GURUKRUPA TECHNICAL SCHOOL

Narasinghpur, Cuttack

# Lecture Notes

# On

# COMPUTER APPLICATION

1st & 2nd Semester

## DEPARTMENT OF

## COMPUTER SCIENCE & ENGINEERING

Prepared by:

Lect. Susant Ku. Nayak

# COMPUTER APPLICATION

## 1. COMPUTER ORGANISATION

Introduction to Computer
Evolution of Computers
Generation of Computers
Classification of Computers
Basic Organisation of Computer (Functional Block diagram) Input Devices, CPU & Output Devices.
Computer Memory and Classification of Memory

## 2. COMPUTER SOFTWARE

Software concept

System software

Application software

Overview of Operating System

Objectives and Functions of O.S

Types of Operating System

Batch Processing, Multiprogramming, Time Sharing OS

Features of DOS, Windows and UNIX

Programming Languages

Compiler, Interpreter

Computer Virus

Different Types of computer virus

Detection and prevention of Virus

Application of computers in different Domain

## 3. COMPUTER NETWORK AND INTERNET

Networking concept, Protocol, Connecting Media,

Date Transmission mode

Network Topologies,

Types of Network

Networking Devices like Hub, Repeater, Switch, Bridge, Router, Gateway & NIC

Internet Services like E-Mail, WWW, FTP, Chatting, Internet Conferencing, Electronic

Newspaper & Online Shopping

Different types of Internet connectivity and ISP

## 4. FILE MANAGEMENT AND DATA PROCESSING

Concept of File and Folder

File Access and Storage methods.

Sequential, Direct, ISAM

Data Capture, Data storage

Data Processing and Retrieval

## 5. PROBLEM SOLVING METHODOLOGY

Algorithm, Pseudo code and Flowchart

Generation of Programming Languages

Structured Programming Language

Examples of Problem solving through Flowchart

**6. OVERVIEW OF C PROGRAMMING LANGUAGE**

Constants, Variables and Data types in C

Managing Input and Output operations.

Operators, Expressions, Type conversion & Typecasting

Decision Control and Looping Statements (If, If-else, If-else-if, Switch, While, Do-while, For,Break, Continue & Goto)

Programming Assignments using the above features.

**7. ADVANCED FEATURES OF C**

Functions and Passing Parameters to the Function (Call by Value and Call by Reference)

Scope of Variables and Storage Classes

Recursion Function and Types of Recursion

One Dimensional Array and Multidimensional Array

String Operations and Pointers

Pointer Expression and Pointer Arithmetic

Programming Assignments using the above features.

Structure and Union (Only concepts, No Programming)

**Books Recommended**

1. Computer Fundamentals and Programming in C by Reema Thareja, Oxford Unversity Press

2. Programming in ANSI C by A.N Kamthane, Pearson Education

3. Computer Application by Kalyani Publisher

4. Let us C by Y Kanetkar, BPB

5. Computer Fundamentals, by E. Balaguruswamy, TMH

# CHAPTER -1

## A History of Information Technology and Systems

- **Four basic periods**

  Characterized by a principal technology used to solve the **input, processing, output and communication problems** of the time:

  A. Premechanical,

  B. Mechanical,

  C. Electromechanical, and

  D. Electronic

### Introduction

A computer is an electronic machine/device which is having ability to –

* Accept user supplied data.
* Input, store and execute program to process the data.
* Output results of the processing in the user defined format.

In other words, a computer can be defined as an electronic device to accept, process and output user given data and provide result in a desired format.

### Evolution of Computer

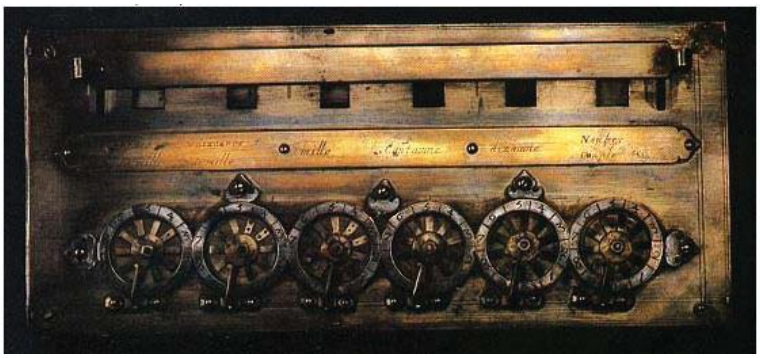
- Abacus
- Pascaline
- Difference engine
- Punched card equipment
- ABC
- UNIVAC - I

Abacus

- The present day computers are a result of an evolutionary process which started way back in 500 B.C. when Egyptian used a machine which is an early form of **Abacus.**
- However the present form of **Abacus** is attributed to the Chinese and Japanese. This is a machine, which was used for addition, subtraction, multiplication and division operation.

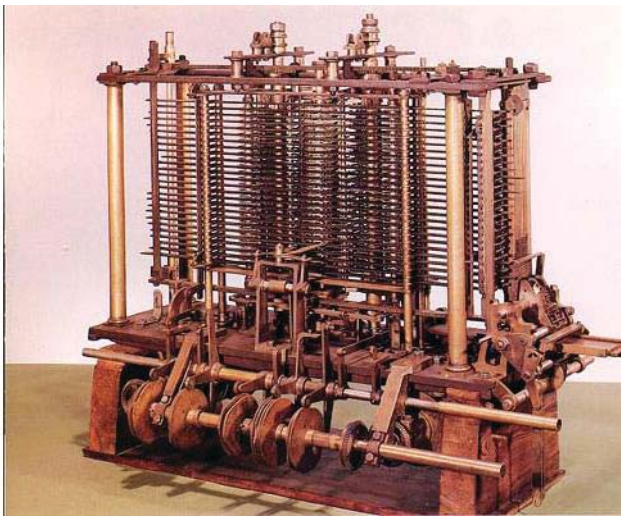
Pascaline

- In 1645 a device known as **Pascaline** was invented by French mathematician Blaise Pascal. The machine was also used per addition and subtraction purpose. The device was operated by dialing a set of wheels. In 1671 Leibniz improved on Pascal's adding machine and invented the Leibniz's Calculator



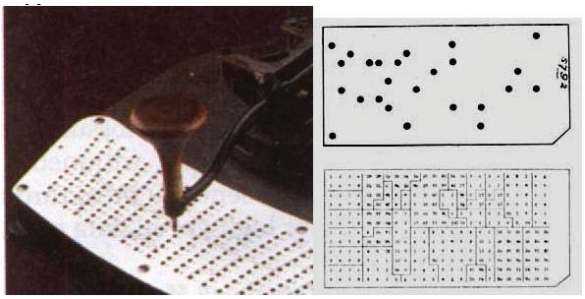
Difference engine

- In 1822 Charles Babbage invented a **Difference Engine.**
- The purpose of this device was to calculate the roots of polynomial equations and prepare astronomy table for the British Navy. He upgraded this to, invent an Analytical engine, which could store program instructions initially coded on punched cards and subsequently shared internally. Therefore Charles Babbage is known as the father of computers.



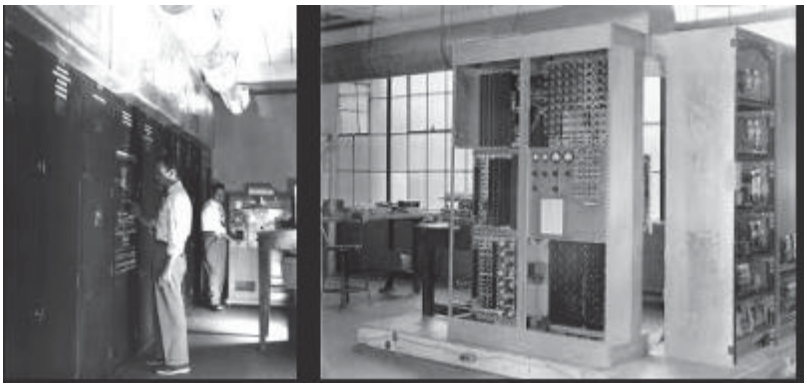
Punched card equipment

- In 1890 Dr. H. Hollerith developed **punched card equipment** (Fig 1.6). This equipment read the holes punched in the card and mechanically performed the statistical analysis.



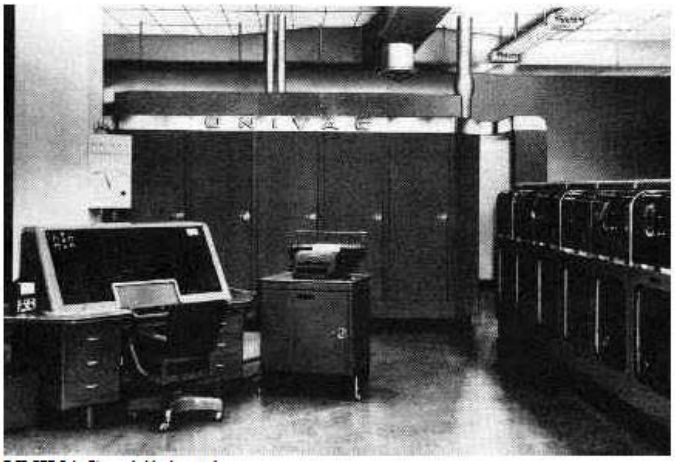
ABC (**Atanasoff-Berry Computer )**

- The first pure electronic computer was invented by J. V. Atanasoff and C. Berry which is known as **Atanasoff-Berry Computer or ABC**.
- It used vacume tubes for both data storage and data computation.
- Subsequently Electronic Numerical Integrator and Calculator (ENIAC) was designed and accepted as the general purpose computer (Fig 1.9).



UNIVAC

- In 1945 John Von Neumann first gave the idea of sharing the same internal memory for storing both data and instruction, which was subsequently adopted in every computer organization.
- On this principle subsequently **Universal Automatic Computer (UNIVAC-1)** was invented




The period from 1954 to 1960 is called first generation computers.
These computers had the following common features :

- Vaccume tubes were used as the main switching unit.
- The computers were bulky.
- The computers required a lot of power for operation.
- The use of computers were mainly restricted to some research organization.
- The heat dissipation by these computer during operation was too high.

In 1947 Bell labs invented solid-state transistors. The second generation of computer begun from 1960.
These computers had the following common features.

- Transistors were used as the main switching device.
- They were small in size as compared to first generation computers.
- They required comparatively less amount of power.

- The power requirement and heat decipation was relatively low.
- These computers were cheaper for manufacturing.

Some popular computers of Second generation were-

- IBM 1620
- Henery Well 800
- Burroughs 5000
- UNIVAC III
- CDC 6600
- GE 65
- RCA Spectra 70

In fact this generation of computers were relatively affordable enabling many to purchase computers for the first time.

Third generation of computers began with the introduction of IBM 360 in 1965.

- This generation of computers were having the following common features.
- The concept of general purpose computing was introduced for the first time.
- They used integrated circuits on wafer chips instead of transistors.
- Different models / versons of the computer were available for different level of usage.
- These computers wee less bulky, required less power and dissipate less heat during operation as compared to their previous generations.

- The revolution in IC manufacturing technology was incorporated in the fourth generation computers. They used Large Scale Integration (LSI) and Very Large Scale Integration (VLSI) ICs as the main switching device. The ICs

manufactured using this technology is called microprocessor, which acts as the main components of a forth generation computer.

Now-a-days research is going for inventing fifth generation computers having following features.

- It should be very small in size and powerful in processing.
- It should be portable and require less amount of power.
- It should have reasonable high amount of primary/secondary memory.
- It should have some artificial intelligence feature.
- It may be based on the latest architecture to support parallel processing and real time processing.
- It should support multimedia computing & artificial intelligence.

Summary of Generation of computers

| **Features** | **1st Generation Computer** | **2nd Generation Computer** | **3rd Generation Computer** | **4th Generation Computer** |
|---|---|---|---|---|
| **Main Switching device** | Vacume Tube | Transistor | Integrated Circuit (IC) | Large Scale Integration (LSI) & VLSI |
| **Component size** | 6000 circuits/ Cubic foot | 10000 circuits/ Cubic foot | 10 millions circuits/ Cubic foot | Over 500 billions circuits/ Cubic foot |
| **Number of instructions/ sec.** | 250 | 30,000 | 2,00,000 | 80 Millions |
| **Meantime between failure** | Hour | Days | Weeks | Months |
| **Internal memory capacity** | 4000 Characters | 30000 Characters | 5,12,000 characters | Over 4 million characters |

**Classification of Computer**

- All the modern computers are broadly classified into the following three categories.

(i) Analog Computer.
(ii) Digital Computer and
(iii) Hybrid Computer.

Analog computers

- Are mostly used in industries in process control activities.
- These computers work on analog data such as variation in temperature, pressure, speed, voltage etc.
- They are not general purpose computers, rather they are specific to a particular application area. Therefore the cost of such computer vary from application to application depending on the complexity.
- The use of such computers are very limited.

Digital computers

- These computers are general purpose computers, which work on digital / binary data.
- The speed and accuracy with which these computers work are very high.
- Digital computer are also having several ranges form super computers to personal computers.

Hybrid computers

- Practically Hybrid computer are used to control the entire process.
- The analog feature of such computer enables it to measure the physical quantities such a temperature, pressure, voltage level etc. and convert them to digital data.
- These data are then processed by the computer by using its digital data processing capability.
- The output form this computer may be taken in a paper as hardcopy, may be seen on a display device or may be converted into analog form to automatically control various process.

Digital computers are classified into the following four categories :

- Super Computers.
- Mainframe Computers.
- Mini Computers.
- Micro Computer

Super computers

- These computers are specifically designed to maximize the processing of floating point instructions.

- This is possible because of parallel processing technique which implements multiple processors to work in parallel manner.
- Such computers are very expensive and used in very high-end numerical processing, geographical information system, etc.
- some of the popular super computers are Cray, Param, Anupam etc.
- The speed of processing of super computers are measured in GFLOPS i.e., Giga Floating Point Operations Per Second.
- These computer used their own operating system and programming language and hence vary from computer to computer.

