let's suppose you want to change the current name of the user account. Let's have a look at the command below:

```
mod -d /home/[user_account_name] -m -l [new_name] [current_name]
```



Modify The Name 1

The current of the user account is linuxworld which has been changed to linuxfordevices. Here, -m is used to move the content of the home directory to the new location whereas -l option notifies the name of the user will be changed to "new user" from "old user". The usermod command consists of the options as useradd command. To have the detailed knowledge, check the Linux Official Documentation for Usermod command.

How to create a group?

As of now, we have learned how to create, modify and delete the user account to perform Linux user administration. Let's learn how to create a group. A group is a collection of users. The main purpose of the group is to manage users collectively. The groupadd command is used to add a newAt the time of creating a new user account, the group is created automatically with the same name. Multiple users can be a part of a group as well as a user can be a part of multiple groups. the **groupadd** command is used to create the group. Let's have a look at the command below:

```
groupadd [options] [group_name]
```

How to modify the group?

Earlier, we learned that usermod command is used to modify the user account. Similarly, groupmod command is used to modify the definition of the specified group. It also contains the options as groupadd command. The groupmod command is used to modify the group id of the group account, name of the group and so on. Let's have a look at the command below:

```
groupmod [options] groupname
```

How to delete a group?

Here, the groupdel command is used to delete the group. This command modifies the system files as well as delete all the group information related to the group. Before removing the group, it's important to remove the user. The command to delete the user is described above. Let's have a look at the syntax below:

```
groupdel groupname
```

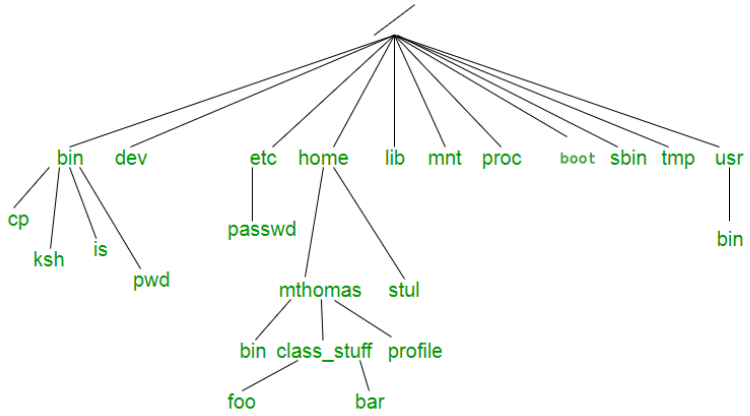
Group administration: The "groupadd" command is used to create new groups, the "groupmod" command is used to modify existing groups, and the "groupdel" command is used to delete groups.

Permissions management: Linux/Unix provides a robust system for setting permissions on files and directories, allowing users and groups to have different levels of access to files. The "chmod" command is used to modify file permissions, while "chown" is used to change the ownership of files and directories.

Resource monitoring and control: The "nice" command is used to set the priority of a process, while the "kill" command is used to terminate a running process.

7.4. FILE SYSTEM AND DISK ADMINISTRATION

Linux/Unix provides a hierarchical file system that allows users to organize files and directories into a logical structure. Here are some of the key features and commands for managing file systems and disk administration:



1. **File system types:** Linux/Unix supports several file system types, including ext2, ext3, ext4, XFS, and btrfs. Each file system type has its

own strengths and weaknesses, and the choice of file system depends on the specific requirements of the system being used.

2. **Mounting file systems:** File systems need to be mounted before they can be accessed by the system. The "mount" command is used to mount file systems, while "umount" is used to unmount them.
3. **Disk partitioning:** Linux/Unix provides several tools for disk partitioning, including fdisk and gdisk. These tools can be used to create, delete, and modify disk partitions.
4. **Disk usage analysis:** The "df" command is used to display disk usage, while "du" is used to display disk usage for specific directories.
5. **File system repair:** The "fsck" command is used to repair file system errors. This command can be used to fix minor errors or to recover data from a corrupted file system.
6. **RAID configuration:** Linux/Unix supports several RAID configurations, including RAID 0, RAID 1, RAID 5, and RAID 6. These configurations provide data redundancy and performance improvements.
7. **Disk cloning:** The "dd" command is used to clone disks. This command can be used to create a backup of a disk or to copy the contents of one disk to another.
8. **LVM management:** Logical Volume Management (LVM) is a feature that allows users to create logical volumes that span multiple disks. The "lvcreate" command is used to create logical volumes, while "lvextend" is used to extend them.

ADD AND MAINTAIN NEW USERS IN THE SYSTEM

Create User Accounts: To add new users, you will first need to create user accounts for them. Each user account should have a unique username and password, which will be used to authenticate them when they log in.

Assign User Roles: Once user accounts have been created, you can assign roles to each user based on their responsibilities and access needs. This will help you manage permissions and ensure that users only have access to the features and information that they need.

Train Users: When adding new users to a system, it is important to train them on how to use the system effectively. This can involve providing them with user manuals, conducting training sessions, or setting up a knowledge base that users can refer to.

Monitor User Activity: To maintain users in the system, it is important to monitor their activity regularly. This will help you identify any potential security issues, as well as help you keep track of user access and usage patterns.

Disable or Delete User Accounts: If a user no longer requires access to the system, their account should be disabled or deleted. This will help you maintain the security and integrity of the system, as well as free up space and resources for other users.

SET PERMISSION TO USERS

In Unix/Linux systems, permissions are set for files and directories to control access to them. There are three types of permissions: read, write, and execute, and these can be assigned to three categories of users: the owner of the file or directory, members of the group that owns the file or directory, and all other users. To set permissions for a file or directory in Unix/Linux, you can use the chmod command. The chmod command allows you to set permissions using numeric or symbolic notation.

Numeric notation: Numeric notation involves assigning a number to each type of permission (read, write, and execute) and each category of users (owner, group, and others). The numbers range from 0 to 7, with each number representing a combination of permissions. For example, the number 7 represents read, write, and execute permissions, while the number 0 represents no permissions.

To set permissions using numeric notation, use the following command:

Bash Copy code

```
chmod xyz file_or_directory
```

In this command, x represents the permissions for the owner, y represents the permissions for the group, and z represents the permissions for others. Each digit is a sum of the following values: 4 for read, 2 for write, and 1 for execute. For example, if you want to give the owner read, write, and execute permissions, the group read and execute permissions, and others only read permissions, you would use the following command:

```
Bash                                                    Copy code

chmod 750 file_or_directory
```

Symbolic notation: Symbolic notation involves using letters to assign permissions. The letters used are u for the owner, g for the group, and o for others. a is used to represent all users. The permissions themselves are represented by r for read, w for write, and x for execute.

To set permissions using symbolic notation, use the following command:

```
Css                                                     Copy code

chmod [who][operator][permissions] file_or_directory
```

In this command, who represents the user category (owner, group, others, or all), operator represents the operation to be performed (+ to add permissions, - to remove permissions, or = to set permissions explicitly), and permissions represents the permissions to be set.

For example, to give the owner read, write, and execute permissions, and remove write permissions from the group and others, you would use the following command:

```
Bash                                                    Copy code

chmod u=rwx, g-w, o-w file_or_directory
```

These are just a few examples of how to set permissions in Unix/Linux. For more information on using the chmod command and setting permissions, consult the documentation for your specific Unix/Linux distribution.