Mainframe computers

- These computers are intended for substantial high volume data processing. These computers are caracterized by–
- Large primary memory.
- Substantial processing capabilities. (MIPS)
- Substantial amount of peripheral devices that can be attached.
- A high data communication capability i.e. ability to connect thousands of terminals.
- Wide variety of memory size and terminal support option.
- Ability to handle large type computer application.

Application of Mainframe Computer

- space research,
- university connectivity,
- wide area network (WAN) implementation etc.

specification of mainframe computers.

- Processing speed – 30 to 100 million instruction per second (MIPS)
- Word length – More than 64 bits.
- I/O device –Wide range of peripheral devices.
- Internal Storage – More than 1 GB.

Application of Mainframe Computer

- space research,
- university connectivity,
- wide area network (WAN) implementation etc.

specification of mainframe computers.

- Processing speed – 30 to 100 million instruction per second (MIPS)
- Word length – More than 64 bits.
- I/O device –Wide range of peripheral devices.
- Internal Storage – More than 1 GB.

Typical features Mini Computers

- Fairly large primary memory.
- Medium scale processing capability i.e., lesser than mainframe but higher than personal computers.
- Can connect upto 500 terminals on LAN.
- Supports wide range of application areas.
- Affordable, unlike mainframe computers by small business organization.

The application of mini computers

- The field of engineering and scientific organizations,
- Educational Institutes,
- Universities,
- small/medium business organizations.

These computers are mainly used for medium or large volume data processing activity.

specification of mini computer

- Processing speed – 10 to 30 MIPS.
- Word length – 32 bits.
- I/O device – Wide range of I/O devices can be connect.
- Internal storage – 66 MB to 512 MB.

Major global manufacturers of mini computers

  - Burroughs
  - DEC
  - IBM
  - Hewlett-Packard
  - Prime
  - Wang etc.

Micro computers

- This is the smallest and least expensive computers are or personal computers popularly known as PC.

**Typical features.**

- These computers are portable.
- They require minimum power.
- Processing power is appropriate for handling most of the tasks.
- Memory capacity is sufficient to handle most of the tasks.
- Ease of use and support to various types of operating systems and application softwares.
- Affordable price tag.

- These micro computers are further classified into three categories i.e., PC, PC-XT and PC-AT.

Typical Specifications of PC

- Processor – I 8086 / I 8088 micro processor.
- Memory – 640 KB of RAM
- – Two 360 K Floppy Disk drive.
- Numerical Processor – I 8087.
- System bus –8 bit data bus & 16 bit address bus.
- Clock speed – 8 MHz.
- A PC-XT or Personal Computer with extended technology is an upgradation of the PC. It is having all the features discussed above. Apart from these the concept of secondary memory / mass storage in the form of hard disk drive, was introduced for the first time.
- The present day PC are PC-AT or personal computer with advanced technology.

Features of PC-AT

- Processor –I 80386 / 80486 / Pentium.
- Memory – 2 MB to 512 MB.
    - –Floppy disk drive 1.44 MB
    - –Hard disk drive 1.2 MB to 80 GB.
- System bus–32 bit to 64 bit.
- Clock speed–Upto 3 GHz
- Operating System –MS-DOS, Windows, UNIX, Linux etc.

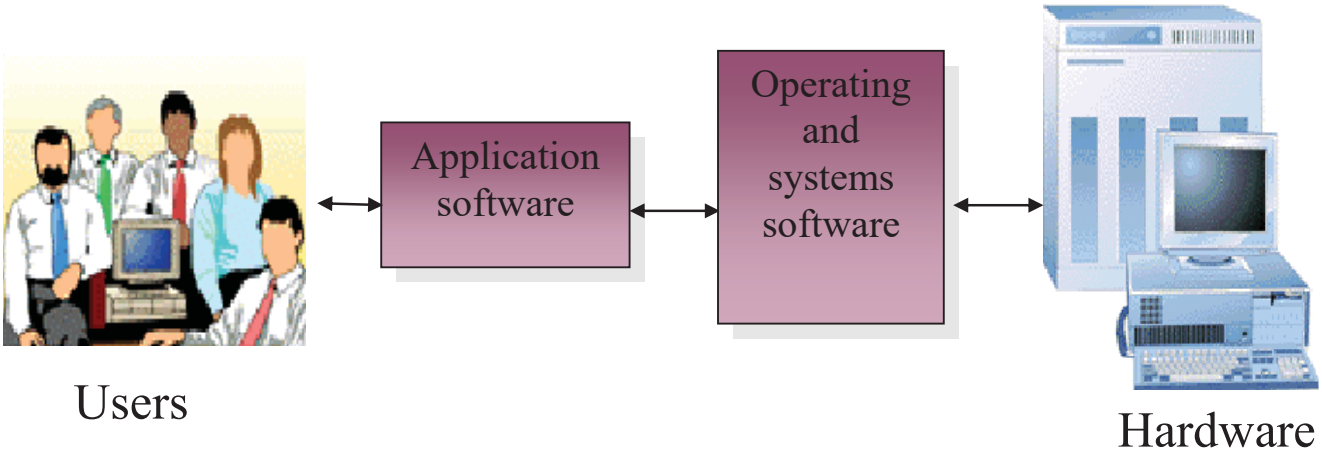
# CHAPTER - 2

## Software concept

- A software is a program which helps the human user to give instruction to computer hardware.

Classes of Software

- Systems software
- Application software




## System software

System Softwareis a set of programs that manage the -

- Resources of a computer system.
- System Software is a collection of system programs that perform a variety of functions such as -
  - File Editing
  - Resource Accounting
  - I/O Management
  - Storage, Memory Management access

System Software can be broadly classified into two types as:

- System control programs
- System support programs
- System development programs

System control programs

- controls the execution of programs
- manage the storage & processing resources of the computer
- perform other management & monitoring function.

- The most important of these programs is the operating system, DBMS & communication monitors.

System support programs

- Provide routine service functions to the other computer programs & computer users: E.g. Utilities, libraries, performance monitors & job accounting.

System development programs

- Assists in the creation of application
- Programs. e.g., language translators such as BASIC interpreter & application generators.

Application software

  - Programs that help users solve particular computing problems.

**Overview of Operating System**

An **operating system** (**OS**) is software that manages computer hardware resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system. Application programs usually require an operating system to function.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and will frequently make a system call to an OS function or be interrupted by it. Operating systems can be found on almost any device that contains a computer—from cellular phones and video game consoles to supercomputers and web servers.

Examples of popular modern operating systems include Android, BSD, iOS, Linux, OS X, QNX, Microsoft Windows, Windows Phone, and IBM z/OS. All these, except Windows, Windows Phone and z/OS, share roots in UNIX.

Types of operating systems

1.1 Real-time

1.2 Multi-user

1.3 Multi-tasking vs. single-tasking

1.4 Distributed

**Objectives and Functions of O.S**

OS typically provides services in thefollowing areas:

1. Program development - The OS provides a variety of facilities and services, such aseditors and debuggers, to assist the programmer in creating programs.

2. Program execution - A number of steps need to be performed to execute a program.Instructions and data must be loaded into main memory, I/O devices and files must beinitialized, and other resources must be prepared. The OS handles these schedulingduties for the user.

3. Access to I/O devices - Each I/O device requires its own peculiar set of instructions orcontrol signals for operation. The OS provides a uniform interface that hides thesedetails so that programmers can access such devices using simple reads and writes.

4. Controlled access to files - For file access, the OS must reflect a detailed understandingof not only the nature of the I/O device (disk drive, tape drive) but also the structure ofthe data contained in the files on the storage medium. In the case of a system withmultiple users, the OS may provide protection mechanisms to control access to the files.

5. System access - For shared or public systems, the OS controls access to the system as awhole and to specific system resources. The access function must provide protection ofresources and data from unauthorized users and must resolve conflicts for resourcecontention.

6. Error detection and response - A variety of errors can occur while a computer system isrunning. These include internal and external hardware errors, such as a memory error,or a device failure or malfunction; and various software errors, such as division by zero,attempt to access forbidden memory location etc. In each case, the OS must provide aresponse that clears the error condition with the least impact on running applications.The response may range from ending the program that caused the error, to retrying theoperation, to simply reporting the error to the application.

7. Accounting - A good OS will collect usage statistics for various resources and monitorperformance parameters such as response time.

The major function of Operating System are it acts as –

- The Resource Manager for Computer.
- The Memory Manager for computer.
- The Device Manager for computer.
- The User manager.

## Types of Operating System

Following are some major types of Operating system.

**Real-time operating system**

A real-time operating system is a multitasking operating system that aims at executing real-time applications. Real-time operating systems often use specialized scheduling algorithms so that they can achieve a deterministic nature of behavior. The main objective of real-time operating systems is their quick and predictable response to events. They have an event-driven or time-sharing design and often aspects of both. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

**Multi-user operating system**

A multi-user operating system allows multiple users to access a computer system at the same time. Time-sharing systems and Internet servers can be classified as multi-user systems as they enable multiple-user access to a computer through the sharing of time. Single-user operating systems have only one user but may allow multiple programs to run at the same time.

**Multi-tasking vs. single-tasking Operating System**

A multi-tasking operating system allows more than one program to be running at the same time, from the point of view of human time scales. A single-tasking system has only one running program.

**DistributedOperating system**

A distributed operating system manages a group of independent computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than

one machine. When computers in a group work in cooperation, they make a distributed system.

**Batch Processing**

Batch processing is the execution of a series of programs ("jobs") on a computer without manual intervention.Jobs are set up so they can be run to completion without human interaction. All input parameters are predefined through scripts, command-line arguments, control files, or job control language. This is in contrast to "online" or interactive programs which prompt the user for such input. A program takes a set of data files as input, processes the data, and produces a set of output data files. This operating environment is termed as "batch processing" because the input data are collected into *batches* or sets of records and each batch is processed as a unit.

**Multiprogramming**

Multiprogramming is a rudimentary form of parallel processing in which several programs are run at the same time on a uniprocessor. Since there is only one processor, there can be no true simultaneous execution of different programs. Instead, the operating system executes part of one program, then part of another, and so on. To the user it appears that all programs are executing at the same time.

**Time Sharing OS**

In computing, **time-sharing** is the sharing of a computing resource among many users by means of multiprogramming and multi-tasking. Its introduction in the 1960s, and emergence as the prominent model of computing in the 1970s, represented a major technological shift in the history of computing.

By allowing a large number of users to interact concurrently with a single computer, time-sharing dramatically lowered the cost of providing computing capability, made it possible for individuals and organizations to use a computer without owning one, and promoted the interactive use of computers and the development of new interactive applications.

**Features of DOS, Windows and UNIX**

The operating system is a system software used for management of computer hardware and application software. It is a interactive interface of computer.

MS-DOS is a very popular operating system for PC, and it is replaced by its extension Windows operating system. In the Windows environment, DOS is also Boss because many utilities programs of MS-DOS are used in trouble shooting of Windows Operating system.

**MS-DOS**

MS-DOS stands for Microsoft Disk Operating System. Tim Paterson developed this **operating system** in 1980. The IBM (International Business Machine) released first PC (Personal Computer) in 1981. MS-DOS version 1.0 was used as operating system in IBM-PC and become talk of town in overnight.

The father of PC Operating System is Gary Kildall of Digital Research. He had his Ph.D in computer and designed more successful operating System called CP/ M. The selling of CP / M is more than 600,000 copies proves its popularity.

The **Microsoft Disk Operating System** or **MS-DOS** was based on QDOS, the Quick and Dirty Operating System written by Tim Peterson of Seattle Computer Products, for their prototype Intel 8086 based computer. QDOS was based on Gray Kildall's CF/M,. Paterson had bought a CP/ M manual and used it as the basis to write his operating system six weeks, QBOS was different enough from CP / M. MS-DOS version 7.0 is lunched in 1997 which is a hidden with Window 95/98 OS.

It has three essential files and many command files. These essential files are: IO.SYS, MSDOS.SYS, and COMMAND.COM. These files are called system files of MS-DOS. The DOS is Boss in real sense, because in the age of Windows operating system, hardware utilities are dependent on the DOS.

The heart of MS-DOS:

- **IO.SYS**: This let DOS communicate with the hardware through the BIOS (Basic Input / Output System).
- **MSDOS.SYS**: This is a DOS kernel.
- **COMMAND.COM**: This is where all the DOS commands are stored and interpreted.
- **COFIG.SYS**: Hardware configuration information is stored here.

- **AUTOEXEC.BAT**: All the programs that are supposed to run at startup are called here.

**Booting**: The booting is a process of loading system files into main memory (RAM). There are two types of booting: **cold booting** and **warm booting**.