FILE SYSTEM AND DISK ADMINISTRATION

File system and disk administration is an important aspect of Unix/Linux system administration. It involves managing file systems and disks to ensure reliable and efficient data storage and retrieval. Some of the key tasks involved in file system and disk administration include:

CREATE FILE SYSTEMS

To create a new file system in Unix/Linux, you can use the "mkfs" command with the appropriate options to specify the type of file system you want to create and the device or partition on which it should be created. Here are the basic steps to create a file system:

1. **Determine the device or partition:** Use the "fdisk" command to view the available disks and partitions on the system, and determine the device or partition on which you want to create the file system. For example, the device may be "/dev/sda1" or "/dev/hda2".
2. **Unmount the device or partition:** If the device or partition is currently mounted, you must unmount it before creating the file system. You can use the "umount" command with the device name or mount point to unmount the device or partition.
3. **Create the file system:** Use the "mkfs" command with the appropriate options to create the file system on the device or partition. The options may vary depending on the type of file system you want to create. For example, to create an ext4 file system on "/dev/sda1", you can use the following command:

Bash Copy code

```
mkfs.ext4 /dev/sda1
```

4. **Mount the file system:** After the file system has been created, you can mount it using the "mount" command with the appropriate options. The options may vary depending on the type of file system and the desired mount point. For example, to mount the newly created ext4 file system at "/mnt/mydata", you can use the following command:

Bash Copy code

```
mount -t ext4 /dev/sda1 /mnt/mydata
```

These are the basic steps to create a file system in Unix/Linux. However, it is important to note that creating a file system will erase all existing data on the device or partition, so you should back up any important data before proceeding.

MANAGE AND REPAIR FILE SYSTEMS

Managing and repairing file systems in Unix/Linux involves a variety of commands and utilities. Here are some common tasks involved in managing and repairing file systems:

1. **Checking the file system:** The "fsck" command can be used to check the file system for errors and repair any issues that are found. This command should be run regularly to prevent data loss or corruption.
2. **Mounting file systems:** The "mount" command is used to mount file systems and make them available for use. The command must be used with the appropriate options to mount the file system correctly.
3. **Unmounting file systems:** The "umount" command is used to unmount file systems before they are removed or shut down. It is important to unmount file systems before removing or unplugging any storage devices to prevent data loss or corruption.
4. **Creating file systems:** The "mkfs" command is used to create a new file system on a storage device. The command must be used with the appropriate options to create the correct type of file system.
5. **Resizing file systems:** The "resize2fs" command is used to resize an existing file system. This command can be used to increase or decrease the size of a file system, but it requires some care to avoid data loss.
6. **Monitoring disk space usage:** The "df" command is used to display information about disk space usage on the file system. This command can be used to identify areas where disk space is running low and take action to free up space.

7. **Defragmenting file systems:** The "defrag" command is used to defragment file systems to improve performance. This command is typically used on file systems that have a high level of fragmentation, such as those used for large file servers.

7.5. SYSTEM ACCOUNTING AND PERFORMANCE MONITORING

System accounting and performance monitoring are important aspects of Unix/Linux system administration. System accounting involves tracking system usage and resource consumption, while performance monitoring involves monitoring system performance and identifying and troubleshooting performance issues. Here's an overview of these two topics:

System Accounting: System accounting involves tracking system usage and resource consumption. This can help system administrators identify trends in system usage, track system activity, and troubleshoot issues related to system resources. Some of the key system accounting tasks include:

1. **Enabling system accounting:** System accounting can be enabled by installing and configuring accounting software such as "acct" or "sar".
2. **Tracking user activity:** System accounting can be used to track user activity, including login/logout times, CPU usage, disk usage, and network activity.
3. **Tracking system usage:** System accounting can also be used to track system usage, including CPU usage, memory usage, disk usage, and network activity.
4. **Generating accounting reports:** System accounting data can be used to generate reports that provide insights into system usage and resource consumption. These reports can be used to identify system bottlenecks and improve system performance.

Performance Monitoring: Performance monitoring involves monitoring system performance and identifying and troubleshooting performance issues. Some of the key performance monitoring tasks include:

1. **Identifying performance metrics:** Performance metrics such as CPU usage, memory usage, disk usage, and network activity can be monitored using tools such as "top", "vmstat", "sar", and "iostat".
2. **Monitoring application performance:** Application performance can be monitored using tools such as "strace" and "lsof" to identify problematic processes and diagnose performance issues.
3. **Analyzing log files:** Log files can be analyzed using tools such as "grep", "sed", and "awk" to identify performance issues and troubleshoot system problems.
4. **Tuning system settings:** System settings can be tuned using tools such as "sysctl" and "ethtool" to optimize system performance.

Kernel monitoring: Linux/Unix provides tools for monitoring the kernel, including "dmesg", "syslog", and "journalctl". These tools can be used to monitor system messages, debug kernel issues, and diagnose system problems.

Performance tuning: Linux/Unix provides a variety of tools for performance tuning, including "sysctl" and "tuned". These tools can be used to adjust system settings to optimize performance for specific workloads.

Log file analysis: Linux/Unix generates log files that contain information about system events, errors, and user activity. Tools such as "grep", "awk", and "sed" can be used to analyze log files and identify system issues.

Network monitoring: Linux/Unix provides several tools for monitoring network performance, including "tcpdump", "wireshark", and "iftop". These tools can be used to capture and analyze network traffic, monitor network connections, and troubleshoot network issues.

DESCRIBE UNIX/LINUX SYSTEM BASIC ACCOUNTING

Unix/Linux systems have basic accounting features that allow system administrators to monitor and track system usage by users and processes. These features include:

1. **Process accounting:** Unix/Linux systems can be configured to track system usage by individual processes, including the amount of CPU time, memory usage, and disk activity. The "acct" package provides utilities such as "accton" and "sa" that enable system administrators to enable and configure process accounting, as well as generate reports on system usage.
2. **User accounting:** Unix/Linux systems can also track system usage by individual users, including the number of logins, the amount of CPU time and memory usage, and the commands executed. The "utmp" and "wtmp" files in the "/var/run" directory contain information about user logins and logouts, and the "last" command can be used to generate reports on user activity.
3. **System resource accounting:** Unix/Linux systems can also track system resource usage, including disk space, memory usage, and network activity. The "df" command can be used to display information about disk space usage, the "free" command can be used to display memory usage, and the "netstat" command can be used to display network activity.
4. **Quotas:** Unix/Linux systems also support quotas, which allow system administrators to limit the amount of disk space or number of files that individual users can use. The "quota" command can be used to enable and configure quotas, as well as generate reports on usage.

These basic accounting features can be useful for system administrators to monitor and manage system resources, identify potential issues or security concerns, and enforce usage policies.

SET UP ACCOUNTING SYSTEM IN UNIX/LINUX

To set up an accounting system in Unix/Linux, you can follow these basic steps:

1. **Install the accounting software:** Depending on your Unix/Linux distribution, you may need to install the accounting software package. For example, on Debian/Ubuntu, you can install the "acct" package using the command:

Arduino Copy code

```
sudo apt-get install acct
```

2. **Enable process accounting:** To enable process accounting, you can use the "accton" command with the desired options. For example, to enable process accounting and store the accounting data in the "/var/log/account/pacct" file, you can use the following command:

Bash Copy code

```
sudo accton /var/log/account/pacct
```

3. **Configure user accounting:** By default, user accounting is enabled on most Unix/Linux systems. However, you may need to configure the location of the "wtmp" and "utmp" files used to store user accounting data. These files are typically located in the "/var/run" directory. You can also configure the "logrotate" utility to rotate the user accounting data files on a regular basis.
4. **Configure system resource accounting:** To enable system resource accounting, you can use various commands and utilities, such as "df", "free", and "netstat", to monitor and track disk space, memory usage, and network activity. You can also use the "quota" command to configure quotas on disk space usage and number of files for individual users.
5. **Generate accounting reports:** Once the accounting system is set up and configured, you can generate reports on system usage using various utilities, such as "sa" and "last". For example, the "sa" command can be used to generate daily, weekly, or monthly reports on process accounting data stored in the "/var/log/account/pacct" file.

These are the basic steps to set up an accounting system in Unix/Linux.