## Windows

It is an operating system, extension of MS-DOS with user friendly GUI and several facilities to control memory, hardware, text, graphics, audio, video, internet connection etc.

| Version | Comments |
|---|---|
| Windows 1.0 | This operating system with user interface is a notification of MS-DOS. The nifty mouse is used to click on desired program to open. It was first called interface manager, but then changed in to the more appealing Windows. Windows 1.0 lunched in November 1985. |
| Windows 2.0 | It was released in 1987 to take advantage of the awesome processing power of the Intel 286 processor. The first version of Microsoft Word and Excel are introduced in this version. |
| Windows 3.0 | It was released in May 1990. It came with a prettier 16-color interface, and new technological bells and whistles that let it make better use of the memory. In 1991, Microsoft brought multimedia support for Windows 3.0, called |

| | |
|---|---|
| | Multimedia Extensions 1.0. It gave Windows support for CD-ROM drives and sound cards. It also contained a basic CD player application for Windows. |
| Windows 3.1 | It was released in April 1992. It was equipped with big, comprehensive API (Application Program Interface), which simplified the task of creating user interface and let them focus more time on developing the core functionality of software.<br>In 1993, windows for Workgroup 3.1 were released, which added support for networking, file and printer sharing. It also added Microsoft Mail Program to send and receive over the network. |
| Windows NT | In 1988, Microsoft had been developing, Windows NT; the NT stands for new technology. This was a whole new kernel, built for data and application security.<br>It is a robust, pre-emptive, multi-threaded, multi-tasking, 32-bits operating system with symmetric multiprocessing support. |
| Windows 98 | It released in 1995. It is equipped with advance technology like AGP (Accelerated Graphics Port), MMX (multimedia Extension), USB (Universal serial Bus), DVD (Digital Video Disk) etc. Its most visible |

| | |
|---|---|
| | feature, through, is the active desktop, which integrates the web browser (internet Explorer), with operating system. |
| Windows 2000 | Its interface is similar to interface for windows 98. It has new security protocol with an encryption facility to authenticate users logging in to the network. It supports 32 FAT file system along with NTFC (New Technology File System), making it easier for users to upgrade for Windows 98. It has quite robust hybrid kernel architecture, make it more stable version. |
| Windows Me | Later in 2000, Windows Millennium (windows Me) Edition was released for the home user. This was the last version of Windows to be based on Windows 98. Windows Me had always been regarded as Microsoft's way of keeping users busy while they waited for Windows XP. |
| Windows XP | Here, XP Stands For eXPerience. It bought together the robust Kernel of windows 2000 and all the friendless and multimedia support of windows Me, and painted on a new face for it.<br>Apart from the merger of Windows 2000 and me, Windows Xp also added new features to enhance its performance. The first of These |

| | |
|---|---|
| | was its ability to work even in low-memory conditions without crashing, using a technique called Memory Throttling, Usually, Windows likes to do many things at once, but when memory falls short, it will throttle its memory access, doing fewer at a time. This shows the system down considerably, but prevents it form crashing. |

## UNIX Operating System

The UNIX (pronounced as YEW-nihks) is a powerful, flexible, multi-user Operating system with GUI and several utilities. Ken Thompson and Dennis Ritchie wrote C compiler under UNIX in 1969 at Bell Labs. In 1973, Thompson and Ritchie rewrote the UNIX kernel using C language. It is based on MULTICS operating system. Its first user was Bell patent department. XENIX, VENIX, MICRONIX, LINUX, UNIXWARE-7 etc are version of UNIX operating system.

The UNIX operating system is made up from three parts:

(a) **Kernel**: It is a hub of operating system dedicated for memory management, file management and communication within system.

(b) **Shell**: It is an interface between kernel and users. When a user logs in, the login program matches the username and password, and then starts shell. The shell is a command line interpreter (CLI) of UNIX.

(c)**Program or command** is used to accomplish specific tasks. When one command is terminated, the shell displays prompt % to accept next command for execution.

**Some UNIX Commands:**

- To display files of current directory: Ls

Example: %Ls

- To make directory: Mkdir<destination>

Example**:** %Mkdir unixst

- To change directory: cd

Example: %cd

- To copy file: cp<source><destination>

Example: 5cp /student /example /bio.txt

In UNIX, the dot means the current directory.

- To move file: mv<file1><file2>

Example: 5mv bio.txt hello/

- To remove or delete directory: rm or rmdir<directory>

Example: %rm unixstd

- To display contents of file: Cat<file1>

Example: %cat bio.txt

- To clean monitor: clear

Example: %clear

**Programming Languages**

**Programming language**

A **programming language** is a formal constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms so that the computer hardware can run with a proper step by step instruction.

The earliest programming languages preceded the invention of the computer and were used to direct the behavior of machines such as Jacquard looms and player pianos. Thousands of different programming languages have been created, mainly in the computer field, and many more still are being created every year. Many programming languages require computation to be specified in an imperative form (i.e., as a sequence of operations to perform), while other languages utilize other forms of program specification such as the declarative form (i.e. the desired result is specified, not how to achieve it).

The description of a programming language is usually split into the two components of syntax (form) and semantics (meaning). Some languages are

defined by a specification document (for example, the C programming language is specified by an ISO Standard), while other languages (such as Perl) have a dominant implementation that is treated as a reference.

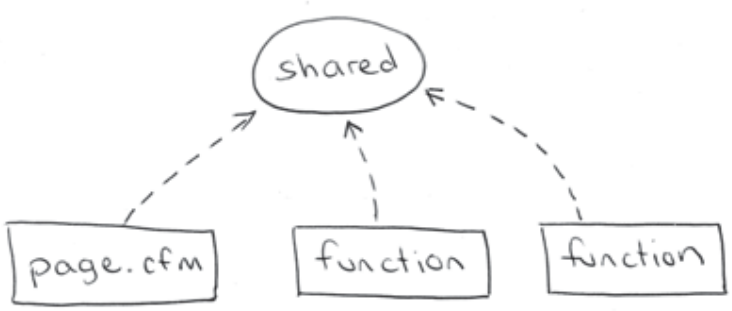
There are two major types of programming languages I,e. Procedural programming language and Object Oriented programming language.

**Procedural programming**

In procedural programing our code is organised into small "procedures" that use and change our data. In ColdFusion, we write our procedures as either custom tags or functions. These functions typically take some input, do something, then produce some output. Ideally your functions would behave as "black boxes" where input data goes in and output data comes out.

The key idea here is that our functions have no intrinsic relationship with the data they operate on. As long as you provide the correct number and type of arguments, the function will do its work and faithfully return its output.

Sometimes our functions need to access data that is not provided as a parameter, i.e., we need access data that is outside the function. Data accessed in this way is considered "global" or "shared" data.

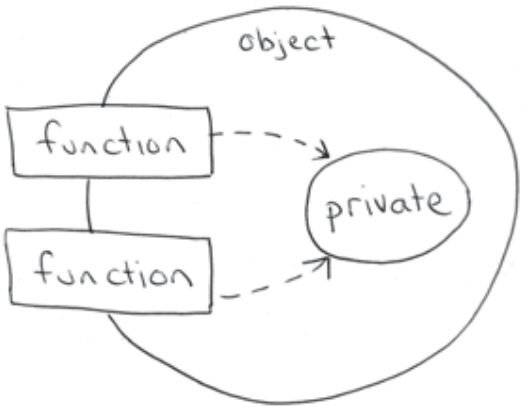



So in a procedural system our functions use data they are "given" (as parameters) but also directly access any shared data they need.

**Object oriented programming**

In object oriented programming, the data and related functions are bundled together into an "object". Ideally, the data inside an object can only be manipulated by calling the object's functions. This means that your data is locked away inside your objects and your functions provide the only means of doing something with that data. In a well designed object oriented system

objects never access shared or global data, they are only permitted to use the data they have, or data they are given.




**Assembler, Compiler, Interpreter, Linker, Loader**

**Assembler:** A computer will not understand any program written in a language, other than its machine language. The programs written in other languages must be translated into the machine language. Such translation is performed with the help of software. A program which translates an assembly language program into a machine language program is called an assembler. If an assembler which runs on a computer and produces the machine codes for the same computer then it is called self-assembler or resident assembler. If an assembler that runs on a computer and produces the machine codes for other computer then it is called Cross Assembler.

Assemblers are further divided into two types: One Pass Assembler and Two Pass Assembler. One pass assembler is the assembler which assigns the memory addresses to the variables and translates the source code into machine code in the first pass simultaneously. A Two Pass Assembler is the assembler which reads the source code twice. In the first pass, it reads all the variables and assigns them memory addresses. In the second pass, it reads the source code and translates the code into object code.

**Compiler:** It is a program which translates a high level language program into a machine language program. A compiler is more intelligent than an assembler. It checks all kinds of limits, ranges, errors etc. But its program run time is more and occupies a larger part of the memory. It has slow speed. Because a

compiler goes through the entire program and then translates the entire program into machine codes. If a compiler runs on a computer and produces the machine codes for the same computer then it is known as a self compiler or resident compiler. On the other hand, if a compiler runs on a computer and produces the machine codes for other computer then it is known as a cross compiler.

**Interpreter:** An interpreter is a program which translates statements of a program into machine code. It translates only one statement of the program at a time. It reads only one statement of program, translates it and executes it. Then it reads the next statement of the program again translates it and executes it. In this way it proceeds further till all the statements are translated and executed. On the other hand, a compiler goes through the entire program and then translates the entire program into machine codes. A compiler is 5 to 25 times faster than an interpreter.

By the compiler, the machine codes are saved permanently for future reference. On the other hand, the machine codes produced by interpreter are not saved. An interpreter is a small program as compared to compiler. It occupies less memory space, so it can be used in a smaller system which has limited memory space.

**Linker:** In high level languages, some built in header files or libraries are stored. These libraries are predefined and these contain basic functions which are essential for executing the program. These functions are linked to the libraries by a program called Linker. If linker does not find a library of a function then it informs to compiler and then compiler generates an error. The compiler automatically invokes the linker as the last step in compiling a program.

Not built in libraries, it also links the user defined functions to the user defined libraries. Usually a longer program is divided into smaller subprograms called modules. And these modules must be combined to execute the program. The process of combining the modules is done by the linker.

**Loader:** Loader is a program that loads machine codes of a program into the system memory. In Computing, a **loader** is the part of an Operating System that is responsible for loading programs. It is one of the essential stages in the process of starting a program. Because it places programs into memory and prepares them for execution. Loading a program involves reading the contents of executable file into memory. Once loading is complete, the operating system

starts the program by passing control to the loaded program code. All operating systems that support program loading have loaders. In many operating systems the loader is permanently resident in memory.

**Computer Virus**

.

A **computer virus** is a malware program that, when executed, replicates by inserting copies of itself (possibly modified) into other computer programs, data files, or the boot sector of the hard drive. When this replication succeeds, the affected areas are then said to be "infected". Viruses often perform some type of harmful activity on infected hosts, such as stealing hard disk space or CPU time, accessing private information, corrupting data, displaying political or humorous messages on the user's screen, spamming their contacts, or logging their keystrokes. However, not all viruses carry a destructive payload or attempt to hide themselves—the defining characteristic of viruses is that they are self-replicating computer programs which install themselves without the user's consent.

Virus writers use social engineering and exploit detailed knowledge of security vulnerabilities to gain access to their hosts' computing resources. The vast majority of viruses target systems running Microsoft Windows, employing a variety of mechanisms to infect new hosts, and often using complex anti-detection/stealth strategies to evade antivirus software. Motives for creating viruses can include seeking profit, desire to send a political message, personal amusement, to demonstrate that a vulnerability exists in software, for sabotage and denial of service, or simply because they wish to explore artificial life and evolutionary algorithms.

Computer viruses currently cause billions of dollars worth of economic damage each year, due to causing systems failure, wasting computer resources, corrupting data, increasing maintenance costs, etc. In response, free, open-source antivirus tools have been developed, and a multi-billion dollar industry of antivirus software vendors has cropped up, selling virus protection to users of various operating systems of which Android and Windows are among the most victimized. Unfortunately, no currently existing antivirus software is able to catch all computer viruses (especially new ones); computer security researchers are

actively searching for new ways to enable antivirus solutions to more effectively detect emerging viruses, before they have already become widely distributed.

**Different Types of computer virus**

There are different types of viruses which can be classified according to their origin, techniques, types of files they infect, where they hide, the kind of damage they cause, the type of operating system, or platform they attack. Let us have a lookatfewofthem.

**Memory Resident**VirusThese viruses fix themselves in the computer memory and get activated whenever the OS runs and infects all the files that are then opened.Hideout: This type of virus hides in the RAM and stays there even after the malicious code is executed. It gets control over the system memory and allocate memory blocks through which it runs its own code, and executes the code when any function is executed.Target: It can corrupt files and programs that are opened, closed, copied, renamed, etc.Examples: Randex, CMJ, Meve, and MrKlunkyProtection: Install an antivirus program.

**Direct Action Viruses**The main purpose of this virus is to replicate and take action when it is executed. When a specific condition is met, the virus will go into action and infect files in the directory or folder that are specified in the AUTOEXEC.BAT file path. This batch file is always located in the root directory of the hard disk and carries out certain operationswhenthecomputerisbooted.FindFirst/FindNext technique is used where the code selects a few files as its victims. It also infects the external devices like pen drives or hard disks by copying itself on them.Hideout: The viruses keep changing their location into new files whenever the code is executed, but are generally found in the hard disk's root directory. Target: It can corrupt files. Basically, it is a file-infecter virus.

Examples:Viennavirus

Protection: Install an antivirus scanner. However, this type of virus has minimal effect on the computer's performance.

**OverwriteViruses**A virus of this kind is characterized by the fact that it deletes the information contained in the files that it infects, rendering them partially or totally useless once they have been infected.Hideout: The virus replaces the file content. However, it doesnotchangethefilesize.Examples:Way,Trj.Reboot,Trivial.88.DProtection: The only way to clean a file infected by an overwrite virus is to delete the file completely, thus losing the original content.However, it is very easy to detect this type of virus, as the original program becomes useless.

**Boot Sector Virus**This type of virus affects the boot sector of a hard disk. This is a crucial part of the disk, in which information of the disk itself is stored along with a program that makes it possible to boot (start) the computer from the disk. This type of virus is also called Master Boot Sector Virus or Master Boot Record Virus.Hideout: It hides in the memory until DOS accesses the floppy disk, and whichever boot data is accessed, the virus infects it.

Examples:Polyboot.B,AntiEXEProtection: The best way of avoiding boot sector viruses is to ensure that floppy disks are write-protected. Also, never start your computer with an unknown floppy disk in the disk drive.

**Macro viruses** infect files that are created using certain applications or programs that contain macros, like .doc, .xls, .pps, .mdb, etc. These mini-programs make it possible to automate series of operations so that they are performed as a single action, thereby saving the user from having to carry them out one by one. These viruses automatically infect the file that contains macros, and also infects the templates and documents that the file contains. It is referred to as a type of e-mail virus.Hideout: These hide in documents that are shared via e-mail or networks.
Examples: Relax, Melissa.A, Bablas, O97M/Y2K
Protection: The best protection technique is to avoid opening e-mails from unknown senders. Also, disabling macros can help to protect your useful data.

**Directory viruses** (also called Cluster Virus/File System Virus) infect the directory of your computer by changing the path that indicates the location of a

file. When you execute a program file with an extension .EXE or .COM that has been infected by a virus, you are unknowingly running the virus program, while the original file and program is previously moved by the virus. Once infected, it becomes impossible to locate the original files.Hideout: It is usually located in only one location of the disk, but infects the entire program in the directory. Examples:Dir-2virusProtection: All you can do is, reinstall all the files from the backup that are infected after formatting the disk.