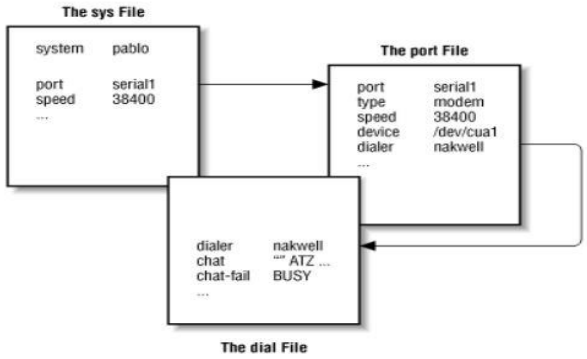
7.6. DEVICE AND MAIL ADMINISTRATION

Device and mail administration are important tasks for Linux/Unix system administrators. Here are some of the key features and commands for device and mail administration:

1. **Device administration:** Linux/Unix provides a variety of tools for device administration, including "udev", "lsusb", and "lspci". These tools can be used to detect and configure hardware devices, manage device drivers, and troubleshoot device issues.
2. **File system management:** Linux/Unix provides tools such as "mount" and "umount" for managing file systems and storage devices. These commands can be used to mount/unmount storage devices such as USB drives, external hard drives, and network storage devices.
3. **Printer management:** Linux/Unix provides tools for managing printers, including "cups" and "lp". These tools can be used to add and configure printers, manage print queues, and troubleshoot printing issues.
4. **Mail administration:** Linux/Unix provides several tools for managing mail services, including "sendmail", "postfix", and "exim". These tools can be used to configure and manage mail servers, monitor mail queues, and troubleshoot mail issues.
5. **User administration:** Linux/Unix provides tools such as "useradd", "usermod", and "userdel" for managing user accounts. These commands can be used to add and remove user accounts, modify user account settings, and manage user groups.
6. **Group administration:** Linux/Unix provides tools such as "groupadd", "groupmod", and "groupdel" for managing user groups. These commands can be used to add and remove user groups, modify group settings, and manage group membership.

7.7. UUCP AND FTP SERVICES ADMINISTRATION

UUCP (Unix to Unix Copy) and FTP (File Transfer Protocol) are two popular file transfer protocols used in Linux/Unix systems. Here are some key features and commands for UUCP and FTP services administration:



UUCP administration: UUCP is a file transfer protocol that is used to transfer files between Unix/Linux systems. To use UUCP, the "uucp" package must be installed on both the sending and receiving systems. Some common commands for UUCP administration include:

- uucp: used to copy files between systems using UUCP protocol
- uucico: used to initiate UUCP file transfers between systems
- uux: used to execute commands on a remote system using UUCP protocol
- uustat: used to check the status of UUCP file transfers

uuname: used to display the name of the local system and the systems it is connected to via UUCP

EXPLAIN UUCP AND FTP SERVICES

UUCP services : UUCP (Unix-to-Unix Copy Protocol) provides a suite of services for file transfer, email delivery, and remote command execution between Unix systems. Here are the key services offered by UUCP:

File Transfer: UUCP allows users to transfer files between Unix systems. This includes copying files from one system to another, synchronizing directories, and managing remote file systems. UUCP supports both single file transfers and batch transfers of multiple files.

Email Delivery: UUCP includes facilities for sending and receiving email messages between Unix systems. It supports email forwarding and delivery to remote mailboxes. UUCP email uses a store-and-forward model, where messages are queued at intermediate systems and delivered when connections are available.

Remote Command Execution: UUCP enables users to execute commands on remote Unix systems. This allows for remote management and administration of systems. Users can run commands on a remote system and retrieve the output on their local system.

Configuration and Routing: UUCP requires manual configuration and routing setup on each participating system. Configuration files specify the connections, system names, dial-up parameters, and routing paths. This allows UUCP to determine the best route for file transfers and email delivery between systems.

Batch Jobs and System Backup: UUCP supports the execution of batch jobs on remote systems. This allows users to schedule tasks, such as system backups or data processing, on remote systems using UUCP commands. This feature enables automation and distributed computing in a Unix environment.

Remote Printing: UUCP can be used for remote printing, where print jobs are submitted from one Unix system and printed on a printer connected to another system. This enables centralized printing and sharing of printers across different Unix systems.