**Polymorphic viruses** encrypt or encode themselves in a different way (using different algorithms and encryption keys) every time they infect a system. This makes it impossible for antivirus software to find them using string or signature searches (because they are different in each encryption). The virus then goes on to create a large number of copies.Examples: Elkern, Marburg, Satan Bug and Tuareg
Protection: Install a high-end antivirus as the normal ones are incapable of detecting this type of virus.

**Companion viruses** can be considered as a type of file infector virus, like resident or direct action types. They are known as companion viruses because once they get into the system they 'accompany' the other files that already exist. In other words, to carry out their infection routines, companion viruses can wait in memory until a program is run (resident virus), or act immediately by making copies of themselves (direct action virus).Hideout: These generally use the same filename and create a different extension of it. For example: If there is a file "Me.exe", the virus creates another file named "Me.com" and hides in the new file. When the system calls the filename "Me", the ".com" file gets executed (as ".com" has higher priority than ".exe"), thus infecting the system.
Examples: Stator, Asimov.1539 and Terrax.1069
Protection: Install an antivirus scanner and also download Firewall.

**FAT Virus**the file allocation table (FAT) is the part of a disk used to store all the information about the location of files, available space, unusable space, etc.Hideout: FAT virus attacks the FAT section and may damage crucial information. It can be especially dangerous as it prevents access to certain

sections of the disk where important files are stored. Damage caused can result in loss of information from individual files or even entire directories.

Examples: Link VirusProtection: Before the virus attacks all the files on the computer, locate all the files that are actually needed on the hard drive, and then delete the ones that are not needed. They may be files created by viruses.

**MultipartiteVirus**These viruses spread in multiple ways possible. It may vary in its action depending upon the operating system installed and the presence of certain files.Hideout: In the initial phase, these viruses tend to hide in the memory as the resident viruses do; then they infect the hard disk.
Examples: Invader, Flip and Tequila
Protection: You need to clean the boot sector and also the disk to get rid of the virus, and then reload all the data in it. However, ensure that the data is clean.

**Web Scripting Virus**Many web pages include complex codes in order to create an interesting and interactive content. This code is often exploited to bring about certain undesirable actions.Hideout: The main sources of web scripting viruses are the web browsers or infected web pages.
Examples: JS.Fortnight is a virus that spreads through malicious e-mails.
Protection: Install the microsoft tool application that is a default feature in Windows 2000, Windows 7 and Vista. Scan the computer with this application.
Worms

A worm is a program very similar to a virus; it has the ability to self-replicate and can lead to negative effects on your system. But they can be detected and eliminated by an antivirus software.Hideout: These generally spread through e-mails and networks. They do not infect files or damage them, but they replicate so fast that the entire network may collapse.
Examples: PSWBugbear.B, Lovgate.F, Trile.C, Sobig.D, Mapson
Protection: Install an updated version of antivirus.

**Trojans** are another unsavory breed of malicious code are Trojans or Trojan horses, which unlike viruses, do not reproduce by infecting other files, nor do

they self-replicate like worms. In fact, it is a program which disguises itself as a useful programorapplication.

- Beware of the fact that these viruses copy files in your computer (when their carrier program is executed) that can damage your data, and even delete it. The attacker can also program the trojans in such a manner that the information in our computer is accessible to them.

**Logic Bombs** are not considered viruses because they do not replicate. They are not even programs in their own right, but rather camouflaged segments of other programs. They are only executed when a certain predefined condition is met. Their objective is to destroy data on the computer once certain conditions have been met. Logic bombs go undetected until launched, the results can be destructive, and your entire data can be deleted.

**Detection and prevention of Virus**

Following steps may be taken for Virus Detection and Prevention.

1. Do not open any files attached to an email from an unknown, suspicious or untrustworthy source.
2. Do not open any files attached to an email unless you know what it is, even if it appears to come from adear friend or someone you know. Some viruses can replicate themselves and spread through email.Better be safe than sorry and confirm that they really sent it.
3. Delete chain emails and junk email. Do not forward or reply to any to them. These types of email areconsidered spam, which is unsolicited, intrusive mail that clogs up the network.
4. Exercise caution when downloading files from the Internet. Ensure that the source is a legitimate and areputable one. Verify that an anti-virus program checks the files on the download site. If you're uncertain,don't download the file at all.
5. Update your anti-virus software regularly. Thousands of viruses are discovered each month, so you'llwant to be protected.
6. Back up your files on a regular basis. If a virus destroys your files, at least you can replace them withyour back-up copy. You should store your backup copy in

a separate location from your work files, onethat is preferably not on your computer.

7. When in doubt, always err on the side of caution and do not open, download, or execute any files oremail attachments. Not executing the files is especially important. Check with your product vendors forupdates which include those for your operating system web browser, and email. One example is thesecurity site section of Microsoft located at http://www.microsoft.com/security.

8. Stay away from Bit torrent sites. Some of the more popular ones include Limewire, BitTorrent,Frostwire and Pirate Bay. These are heavily laden with viruses, malware and spyware. Downloadingmaterial from these websites is one of the easiest ways to become infected. It's in your best interest tojust avoid these websites completely.

9. Be careful when searching on the internet, the links that come up from your search engine may contain avirus. Never go to sites that sound suspicious.

10. Due to the popularity of the social networking websites such as MySpace, Facebook, and Twitter, virusmakers target them more than any other website. Online gaming and gambling websites also are high riskwebsites. It's best to avoid these kinds of websites altogether.

11. If you happen to see a popup message when on the internet about being infected and to buy theirsoftware to protect yourself, do not fall for it! Most of the time these messages are easy to see as theytend to have bad grammar and spelling errors. Common names are XP Antivirus, Security Tools,ThinkPoint, Security Shield, Win 7 Security 2011, and similar variations. If do see one of these popups, donot click on them, immediately shut down your computer. If you click on any part of thosewindows you will give the virus permission to install and bypass your antivirus program.

12. If you see any suspicious pop-ups appear on your screen, do not click on them. If you do, it is verylikely you will infect your computer. Instead use the following keyboard command, which will allow you toclose the pop-up, without having the click on it or infecting yourself. The keyboard command is ALT + F4.
If that fails, then shut down the computer.

**Application of computers in different Domain**

**Uses of Computer at Home**

Computer can be used at home in the following ways.

**Home Budget**

Computer can be used to manage Home Budget. You can easily calculate your expenses and income. You can list all expenses in one column and income in another column. Then you can apply any calculation on these columns to plan your home budget. There are also specialize software that can manage your income and expenses and generate some cool reports.

**Computer Games**

An important use of computers at home is playing games. Different types of games are available. These games are a source of entertainment and recreation. Many games are available that are specially developed to improve your mental capability and thinking power.

**Working from Home**

People can manage the office work at home. The owner of a company can check the work of the employees from home. He can control his office while sitting at home.

**Entertainment**

People can find entertainment on the internet. They can watch movies, listen to songs, and watch videos download different stuff. They can also watch live matches on the internet.

**Information**

People can find any type of information on the internet. Educational and informative websites are available to download books, tutorials etc. to improve their knowledge and learn new things.

**Chatting & Social Media**

People can chat with friends and family on the internet using different software like Skype etc. One can interact with friends over social media websites like Facebook, Twitter & Google Plus. They can also share photos and videos with friends.

**Uses of Computers in Education**

CBT are different programs that are supplied on CD-ROM. These programs include text, graphics and sound. Audio and Video lectures are recorded on the CDs. CBT is a low cost solution for educating people. You can train a large number of people easily.

**Benefits of CBT**

Some benefits of CBT are as follows:

1. The students can learn new skills at their own pace. They can easily acquire knowledge in any available time of their own choice.
2. Training time can be reduced.
3. Training materials are interactive and easy to learn. It encourages students to learn the topic.
4. Planning and timing problems are reduced or eliminated.
5. The skills can be taught at any time and at any place.
6. It is very cost effective way to train a large number of students.
7. Training videos and audios are available at affordable prices.

**Computer Aided Learning (CAL)**

Computer aided learning is the process of using information technology to help teaching and enhance the learning process. The use of computer can reduce the time that is spent on preparing teaching material. It can also reduce the administrative load of teaching and research. The use of multimedia projector and PowerPoint presentations has improved the quality of teaching. It has also helped the learning process.

**Distance Learning**

Distance learning is a new learning methodology. Computer plays the key role in this kind of learning. Many institutes are providing distance learning programs. The student does not need to come to the institute. The institute provides the reading material and the student attends virtual classroom. In virtual classroom, the teacher delivers lecture at his own workplace. The student can attend the lecture at home by connecting to a network. The student can also ask questions to the teacher.

**Online Examination**

The trend of online examination is becoming popular. Different examination like GRE, GMAT and SAT are conducted online all over the world. The questions are marked by computer. It minimizes the chance of mistakes. It also enables to announce the result in time.

**Uses of Computers in Business**

The use of computer technology in business provides many facilities. Businessmen are using computers to interact with their customers anywhere in the world. Many business tasks are performed more quickly and efficiently. Computers also help them to reduce the overall cost of their business. Computer can be used in business in the following ways.

**Marketing**

An organization can use computers for marketing their products. Marketing applications provide information about the products to customers. Computer is also used to manage distribution system, advertising and selling activities. It can also be used in deciding pricing strategies. Companies can know more about their customers and their needs and requirements etc.

**Stock Exchange**

Stock Exchange is the most important place for businessmen. Many stock exchanges use computers to conduct bids. The stockbrokers perform all trading activities electronically. They connect with the computer where brokers match the buyers with sellers. It reduces cost as no paper or special building is required to conduct these activities.

**Uses of computers in Medical Field**

**Hospital Management System**

Specialized hospital management softwares are used to automate the day to day procedures and operations at hospitals. These tasks may be Online appointments, payroll admittance and discharge records etc.

**Patient History**

Hospital management systems can store data about patients. Computers are used to store data about patients, their diseases & symptoms, the medicines that are prescribed.

**Patients Monitoring**

Monitoring systems are installed in medical wards and Intensive care units to monitoring patients continously. These systems can monitor pulse, blood pressure and body temperature and can alert medical staff about any serious situations.

**Life Support Systems**

Specialized devices are used to help impaired patients like hearing aids.

**Diagnosis Purpose**

A variety of software are used to investigate symptoms and prescribed medication accordingly. Sophisticated systems are used for tests like CT Scan, ECG, and other medical tests.

# CHAPTER - 3

## Networking concept, Protocol, Connecting Media

### Introduction

In the world of computers, networking is the practice of linking two or more computing devices together for the purpose of sharing data and other hardware & software resources. Networks are built with a combination of computer hardware and computer software. Some explanations of networking found in books and tutorials are highly technical, designed for students and professionals, while others are geared more to home and business uses of computer networks.

### Types of Data Transmissions

Following are two basic ways in which we can establish a network through a transmission mechanism

**Serial transmission**

In communications, serial transmission is the sequential transmission of signal elements of a group representing a character or other entity of data.

The characters are transmitted in a sequence over a single line, rather than simultaneously over two or more lines, as in parallel transmission. The sequential elements may be transmitted with or without interruption. The internal channels within the computer are typically parallel. The channel between the CPU and

memory is parallel, and the channels between the CPU and peripheral devices are typically parallel; however, parallel is giving way to serial. For example, ATA disk drives (IDE drives) used parallel pathways for years, but Serial ATA superseded Parallel ATA. That might seem like a step backward, but serial circuits are easier to design than parallel, and today's chips are fast enough to make serial a more viable channel/bus technology

**Synchronous transmission**

The transmission of data in which both stations are synchronized. Codes are sent from the transmitting station to the receiving station to establish the synchronization, and data is then transmitted in continuous streams. Modems that transmit at 1,200 bps and higher often convert the asynchronous signals from a computer's serial port into synchronous transmission over the transmission line. Contrast with asynchronous transmission.

**Asynchronous transmission**

The transmission of data in which each character is a self-contained unit with its own start and stop bits. Intervals between characters may be uneven. It is the common method of transmission between a computer and a modem, although the modem may switch to synchronous transmission to communicate with the other modem. Also called "start/stop transmission." Contrast with synchronous transmission.

Examples of serial communication architectures

- Morse code telegraphy
- RS-232 (low-speed, implemented by Serial Ports)
- RS-423
- RS-485
- Universal Serial Bus (moderate-speed, for connecting computers to peripherals)
- FireWire
- Ethernet
- Fibre Channel (high-speed, for connecting computers to mass storage devices)
- InfiniBand (very high speed, broadly comparable in scope to PCI)

- MIDI control of electronic musical instruments
- DMX512 control of theatrical lighting
- Serial Attached SCSI
- Serial ATA
- PCI Express
- SONET and SDH (high speed telecommunication over optical fibers)
- T-1, E-1 and variants (high speed telecommunication over copper pairs)

**Parallel communication**

In telecommunication and computer science, parallel communication is a method of sending several data signals simultaneously over a communication link *(comprising of several wired channels in parallel)* at one time. It contrasts with serial communication; this distinction is one way of characterizing a communications link.

The basic difference between a parallel and a serial communication channel is the number of distinct wires or strands at the physical layer used for simultaneous transmission from a device. Parallel communication implies more than one such wire/strand, in addition to a ground connection.

Examples of parallel communication systems

- Computer peripheral buses: ISA, ATA, SCSI, PCI and Front side bus, and the once-ubiquitous IEEE-1284 / Centronics "printer port"
- Laboratory Instrumentation bus IEEE-488

Comparison with serial links

Before the development of high-speed serial technologies, the choice of parallel links over serial links was driven by these factors:

- Speed: Superficially, the speed of a parallel data link is equal to the number of bits sent at one time times the bit rate of each individual path; doubling the number of bits sent at once doubles the data rate. In practice, skew reduces the speed of every link to the slowest of all of the links.
- Cable length: Crosstalk creates interference between the parallel lines, and the effect worsens with the length of the communication link. This places an upper limit on the length of a parallel data connection that is usually shorter than a serial connection.

- Complexity: Parallel data links are easily implemented in hardware, making them a logical choice. Creating a parallel port in a computer system is relatively simple, requiring only a latch to copy data onto a data bus. In contrast, most serial communication must first be converted back into parallel form by a Universal asynchronous receiver transmitter before they may be directly connected to a data bus.

The decreasing cost of integrated circuits, combined with greater consumer demand for speed and cable length, has led to parallel communication links becoming deprecated in favor of serial links; for example, IEEE 1284 printer ports vs. USB, Advanced Technology Attachment vs. Serial ATA, SCSI vs. FireWire.

On the other hand, there has been a resurgence of parallel data links in RF communication. Rather than transmitting one bit at a time (as in Morse code and BPSK), well-known techniques such as PSM, PAM, and Multiple-input multiple-output communication send a few bits in parallel. (Each such group of bits is called a "symbol (data)"). Such techniques can be extended to send an entire byte at once (256-QAM). More recently techniques such as OFDM have been used in Asymmetric Digital Subscriber Line to transmit over 224 bits in parallel, and in DVB-T to transmit over 6048 bits in parallel.

**Network protocol**

A network protocol defines rules and conventions for communication between network devices. Protocols for computer networking all generally use packet switching techniques to send and receive messages in the form of *packets*. Network protocols include mechanisms for devices to identify and make connections with each other, as well as formatting rules that specify how data is packaged into messages sent and received. Some protocols also support message acknowledgement and data compression designed for reliable and/or high-performance network communication. Hundreds of different computer network protocols have been developed each designed for specific purposes and environments.

**Internet Protocols**

The Internet Protocol family contains a set of related (and among the most widely used network protocols. Besides Internet Protocol (IP) itself, higher-level protocols like TCP, UDP, HTTP, and FTP all integrate with IP to provide additional capabilities. Similarly, lower-level Internet Protocols like ARP and ICMP also co-exist with IP.

These higher level protocols interact more closely with applications like Web browsers while lower-level protocols interact with network adapters and other computer hardware.

Routing Protocols

Routing protocols are special-purpose protocols designed specifically for use by network routers on the Internet. Common routing protocols include EIGRP, OSPF and BGP.

**Implementation of Network Protocols**

Modern operating systems like Microsoft Windows contain built-in services or daemons that implement support for some network protocols. Applications like Web browsers contain software libraries that support the high level protocols necessary for that application to function. For some lower level TCP/IP and routing protocols, support is implemented in directly hardware (silicon chipsets) for improved performance

Client-server

The client-serversoftware architecture model distinguishes client systems from server systems, which communicate over a computer network. A client-server application is a distributed system comprised of both client and server software. A client software process may initiate a communication session, while the server waits for requests from any client.

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfills the request. Although the client/server idea can be used by programs within a single computer, it is a more important idea in a network. In a network, the client/server model provides a convenient way to efficiently interconnect programs that are distributed across different locations. Computer transactions using the client/server model are very common. Most Internet applications, such as email, web access and database access, are based on the client/server model. For example, a web browser is a client program at the user computer that may access information at any web server in the world. To check your bank account from your computer, a web browser client program in your computer forwards your request to a web server program at the bank. That program may in turn forward the request to its own database client program that sends a request to a database server at another bank computer to retrieve your account balance. The balance is returned back to the bank

database client, which in turn serves it back to the web browser client in your personal computer, which displays the information for you.

The client/server model has become one of the central ideas of network computing. Most business applications being written today use the client/server model. So do the Internet's main application protocols, such as HTTP, SMTP, Telnet, DNS, etc. In marketing, the term has been used to distinguish distributed computing by smaller dispersed computers from the "monolithic" centralized computing of mainframe computers. But this distinction has largely disappeared as mainframes and their applications have also turned to the client/server model and become part of network computing.

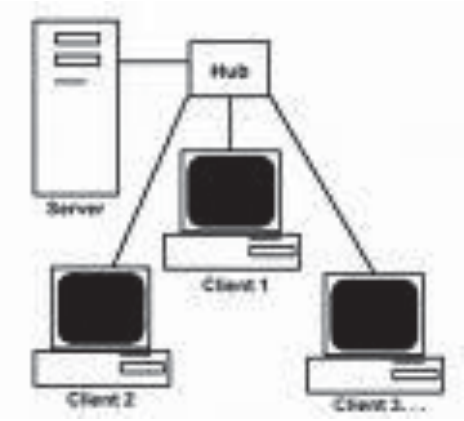
Each instance of the client software can send data requests to one or more connected *server*s. In turn, the servers can accept these requests, process them, and return the requested information to the client. Although this concept can be applied for a variety of reasons to many different kinds of applications, the architecture remains fundamentally the same.

The most basic type of client-server architecture employs only two types of hosts: clients and servers. This type of architecture is sometimes referred to as *two-tier*. It allows devices to share files and resources.

These days, clients are most often web browsers, although that has not always been the case. Servers typically include web servers, database servers and mail servers. Online gaming is usually client-server too. In the specific case of MMORPG, the servers are typically operated by the company selling the game; for other games one of the players will act as the host by setting his game in server mode.

The interaction between client and server is often described using sequence diagrams. Sequence diagrams are standardized in the Unified Modeling Language.

When both the client- and server-software are running on the same computer, this is called a *single seat* setup.




[Client Server architecture]

**A peer-to-peer based network.**

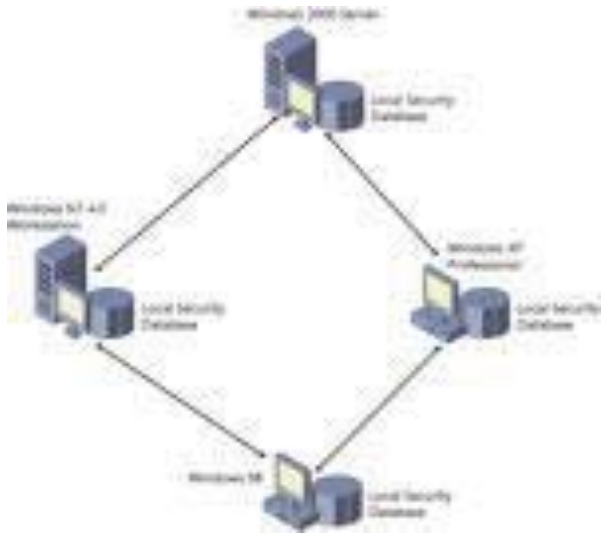
http://en.wikipedia.org/wiki/Image:Server-based-network.svgA peer to peer (or P2P) computer network uses diverse connectivity between participants in a network and the cumulative bandwidth of network participants rather than conventional centralized resources where a relatively low number of servers provide the core value to a service or application. P2P networks are typically used for connecting nodes via largely *ad hoc* connections. Such networks are useful for many purposes. Sharing content files containing audio, video, data or anything in digital format is very common, and realtime data, such as telephony traffic, is also passed using P2P technology.

A pure P2P network does not have the notion of clients or servers but only equal *peer*nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the client-server model where communication is usually to and from a central server. A typical example of a file transfer that is not P2P is an FTP server where the client and server programs are quite distinct, the clients initiate the download/uploads, and the servers react to and satisfy these requests.

The earliest P2P network in widespread use was the Usenet news server system, in which peers communicated with one another to propagate Usenet news articles over the entire Usenet network. Particularly in the earlier days of Usenet, UUCP was used to extend even beyond the Internet. However, the news server system also acted in a client-server form when individual users accessed a local news server to read and post articles. The same consideration applies to SMTP email in the sense that the

core email relaying network of Mail transfer agents is a P2P network while the periphery of Mail user agents and their direct connections is client server. P2P architecture embodies one of the key technical concepts of the Internet, described in the first Internet Request for Comments, RFC 1, "Host Software" dated 7 April1969. More recently, the concept has achieved recognition in the general public in the context of the absence of central indexing servers in architectures used for exchanging multimedia files.

The concept of P2P is increasingly evolving to an expanded usage as the relational dynamic active in distributed networks, i.e. not just computer to computer, but human to human.



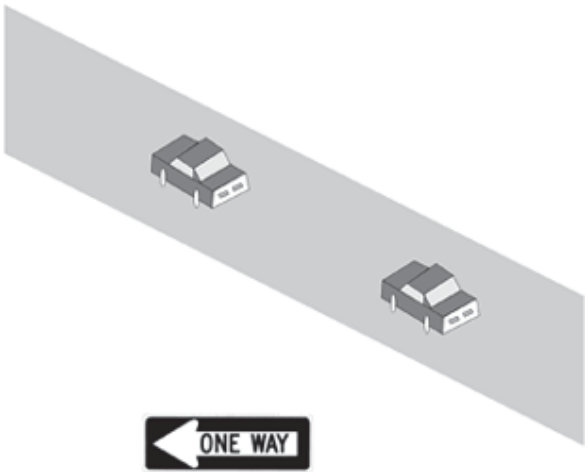
[Peer-Peer architecher]

**Date Transmission mode**

Network devices use three transmission modes (methods) to exchange data, or "talk" to each other, as follows: simplex, half duplex, and full duplex.

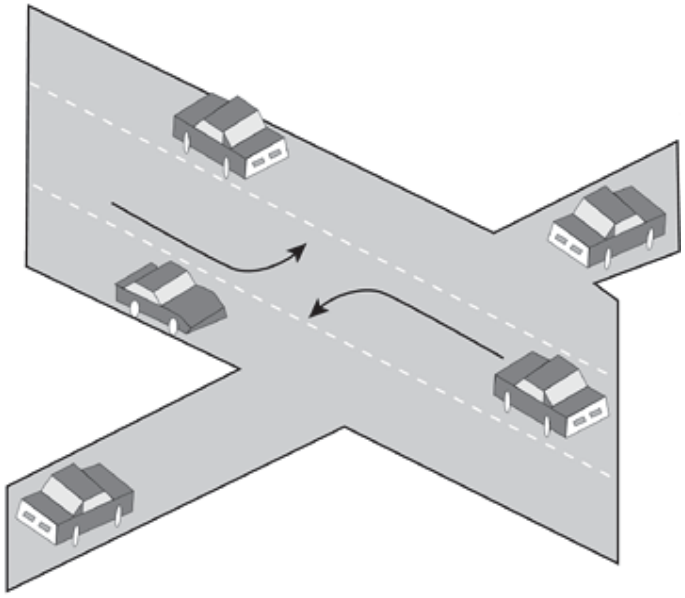
- Simplex
- Half Duplex
- Full Duplex

Simplex transmission is like a one-way street where traffic moves in only one direction. Simplex mode is a one-way-only transmission, which means that data can flow only in one direction from the sending device to the receiving device. Figure 1-7 illustrates simplex transmission.



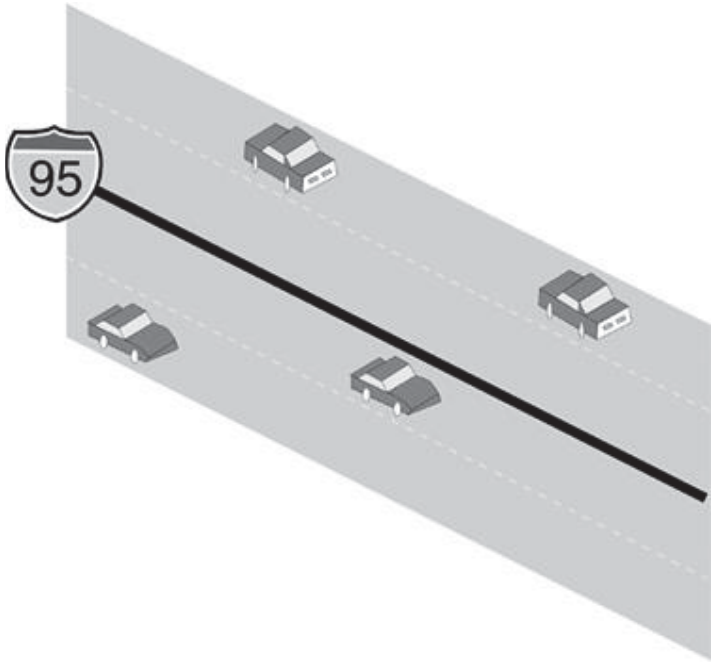

[Simplex (One-Way Street)]

Half-duplex transmission is like the center lane on some three-lane roads. It is a single lane in which traffic can move in one direction or the other, but not in both directions at the same time. Half-duplex mode limits data transmission because each device must take turns using the line. Therefore, data can flow from A to B and from B to A, but not at the same time. Figure 1-8 illustrates half-duplex transmission.



[Half Duplex (Center Turn Lane)]

Full-duplex transmission is like a major highway with two lanes of traffic, each lane accommodating traffic going in opposite directions. Full-duplex mode accommodates two-way simultaneous transmission, which means that both sides can send and receive at the same time. In full-duplex mode, data can flow from A to B and B to A at the same time. Figure 1-9 illustrates full-duplex transmission.




[Full Duplex (Interstate Highway)]

Full-duplex transmission is, in fact, two simplex connections: One connection has traffic flowing in only one direction; the other connection has traffic flowing in the opposite direction of the first connection

In computer networking, *topology* refers to the layout of connected devices. This article introduces the standard topologies of networking.

**Network Topologies**

A topology is a network's virtual shape or structure. This shape does not necessarily correspond to the actual physical layout of the devices on the network. For example, the computers on a home LAN may be arranged in a circle in a family room, but it would be highly unlikely to find a ring topology there.

Network topologies are categorized into the following basic types:

- bus
- ring

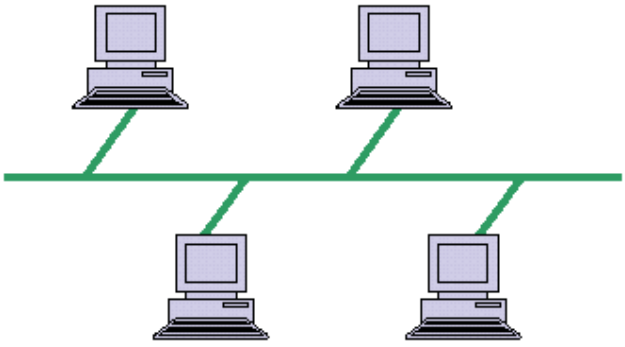
- star
- tree
- mesh

More complex networks can be built as hybrids of two or more of the above basic topologies.