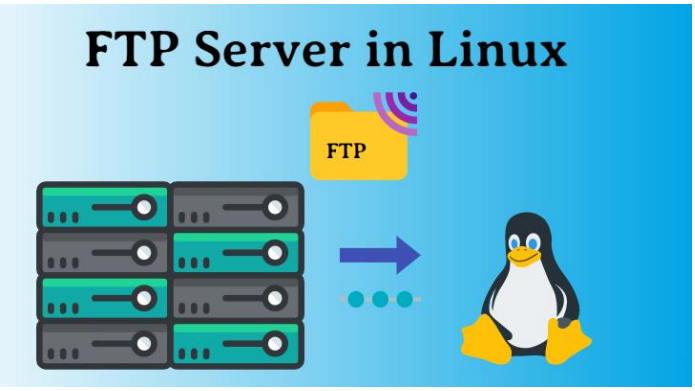
Collaboration and News Distribution: UUCP provides capabilities for collaborative file sharing and news distribution within the Unix community. Users can exchange files, publish news articles, and distribute software updates using UUCP's file transfer and email services.

DESCRIBE FTP PROTOCOL

FTP (File Transfer Protocol) is a standard network protocol used for transferring files between a client and a server over a TCP/IP-based network, such as the internet. It provides a simple and reliable method for file exchange.

FTP administration: FTP is a popular protocol used to transfer files between systems over a network. Linux/Unix systems provide several tools for managing FTP services, including:

- vsftpd: a popular FTP server for Linux/Unix systems
- ftp: a command-line client for FTP
- sftp: a secure version of FTP that uses SSH for encryption
- ftpusers: a file that lists users who are not allowed to use FTP
- vsftpd.conf: a configuration file for the vsftpd FTP server



Some common tasks related to FTP administration include configuring FTP servers, creating and managing FTP user accounts, setting permissions and restrictions on file transfers, and monitoring FTP activity.

FTP SERVICES

FTP (File Transfer Protocol) provides a set of services for transferring files between a client and a server over a TCP/IP network. Here are the main services offered by FTP:

File Transfer: The primary purpose of FTP is to facilitate the transfer of files between a client and a server. Users can upload files from their local system

to the server (put), download files from the server to their local system (get), or perform bi-directional file transfers. FTP supports the transfer of both individual files and entire directories.

Directory Listing: FTP allows clients to obtain a directory listing of files and directories on the server. Clients can request the listing of the current directory or specify a specific directory to retrieve its contents. The directory listing provides information such as file names, sizes, timestamps, and permissions.

File Manipulation: FTP provides commands for manipulating files on the server. Users can rename files, delete files, create directories, remove directories, and change file permissions on the server using FTP commands. These operations enable basic file management capabilities on the server side.

Authentication: FTP incorporates authentication mechanisms to verify the identity of users accessing the FTP server. Users typically provide a username and password to log in to the FTP server. Depending on the server configuration, anonymous FTP access may also be allowed, which allows users to connect to the server without providing any credentials.

Error Handling and Logging: FTP includes mechanisms for handling errors and reporting them to the client. If an error occurs during a file transfer or any other operation, the FTP server sends an appropriate error code to the client, indicating the nature of the error. Additionally, FTP servers usually maintain logs that record the activities and events related to file transfers and client connections.

Passive and Active Mode: FTP supports two modes of operation for data transfer: passive mode and active mode. In passive mode, the server provides the client with an IP address and port for establishing a data connection. The client initiates the connection to the server for data transfer. In active mode, the server initiates the data connection to the client. Passive mode is commonly used when the client is behind a firewall or NAT (Network Address Translation) device.

Security and Encryption: By default, FTP transfers data in plain text, making it susceptible to interception. To enhance security, FTP can be secured using SSL/TLS encryption, known as FTPS (FTP over SSL/TLS). FTPS encrypts both control and data connections, protecting the confidentiality and integrity of the transferred data.

Scripting and Automation: FTP supports scripting and automation capabilities, allowing users to automate file transfer tasks. Users can create scripts or batch files that include FTP commands, enabling the execution of multiple file transfer operations in a sequence or as part of a scheduled task.