**Bus Topology**

Bus networks (not to be confused with the system bus of a computer) use a common backbone to connect all devices. A single cable, the backbone functions as a shared communication medium that devices attach or tap into with an interface connector. A device wanting to communicate with another device on the network sends a broadcast message onto the wire that all other devices see, but only the intended recipient actually accepts and processes the message.

Ethernet bus topologies are relatively easy to install and don't require much cabling compared to the alternatives. 10Base-2 ("ThinNet") and 10Base-5 ("ThickNet") both were popular Ethernet cabling options many years ago for bus topologies. However, bus networks work best with a limited number of devices. If more than a few dozen computers are added to a network bus, performance problems will likely result. In addition, if the backbone cable fails, the entire network effectively becomes unusable.

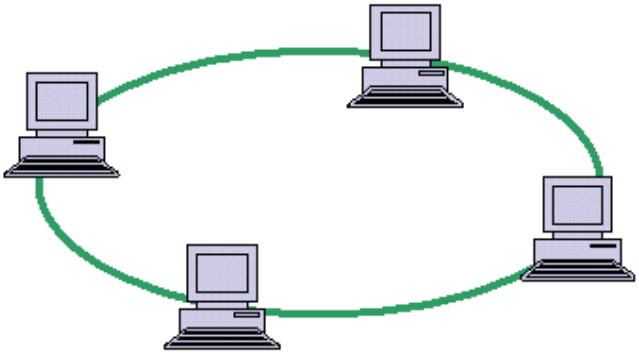


[ Bus Topology Diagram]

**Ring Topology**

In a ring network, every device has exactly two neighbors for communication purposes. All messages travel through a ring in the same direction (either "clockwise" or "counterclockwise"). A failure in any cable or device breaks the loop and can take down the entire network.

To implement a ring network, one typically uses FDDI, SONET, or Token Ring technology. Ring topologies are found in some office buildings or school campuses.

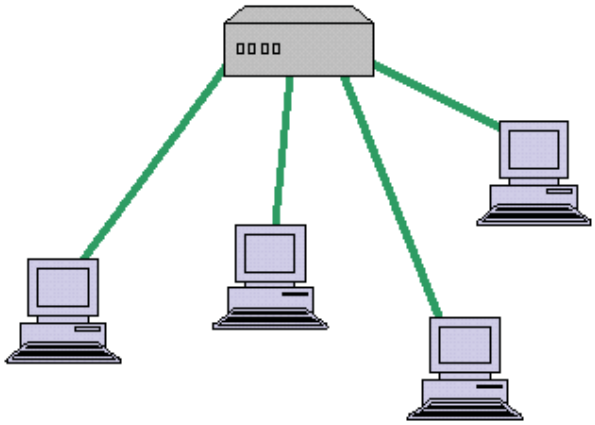


[Ring Topology Diagram ]

**Star Topology**

Many home networks use the star topology. A star network features a central connection point called a "hub" that may be a hub, switch or router. Devices typically connect to the hub with Unshielded Twisted Pair (UTP) Ethernet.

Compared to the bus topology, a star network generally requires more cable, but a failure in any star network cable will only take down one computer's network access and not the entire LAN. (If the hub fails, however, the entire network also fails.)

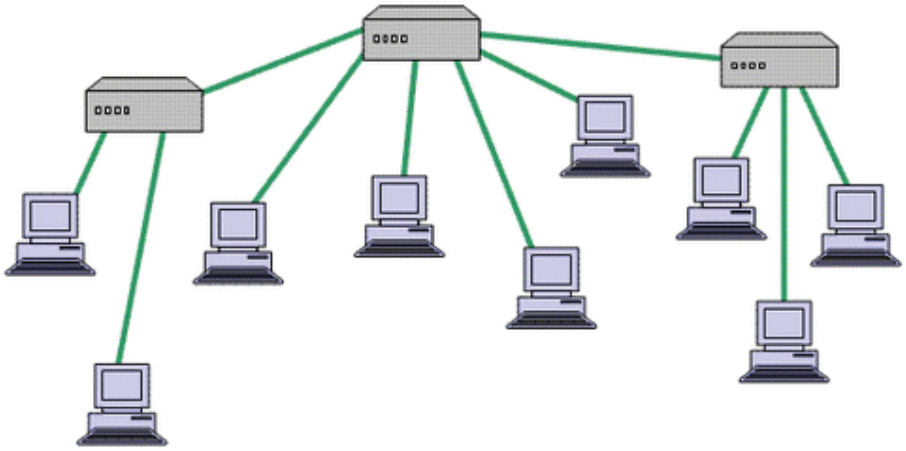


[ Star Topology Diagram]

**Tree Topology**

Tree topologies integrate multiple star topologies together onto a bus. In its simplest form, only hub devices connect directly to the tree bus, and each hub functions as the "root" of a tree of devices. This bus/star hybrid approach supports future

expandability of the network much better than a bus (limited in the number of devices due to the broadcast traffic it generates) or a star (limited by the number of hub connection points) alone.

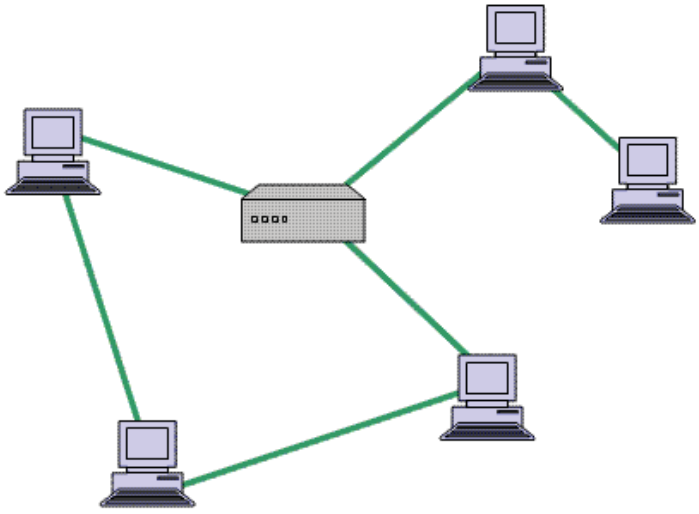


[Tree Topology Diagram]

**Mesh Topology**

Mesh topologies involve the concept of routes. Unlike each of the previous topologies, messages sent on a mesh network can take any of several possible paths from source to destination. (Recall that even in a ring, although two cable paths exist, messages can only travel in one direction.) Some WANs, most notably the Internet, employ mesh routing.

A mesh network in which every device connects to every other is called a full mesh. As shown in the illustration below, partial mesh networks also exist in which some devices connect only indirectly to others.



[ Mesh Topology Diagram]

Topologies remain an important part of network design theory. You can probably build a home or small business computer network without understanding the difference between a bus design and a star design, but becoming familiar with the standard topologies gives you a better understanding of important networking concepts like hubs, broadcasts, and routes.

**Types of Network**

Below is a list of the most common types of computer networks in order of scale from less number of computers and geographical area coverage.

- Personal area network
- Local Area Network
- Campus Area Network
- Metropolitan Area Network
- Wide Area Network
- Global Area Network
- Internetwork
- Intranet
- Extranet
- Internet

**Personal Area Network (PAN)**

A personal area network (PAN) is a computer network used for communication among computer devices close to one person. Some examples of devices that are used in a PAN are printers, fax machines, telephones, PDAs or scanners. The reach of a PAN is typically within about 20-30 feet (approximately 6-9 metres).

Personal area networks may be wired with computer buses such as USB and FireWire. A wireless personal area network (WPAN) can also be made possible with network technologies such as IrDA and Bluetooth..

**Local Area Network (LAN)**

A network covering a small geographic area, like a home, office, or building. Current LANs are most likely to be based on Ethernet technology. For example, a library may have a wired or wireless LAN for users to interconnect local devices (e.g., printers

and servers) and to connect to the internet. On a wired LAN, PCs in the library are typically connected by category 5 (Cat5) cable, running the IEEE 802.3 protocol through a system of interconnection devices and eventually connect to the internet. The cables to the servers are typically on Cat 5e enhanced cable, which will support IEEE 802.3 at 1 Gbit/s. A wireless LAN may exist using a different IEEE protocol, 802.11b or 802.11g. The staff computers (bright green in the figure) can get to the color printer, checkout records, and the academic network *and* the Internet. All user computers can get to the Internet and the card catalog. Each workgroup can get to its local printer. Note that the printers are not accessible from outside their workgroup.

The defining characteristics of LANs, in contrast to WANs (wide area networks), include their higher data transfer rates, smaller geographic range, and lack of a need for leased telecommunication lines. Current Ethernet or other IEEE 802.3 LAN technologies operate at speeds up to 10 Gbit/s. This is the data transfer rate. IEEE has projects investigating the standardization of 100 Gbit/s, and possibly 40 Gbit/s.

**Campus Area Network (CAN)**

A network that connects two or more LANs but that is limited to a specific and contiguous geographical area such as a college campus, industrial complex, or a military base. A CAN may be considered a type of MAN (metropolitan area network), but is generally limited to an area that is smaller than a typical MAN. This term is most often used to discuss the implementation of networks for a contiguous area. This should not be confused with a Controller Area Network. A LAN connects network devices over a relatively short distance. A networked office building, school, or home usually contains a single LAN, though sometimes one building will contain a few small LANs (perhaps one per room), and occasionally a LAN will span a group of nearby buildings. In TCP/IP networking, a LAN is often but not always implemented as a single IP subnet.

**Metropolitan Area Network (MAN)**

A Metropolitan Area Network is a network that connects two or more Local Area Networks or Campus Area Networks together but does not extend beyond the boundaries of the immediate town/city. Routers, switches and hubs are connected to create a Metropolitan Area Network.

**Wide Area Network (WAN)**

A WAN is a data communications network that covers a relatively broad geographic area (i.e. one city to another and one country to another country) and that often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer.

**Global Area Network (GAN)**

Global area networks (GAN) specifications are in development by several groups, and there is no common definition. In general, however, a GAN is a model for supporting mobile communications across an arbitrary number of wireless LANs, satellite coverage areas, etc. The key challenge in mobile communications is "handing off" the user communications from one local coverage area to the next. In IEEE Project 802, this involves a succession of terrestrial Wireless local area networks (WLAN).

**Internetwork**

Two or more networks or network segments connected using devices that operate at layer 3 (the 'network' layer) of the OSI Basic Reference Model, such as a router. Any interconnection among or between public, private, commercial, industrial, or governmental networks may also be defined as an internetwork.

In modern practice, the interconnected networks use the Internet Protocol. There are at least three variants of internetwork, depending on who administers and who participates in them:

- Intranet
- Extranet
- Internet

Intranets and extranets may or may not have connections to the Internet. If connected to the Internet, the intranet or extranet is normally protected from being accessed from the Internet without proper authorization. The Internet is not considered to be a part of the intranet or extranet, although it may serve as a portal for access to portions of an extranet.

Intranet

An intranet is a set of interconnected networks, using the Internet Protocol and uses IP-based tools such as web browsers and ftp tools, that is under the control of a single administrative entity. That administrative entity closes the intranet to the rest of the world, and allows only specific users. Most commonly, an intranet is the

internal network of a company or other enterprise. A large intranet will typically have its own web server to provide users with browseable information.

Extranet

An extranet is a network or internetwork that is limited in scope to a single organization or entity but which also has limited connections to the networks of one or more other usually, but not necessarily, trusted organizations or entities (e.g. a company's customers may be given access to some part of its intranet creating in this way an extranet, while at the same time the customers may not be considered 'trusted' from a security standpoint). Technically, an extranet may also be categorized as a CAN, MAN, WAN, or other type of network, although, by definition, an extranet cannot consist of a single LAN; it must have at least one connection with an external network.

Internet

A specific internetwork, consisting of a worldwide interconnection of governmental, academic, public, and private networks based upon the Advanced Research Projects Agency Network (ARPANET) developed by DARPA of the U.S. Department of Defense – also home to the World Wide Web (WWW) and referred to as the 'Internet' with a capital 'I' to distinguish it from other generic internetworks.

Participants in the Internet use the Internet Protocol Suite and IP Addresses allocated by address registries. Service providers and large enterprises exchange information about the reachability of their address ranges through the Border Gateway Protocol (BGP).

**Networking Devices like Hub, Repeater, Switch, Bridge, Router, Gateway & NIC Network Hardware**

Let's say you're in charge of setting up a computer network for an office building. Every desk has a computer on it, and there are also some printers and other peripheral devices sitting around. What are you going to need in terms of hardware to set up the network? You probably guessed that you're going to need some cables, but what else? Time to sit down and make a shopping list before you head out to the computer store.

**Transmission Media**

The first thing to consider is how you plan to transmit data across the network. The **transmission media** of a computer network describes the material substances that carry energy waves, which include the data being transferred. The two main categories are wired, or guided, which uses physical cables, and wireless, or unguided, which uses electromagnetic waves that can travel through a vacuum or a medium, such as air. Wireless signals do not require a physical medium, such as cables.

The most commonly used wired connections use twisted-pair cables, coaxial cables and fiber optic cables. Twisted-pair cables consist of individual copper wires that are twisted into pairs. The wires are wrapped in an insulation material. Twisted-pair cables are widely used for telephone service.

A coaxial cable consists of a copper or aluminum wire wrapped inside an insulating layer. Most cable TV companies use coaxial cables. A fiber optic cable consists of a central fiberglass core surrounded by several layers of protective material. This type of cable transmits light rather than electronic signals. A light emitting diode (LED) or laser is used to create the light pulses. The transmission speed of a fiber optic cable is much faster compared to the other cables. Most networks built today use a fiber optic cable because of its superior speed, but coaxial cable is also very common.

The most commonly used wireless connections use radio waves, microwaves and infrared waves. You obviously don't need cables for a wireless connection, but you will need other hardware for the transmission of wireless signals through the air.

**Repeaters, Hubs, Bridges and Switches**

A **repeater** is a device that receives electronic signals, cleans them and retransmits them at a higher power level. Signals transmitted over cable tend to degrade over long distances. Repeaters are needed so that the signal can travel longer distances.