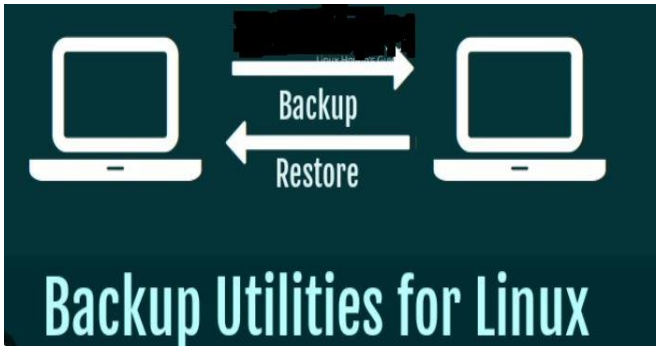
SETUP AND ADMINISTER FTP SERVICES

Setting up and administering FTP (File Transfer Protocol) services involves configuring an FTP server to allow users to connect and transfer files. Here's a general overview of the steps involved:

- Choose an FTP Server Software
- Install and Configure the FTP Server
- Configure User Access
- Configure Security
- Set Permissions and Directories
- Start and Test the FTP Server
- Monitor and Maintain

7.8. BACKING UP AND RESTORING THE SYSTEM

Backing up and restoring the system is an important task for Linux/Unix system administrators. It helps to protect critical data in the event of hardware failure, software corruption, or other disasters. Here are some key features and commands for backing up and restoring the system:



1 **Backup tools**: Linux/Unix provides several backup tools that can be used to create backups of the system or individual files and directories. Some popular backup tools include:
tar: a command-line tool used to create and extract archive files
rsync: a tool used for incremental backup and file synchronization
dump and restore: tools used to backup and restore file systems, including the root file system

2 **Backup media**: Backups can be stored on a variety of media, including hard drives, tape drives, external storage devices, and cloud storage services. It is important to choose a backup media that is appropriate for the size of the backup and the available resources.

3 **Backup schedule**: It is important to establish a regular backup schedule to ensure that critical data is backed up on a regular basis. A backup schedule can be based on time intervals, events (such as system updates), or other criteria.

4 **Restoration**: In the event of a system failure, data can be restored from backups using tools such as tar, rsync, and restore. It is important to test backup and restoration procedures to ensure that data can be restored successfully in the event of a disaster.

Some common tasks related to backup and restoration in Linux/Unix systems include creating and managing backup schedules, selecting backup media, monitoring backup activity, testing restoration procedures, and troubleshooting backup and restoration issues.

EXPLAIN PURPOSE OF BACKUP

The purpose of backup is to create copies of data, files, or entire systems in order to protect them against data loss, accidental deletion, hardware failures, software corruption, natural disasters, or any other unforeseen events.

BACK UP AND RESTORE UNIX/ LINUX SYSTEM

To back up and restore a UNIX/Linux system, you can follow these general steps:

1. Determine what data you need to back up: Identify the critical files and directories that you want to include in your backup. This may include system files, user files, configuration files, and databases.
2. Choose a backup method: There are several backup methods available, such as full backups, incremental backups, and differential backups. Each method has its advantages and disadvantages, so choose the one that suits your needs.
3. Select a backup destination: Determine where you want to store your backups. This can be an external hard drive, network storage, cloud storage, or another server. Ensure that you have enough storage space for your backups.
4. Perform the backup: Depending on the backup method you chose, execute the backup command or use a backup tool. The specific command or tool will vary based on your Linux distribution. Here are a few common examples:
   - rsync: A powerful utility for incremental backups that only copies changed files.
   - tar: A command-line tool for creating compressed archive files.
   - cpio: A command-line tool for creating or extracting archives.
   - Amanda: An open-source backup and recovery software suite.
   - Bacula: A network backup solution with a client-server architecture.
   - Consult the documentation or man pages for the specific command or tool you choose to understand its usage and options.
5. Verify the backup: After the backup completes, verify that the backup files are created successfully and contain the data you intended to back

up. This step ensures that you have a valid backup in case you need to restore your system.