A **hub** is a networking device used to connect multiple devices directly to the network using cables. Each connection is called a 'port.' The connections typically consist of a fiber optic Ethernet cable. When the hub receives data at one of its ports, it distributes the data to the other ports in the network. Typically, a hub sends all the data it receives to all the other ports.

**Switches** contain many ports to connect different network segments. They are similar to hubs, but offer greater performance. When a network contains a large

number of devices, switches are needed instead of hubs to make sure the communications between devices does not slow down. Contrary to hubs, switches send the data it receives only to specific ports.

**Bridges** are networking devices that divide up the network into different segments to manage the amount of traffic. This prevents unnecessary traffic from entering other parts of the network and reduces congestion. As a network becomes more complex, bridges make sure your network speed doesn't drop dramatically.

### Routers and Gateways

**Routers** are communication devices used to connect two different networks. A router sorts incoming data and distributes it to the correct destination. For example, if you have a network within a single office building, many different devices from within the network may access resources outside the network. The best example of this would be the Internet. A router ensures that requests from within the network for information over the Internet are distributed to the correct computer within the network.

The Internet itself uses numerous routers to direct all the traffic taking place. Such routers are typically very large and heavy-duty pieces of hardware, designed to handle huge amounts of data traffic. Routers can be used for wired connection, wireless connections or both. A router that provides a wireless connection is referred to as a 'wireless router.'

While routers are used to connect different networks, they only work if the network protocols are the same. A **gateway** interfaces networks that use different protocols. You can think of a gateway as a router that includes protocol translators. The terms 'router' and 'gateway' are often used interchangeably, but it is important to remember that only gateways make it possible to connect networks using different protocols. Since the Internet and many other computer networks use the same TCP/IP protocols, routers are sometimes all that is needed for a particular network. However, any network that also includes a mainframe system will need a gateway since this type of network uses different communication protocols.

### Modems

A **modem** is used to modulate and demodulate data signals. The term itself is a combination of the first two letters of modulator and the first three letters of demodulator. What does this really mean? While all computer data is digital, signals

over certain types of connections are analog. A modem is used to encode digital information onto an analog carrier signal and to decode the transmitted information back to digital.

**Internet Services like E-Mail, WWW, FTP, Chatting, Internet Conferencing**

**Electronic Mail (E-mail)**

•Email uses TCP/IP

•SMTP (Simple Mail Transfer Protocol)

–File attachments

•MIME (Multipurpose Internet Mail Extension)

How E-mail is delivered

–TCP/IP

–Gateways translate e-mail formats

–Routers internal/external

•E-mail software

–Mail clients –Outlook, Thunderbird, AppleMail (etc) etc.

•Create folders – search messages, address books, mailing lists

–Most e-mail software reads HTML based pages

E-Mail Basic form: username@hostname.subdomain.domain

Advantages

•Convenience

•Speed

•Inexpensive

•Printable

•Reliable

•Global

Disadvantages

•Misdirection

•Interception

•Forgery

•Overload

•Junk

•No response

**The World Wide Web (www)**

•Like an Internet library with millions of books and documents

•Non-Linear structure (documents read in any order)

•Navigate by clicking on Hypertext links

Features of the WWW

•Graphical

•Easy to use

•Cross platform

•Distributed

•Dynamic

•Interactive

**File Transfer Protocol (FTP)**

•Used to transfer files (any type) from one computer to another

•FTP sites all use Login and Password

–Anonymous FTP sites (anonymous login and E-mail address as password) or automatic logon

–FTP runs on Client/Server model

•Windows has a client command-line FTP program, but there are other, easier to use programs(WinSCP)

•FTP Daemon runs on FTP server, handling all FTP transactions, asks for account name and password

•Connection command link opened

•Downloading opens second link – data connection link

–Data connection link closes automatically after data uploaded/downloaded

•File compression used for large files

•Note that there is a secure variant known as SFTP, where S stands for “Secure”.

**Chat and Instant Messaging**

•Instant Messaging (IM) – usually just between two people, but may be more – e.g. Facebook Chat, Skype etc.

•Internet Relay Chat (IRC) – requires software on client – many chat rooms (channels)
•Though text-based, still very popular particularly in the tech community.

**Internet Conferencing or Web conferencing** refers to a service that allows conferencing events to be shared with remote locations. These are sometimes referred to as **webinars** or, for interactive conferences, **online workshops**. In general, the service is made possible by Internet technologies, particularly on TCP/IPconnections. The service allows real-timepoint-to-point communications as well as multicastcommunications from one sender to many receivers. It offers data streams of text-based messages, voice and video chat to be shared simultaneously, across geographically dispersed locations. Applications for web conferencing include meetings, training events, lectures, or short presentations from any computer.

**Electronic Online newspaper**

An **online newspaper** is the online version of a newspaper, either as a stand-alone publication or as the online version of a printed periodical.

Going online created more opportunities for newspapers, such as competing with broadcast journalism in presenting breaking news in a more timely manner. The credibility and strong brand recognition of well-established newspapers, and the close relationships they have with advertisers, are also seen by many in the newspaper industry as strengthening their chances of survival. The movement away from the printing process can also help decrease costs.

Online newspapers are much like hard-copy newspapers and have the same legal boundaries, such as laws regarding libel, privacy and copyright, also apply to online publications in most countries, like in the UK. Also in the UK the Data Protection Act applies to online newspapers and news pages, as well as the PCC rules in the UK. But the distinction was not very clear to the public in the UK as to what a blog or forum site was and what an online newspaper was. In 2007, a ruling was passed to formally regulate UK based online newspapers, news audio, and news video websites covering the responsibilities expected of them and to clear up what is, and what isn't, an online publication. news reporters are being taught to shoot video also see firstmonday orq issues issue 5 News reporters are being taught to shoot video and to write in the succinct manner necessary for the Internet news pages. Many are

learning how to implement blogs and the ruling by the UK's PCC should help this development of the internet. Some newspapers have attempted to integrate the internet into every aspect of their operations, i.e., reporters writing stories for both print and online, and classified advertisements appearing in both media; others operate websites that are more distinct from the printed newspaper. The Newspaper National Network LP is an online advertising sales partnership of the Newspaper Association of America and 25 major newspaper companies

**Online shopping** or **e-shopping** is a form of electronic commerce which allows consumers to directly buy goods or services from a seller over the Internet using a web browser. Alternative names are: e-web-store, e-shop, e-store, Internet shop, web-shop, web-store, online store, online storefront and virtual store. Mobile commerce (or m-commerce) describes purchasing from an online retailer's mobile optimized online site or app.

An online shop evokes the physical analogy of buying products or services at a bricks-and-mortar retailer or shopping center; the process is called business-to-consumer (B2C) online shopping. In the case where a business buys from another business, the process is called business-to-business (B2B) online shopping. The largest of these online retailing corporations are Alibaba, Amazon.com,and eBay. Retail success is no longer all about physical stores. This is evident because of the increase in retailers now offering online store interfaces for consumers. With the growth of online shopping, comes a wealth of new market footprint coverage opportunities for stores that can appropriately cater to offshore market demands and service requirements.

**Different types of Internet connectivity and ISP**

There are several different ways to connect to the Internet that will give you a powerful connection and let you browse the Internet in no time. Depending on where you live, not all Internet connections are available, so you may have to shop around. Here are five types of Internet connections for you to look into.

**Cable: Most Common High-Speed Internet Connection**

Cable is the most common type of high-speed Internet used today. You can easily add Internet service to your current cable package for a nominal monthly fee. Most cable companies offer different packages depending on the speed you require for

your Internet needs, and the number of computers in your house that will run off of it. They'll usually offer high-speed Internet at a reduced cost if purchased as part of a package deal.

**DSL: One of the Cheaper Options of Internet Service**

DSL Internet connection is usually provided by your local phone company and can also be modified to provide Internet service to a certain number of computers in your home. The speed is slightly different than a cable modem provides, but the cost may make the difference worthwhile. Another advantage to DSL over cable is you're not sharing the modem line with other subscribers in your area. The line is hardwired directly to your house, so there is no "down time" when a lot of people are logged on. You might need to have at least one phone line connected through the company in order to add DSL.

**WiFi: Most Convenient High-Speed Connection**

WiFi is the best option if you have a laptop or handheld devices you want to use around the house. You can set the WiFi connection up on all of your wireless devices and use them from anywhere in the house, perhaps even outside if the signal is strong enough. You need to be careful to keep your WiFi protected from hackers. Be sure to set up a password that only you know.

**3G/4G: High-Speed Wireless Connection for Smartphone and Devices**

If you already have a 3G or 4G cell phone, you can set your other wireless devices directly to the connection and use them anywhere you take your cell phone. If you use your laptop or handheld device on the road a lot, using your high speed cell is the best connection for you.

**Satellite: Viable Option for Those without Cable and DSL**

Satellite service is great if you live in a rural area that doesn't provide cable modem or DSL service. You can set up a satellite on you property and choose from several Satellite Internet Plans to log on to the Internet. The speed is usually matched to the speed of a cable modem, and will allow you to connect more than one computer to your service.

There are many different factors that should be taken into consideration when determining what type of Internet connection to use, including the number of computers that will be hooked to the modem, whether you have wireless needs, and

cost. Factor all of this in when you do your research on these five types of connections, and choose the one that is best for you.

# CHAPTER – 4

## File

File is nothing but an Electronic document. The contents can be ordinary Text or it can be an executable program. Each file is given a file name to identify it. The File name is in the form

File Name . Extension

Filename can consist of Alphabets or combinations of alphabets, numerals and special characters. Extension indicates the type of file.

Example : XY.Doc

Here File name is XY

Extension name is .DOC which indicates Document file.

## Folder

Folder contains a group of files. Folder is otherwise called as directory. Folder may have a set of files under it. It may have other folders under it also. This files and folders can be arranged in hierarchical manner or a tree like structure.

## File organization

The arrangement of records in a file is known as File Organisation. File Organisation deals with the arrangement of data items in the secondary storage devices like magnetic disk. That is, the file organisation deals with how the logical tuples (rows) of tables (relations) are organised on the physical storage medium.

For organising records efficiently in the form of a computer file, following three things are important:

(a) A logical method should be selected to organise records in a file.

(b) File structure should be so designed that it would allow quick access to needed data items.

(c) Means of adding or deleting data items or records from files must be present.

Depending on the above considerations, a file may be organised as:

(a) Sequential file

(b) Direct or random access file

(c) Indexed-sequential file

**Sequential file**

A sequential file is a file in which the records are stored in some order, say the student file contains records of students in the ascending order of roll number of students. It is not necessary that all the records of a sequential file should be in physical adjacent positions. On a magnetic tape, the records are written one after the other along the length of the tape. In case of magnetic disks, the records of a sequential file may not be in contiguous locations. The sequential order may be given with the help of pointers on each record.

Sequential files are preferable when they are to be stored on sequential access devices such as Magnetic tapes.

| 1001 | 1002 | 1003 | 1004 | 1005 | 1006 |
|---|---|---|---|---|---|
| Rec.1 | Rec.2 | Rec.3 | Rec.4 | Rec.5 | Rec.6 |

A sequential file on tape

| 1001 | 1002 | | 1003 | 1004 |
|---|---|---|---|---|
| Rec.1 | Rec.2 | | Rec.3 | Rec.4 |

| 1015 | 1017 |
|---|---|
| Rec.1 | Rec.2 |

A sequential file on disk

**The main advantages of sequential file organisation are:**

(a) File design is simple.

(b) Location or records requires only the record key.

(c) When the activity rate is high, simplicity of the accessing method makes processing efficient.

(d) Low-cost file media such as magnetic tapes can be used for storing data.

**The main drawbacks of sequential file organisation are.**

(a) Updating requires that all transaction records are sorted in the record key sequence.

(b) A new master file, physically separate and exclusive, is always created as a result of sequential updating.

(c) Addition and deletion of records is not simple.

## Direct Access File.

A sequential file is not suitable for on-line enquiry. Suppose a customer at a bank wishes to know the balance amount in his savings account. If the customer file Ls organised sequentially, the record of this customer has to be obtained by searching sequentially from the beginning. There is no way of picking out the particular record wit'hout traversing the file from the beginning and this mpy take a long time. Hence, in such situations, random access or direct access file organisation provides a means of accessing records speedily.

In random access or direct access method of file organisation, each record has its own address on the file. With the help of this physical address, the record can be directly accessed for reading or writing. The records need not be in any sequence within the file and also need not be in adjacent locations on the storage medium. Such a file cannot be created on a magnetic tape medium. Random (or direct) files

are created only on magnetic disks. Since every record can be independently accessed, every transaction can be manipulated individually.

Random access file organisation is best suited for on-line processing systems where current information is the one that is always required.

**The advantages of Direct Access file organisation are:**

(a) Immediate access to records is possible.

(b) Up-to-date information will always be available on the file.

(c) Several files can be simultaneously updated.

(d) Addition and deletion or records is not very complex.

(e) No new master file is created for updating a random access file.

**The disadvantages of Direct Access file organisation are:**

(a) Less efficient in the use of storage space.

(b) Uses a relatively expensive medium.

(c) Not well suited for batch processing.

(d) Data security is less due to direct access facility.

**Indexed Sequential File**

Some files may be required to support both batch processing and on-line processing. For example, an inventory or stock file may be updated periodically by batch processing and at the same time may have to provide current information about stock availability on-line. They can be thus, organised as indexed sequential files. Indexed Sequential file combines the advantages of sequential and direct access file organisations.

An indexed sequential file is basically a sequential file organised serially on key fields. In addition, an index is maintained which speedsúp the access of isolated records. Just as you may se indexes to locate information in book, similarly an index

is provided for the file. The file is divided into a number of blocks and the highest key in each block is indexed.

| **Key** | **Starting address of the block** |
|---|---|
| 125 | 12 |
| 860 | 19 |
| 1420 | 24 |
| 1600 | 42 |
| 1829 | 49 |
| 2225 | 159 |
| 2890 | 165 |
| 3200 | 807 |

Indexed sequential files are also known as Indexed Sequential Access Method (ISAM) files.

Within each block, the record is searched sequentially. This method is much faster than searching the entire file sequentially. It is also possible to have more than one level of indexing to make the search process faster.

The main advantage of indexed sequential file organisation is that it is suitable for both sequential and on-line or direct access processing.