TO RESTORE THE SYSTEM, FOLLOW THESE STEPS:

1. Boot into a rescue environment: If your system is not bootable, use a live CD/DVD or a rescue disk to boot into a minimal environment.
2. Mount the backup destination: Mount the backup storage location where your backups are stored. If you used an external drive, network storage, or cloud storage, make sure it is accessible.
3. Restore the system files: Copy the backed-up system files and directories to their respective locations on your system. This may include the root directory, configuration files, and any other important system files.
4. Restore user files: Copy the backed-up user files to their original locations or to the appropriate user directories. These files may include documents, settings, and user-specific configurations.
5. Verify the restore: After the restore process is complete, verify that the system is functioning as expected. Test critical functionality, services, and applications to ensure everything is working correctly.

Multiple Choice Questions

Q.1: Which command is used to create a new user account in a Unix/Linux system?

(a) adduser (b) useradd

(c) createuser (d) newuser

Q.2: Which command is used to change the ownership of a file or directory in Unix/Linux?

(a) chown (b) chmod

(c) chgrp. (d) own

Q.3: Which command is used to display the current system date and time in Unix/Linux?

(a) date (b) Time

(c) Datetime (d) Now

Q.4: Which command is used to list the currently running processes in Unix/Linux?

(a) ls (b) ps

(c) top (d) process

Q.5: Which command is used to shut down a Unix/Linux system?

(a) shutdown (b) poweroff

(c) halt (d) all of the above

Q.6: Which file contains the system-wide environmental variables in Unix/Linux?

(a) /etc/hosts (b) /etc/environment

(c) /etc/profile (c) /etc/variables

Q.7: Which file contains the list of valid usernames on a Unix/Linux system?

(a) /etc/passwd (b) /etc/users

(c) etc/groups (d) /etc/shadow

Q.8: Which command is used to display the current logged-in users in Unix/Linux?

(a) users (b) listusers

(c) who (d) loggedin

Q.9: Which command is used to format a disk in Unix/Linux?

(a) format (b) mkfs

(c) diskformat (d) mkfs

Q.10: Which command is used to display the disk space usage of a directory in Unix/Linux?

(a) df (b) ls

(c) du (d) diskusage

ANSWER KEY

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

Short Questions

1. Describe Unix/Linux system administration?
2. Describe tasks of Unix/Linux system administration?
3. Identify hardware requirements of Unix/Linux system?
4. Describe how to Install Unix/Linux?
5. How to add and maintain new users in the system?
6. Describe set permission to the users?
7. How to create a file system?
8. How to manage and repair file systems?
9. Describe performance monitoring in Unix/Linux?
10. Describe services facility provided by Unix/Linux?
11. Describe device administrative tasks?
12. How to Install printer to Unix/Linux system?
13. Describe email facility of Unix/Linux?
14. Describe mail transfer agents?
15. UUCP stands for?
16. Describe UUCP administration?
17. Describe FTP administration
18. Describe FTP protocol?
19. Explain purpose of backup?
20. Describe Back -up and restore Unix/Linux system

Long Questions

1. Explain the System administration Task?
2. What is resource and user administration?
3. Explain file system and disk administration?
4. Explain the system accounting and performance monitoring?
5. Explain UUCP and FTP services?
6. What are the steps involved in backing-up and restoring the system?

Bibliography

1. Operating System Concepts, 5Ed., A. Silverschatz and P. Galvin, Addison-Wesley Publishing Co.
2. Unix/Linux Unleashed, 3Ed, Robin Burk, et al., Sams Publishing
3. The Linux User's Guide, Larry Greenfield
4. Unix System Management, Robert King Ables
5. Red Hat Linux 6.0, Red Hat Software, Inc.
6. Hand-on Unix: A Practical Guide with the Essentials, Sobell
7. The Linux Users' Guide, Larry Greefield
8. UNIX-The Text Book by Mansoor Sarwar.