**The main disadvantages of this organisation are:**

(a) Less efficient in the use of storage space.

(b) Additions and deletions of records are more complex as they effect both the index and the record number in the file.

## Deciding on a File Organisation

The major factors to be considered while deciding which file organisation should be chosen are the following:

(a) Percentage of actual records processed in a day. If large number of records are accessed at a time, direct access file organisation should be used. If very few records are accessed, sequential file organisation will be more suitable and cheaper.

(b) Files that are frequently updated must be stored on a direct access storage device, such as disk.

(c) Selection of file organisation also depend on the mode of processing i.e. where the system requires an online processing or batch processing.

## DATA PROCESSING

Data Processing means, processing the input data to produce some meaningful and purposeful information. Computer has the capability of processing high-volume of data in less time with higher accuracy. Hence the data processing performed by computer is sometimes called Electronic Data Processing (EDP). Data processing involves 5 distinct steps.

- Data capturing
- Data validation
- Processing / Execution
- Data storage
- Data Retrieval/Out generation

Data capturing encompasses the activities of inputing data to the computer. Before giving input to the system the required data are to be first identified and put in the defined format called source data layout. The aim of this layout is to have faster data entry. To reduce the volume of data and also have better organisation and easy

access to data those can suitably be coded. After data are ready they can be entered to the computer through keyboard. This is sometimes called data capture through intelligent terminal. The other form of data capture is through scanners or optical devices. In this type of data capturing data are not entered rather data are captured from the source document or paper as it is Photographs, fingerprints, signatures, objective multiple type answers in answer papers etc. are captured through this method. Another form of data capturing is through some interfacing devices from where data can be transferred directly to the computer. Example of this is Electronic cash Registers used in shops and cash counters.

After data capturing, the data is validated. Data validation involves checking of input data to fit to requirements or specifications. For example price of a book can be numeric only. If by mistake Alphabetic data are entered then it is checked and error is shown to revalidate the data. This prevents unwanted and unspecified data to enter into the system and causing errors.

The valid input data are stored in file or database and processed as per the instructions. The instructions are put in programs or software. This software or program, when executed does the processing of input data and produces the output. Again outputs are stored in files in memory.

After processing of data, the outputs are produced. The outputs may be formed in different ways depending on the requirements and specifications. The same set of data can be printed in tabular form or in form of graphs. There are variety of ways for presenting data. The printed output is sometimes called hardcopy. There can be provision of answering to queries of user where the answer is displayed on the monitor screen itself. So it depends on the requirements of user.

An important step of Data Processing is maintaining Database. Database is nothing but collection of data which is controlled centrally with many provisions of data security. This is where, normally data are stored for reference. The input data and output data are stored in database. Even after processing of data and producing of data is over, database is maintained properly with safety for future needs and reference. The important tool of data processing is file. File is nothing but an electronic document where data can be stored. Depending on the type of file structure and organisation the data access speed varies.

# CHAPTER - 5

## Algorithm & Flowchart

Algorithm is defined as the step by step solution of problem in user's language.

It is considered as an effective procedure for solving a problem in finite number of steps. The characteristics of Algorithm are

- Precise
- Unambiguous
- Finite termination
- Unique solution

Once algorithm is written, it can be coded into a program using any programming language. Algorithm uses 3 different constructs

- Sequence
- Branching or Decision making
- Repetition

Sequence says that instructions are to be executed in what order or sequence. Branching involves testing of condition and based on the outcome of the condition testing different instructions are executed. Repetition means one or more instructions shall be repeated for a number of times. This is otherwise called as loop. There are different types of loops such as

While- do , do-while , for

Example:

1. Algorithm to find out sum of two numbers to be taken as input.

Step-1 Read the 1$^{st}$ number x

Step-2 Read the 2$^{nd}$ number y

Step-3 Sum=x+y

Step-4 Print Sum

This is an example where only sequence is exhibited

2. Algorithm to find out larger between numbers to be taken as input.

Step-1 Read the 1st number x

Step-2 Read the 2nd number y

Step-3 If x > y

Then Print x

Else if x< y

Then Print y

Else Print " Both are Equal "

This is an example where Branching is exhibited

3. Algorithm to find out sum of first 10 natural numbers.

Step-1 i=1, Sum=0

Step-2 Repeat step 3 and 4 while i<>10

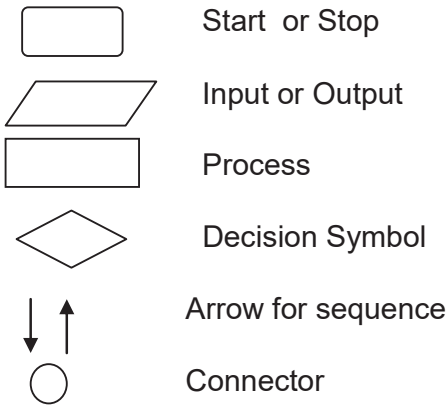
Step-3 Sum= Sum+i

Step-4 i=i+1

Step-5 Print Sum

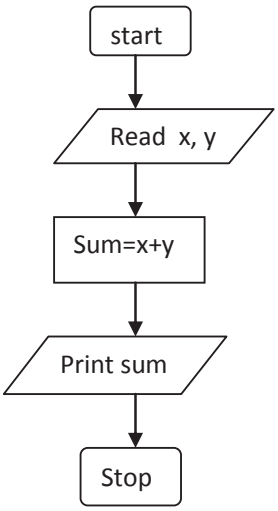
This is an example where Repetition is exhibited

Flowchart is a graphical or symbolic representation of the process of solution to a problem or algorithm. It helps to visualize the complex logic of the solution of the problem in a simplified manner through diagrammatic representation. Each step of the algorithm is presented using a symbol and a short description. The different symbols used for the flowchart are

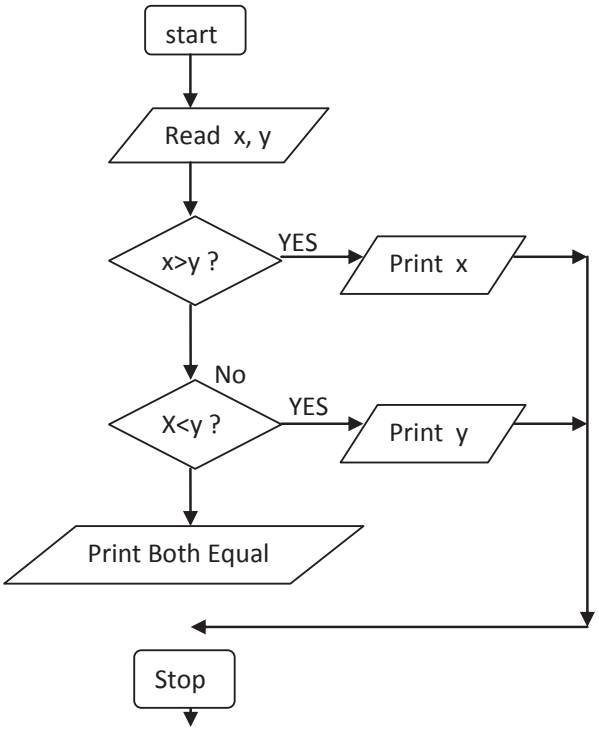



Example

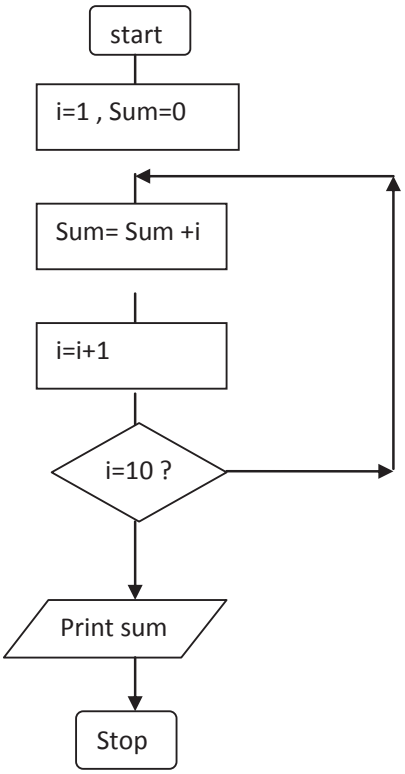
1. Flowchart to find out sum of two numbers to be taken as input



2. Flowchart to find out larger between two numbers to be taken as input




3. Flowchart to find out sum of first 10 natural numbers



**Pseuodocode**

It is a concise description algorithm in English language that uses programming language constructs. It contains outlines of the program that can be easily converted to program. It focuses on the logic of the algorithm without giving stress on the syntax of programming language. This is meant for understanding the logic of the program easily. Flowchart can be considered as an alternative to pseudocode. Several constructs/key words of programming language can be used in the algorithm to write the pseudocode. Some of them are

If ... Endif

Do while ... enddo

While ... endwhile

Repeat ... until

For ... endfor

Case .... endcase

Call

Return

**Programming Language**

Programming language is a tool to express the logic or instructions for understanding of the computer. Any programming language has two components:

- Syntax
- Semantics

Syntax refers to the rules to be followed for writing valid program statements. Compiler can detect errors in syntax while compiling the program.

Semantics is associated with logic of the program. Compiler can not detect the semantic error. The user of programmer can diagnose semantic error.

There are good number of High level languages, each meant for specific area of data processing. Commonly known languages are BASIC,FORTRAN, COBOL, Pascal, C, C++ etc. While FORTRAN is good for Numerical and scientific calculation, COBOL is good for Business applications involving large amount of data handling.

**Generations of Programming Language**

The Programming languages can be classified into 4 generations

1st Generation: Machine Language

2nd Generation: Assembly Language

3rd Generation: High Level Language

4th Generation: Very High Level Language

Machine Level language contains instructions in binary form i.e. in 0s and 1s. Thus writing instruction was very difficult and needs heavy expertise. This was used in early days computers.

Assembly level language instructions were written using symbolic codes known as mnemonics. In comparison to Machine language it is relatively easier to write program, but still it requires lot of expertise. A translator called assembler is used to translate assembly language program to machine level language.

High level language contains instructions in English like words so that user will feel easier to formulate and write the logical statements of the program. Here the logic may spread over multiple statements as against a single statement in assembly language. It uses a translator called compiler for translation of High level language program to machine level language program. There are many High level languages used for programming such as BASIC, FORTRAN, COBOL, PASCAL, C, C++ etc.

Very High Level language other wise called as 4GL uses nonprocedural logical statements. A typical example of 4GL is the query language such as SQL.

**Structured Programming Language**

Structured Programming is also known as Modular Programming. In this type of programming technique, the program shall be broken into several modules. This helps in managing memory efficiently as the required module of the program will be

loaded into the memory only and not the entire program. This will also enhance code reuse. Writing, understanding, debugging and modifying the individual module of the program is also easier.

# CHAPTER – 6

**The C Language**

C is a professional programmer's language. It was designed to get in one's way as little as possible. Kernighan and Ritchie wrote the original language definition in their book, *The C Programming Language* (below), as part of their research at AT&T. Unix and C++ emerged from the same labs. For several years I used AT&T as my long distance carrier in appreciation of all that CS research, but hearing "thank you for using AT&T" for the millionth time has used up that good will.

The C Language

C is a professional programmer's language. It was designed to get in one's way as little as possible. Kernighan and Ritchie wrote the original language definition in their book, *The C Programming Language* (below), as part of their research at AT&T. Unix and C++ emerged from the same labs. For several years I used AT&T as my long distance carrier in appreciation of all that CS research, but hearing "thank you for using AT&T" for the millionth time has used up that good will.

Some languages are forgiving. The programmer needs only a basic sense of how things work. Errors in the code are flagged by the compile-time or run-time system, and the programmer can muddle through and eventually fix things up to work correctly. The C language is not like that.The C programming model is that the programmer knows exactly what they want to do and how to use the language constructs to achieve that goal. The language lets the expert programmer express what they want in the minimum time by staying out of their way. C is "simple" in that

the number of components in the language is small-- If two language features accomplish more-or-less the same thing, C will include only one. C's syntax is terse and the language does not restrict what is "allowed" -- the programmer can pretty much do whatever they want C's type system and error checks exist only at compile-time. The compiled code runs in a stripped down run-time model with no safety checks for bad type casts, bad array indices, or bad pointers. There is no garbage collector to manage memory. Instead the programmer manages heap memory manually. All this makes C fast but fragile.

Analysis -- Where C Fits

Because of the above features, C is hard for beginners. A feature can work fine in one context, but crash in another. The programmer needs to understand how the features work and use them correctly. On the other hand, the number of features is pretty small. Like most programmers, I have had some moments of real loathing for the C language. Itcan be irritatingly obedient -- you type something incorrectly, and it has a way of compiling fine and just doing something you don't expect at run-time. However, as I have become a more experienced C programmer, I have grown to appreciate C's straight-to-the point style. I have learned not to fall into its little traps, and I appreciate its simplicity.

Perhaps the best advice is just to be careful. Don't type things in you don't understand. Debugging takes too much time. Have a mental picture (or a real drawing) of how your C code is using memory. That's good advice in any language, but in C it's critical.

Perl and Java are more "portable" than C (you can run them on different computers without a recompile). Java and C++ are more structured than C. Structure is useful for large projects. C works best for small projects where performance is important and the progammers have the time and skill to make it work in C. In any case, C is a very popular and influential language. This is mainly because of C's clean (if minimal) style, it's lack of annoying or regrettable constructs, and the relative ease of writing a C compiler.

Other Resources

• *The C Programming Language*, 2nd ed., by Kernighan and Ritchie. The thin book which for years was the bible for all C programmers. Written by the original designers of the language. The explanations are pretty short, so this book is better as a reference than for beginners.

• http://cslibrary.stanford.edu/102/ Pointers and Memory -- Much more detail about local memory, pointers, reference parameters, and heap memory than in this article, and memory is really the hardest part of C and C++.

• http://cslibrary.stanford.edu//103/ Linked List Basics -- Once you understand the basics of pointers and C, these problems are a good way to get more practice.

Basic Types and Operators

C provides a standard, minimal set of basic data types. Sometimes these are called “primitive" types. More complex data structures can be built up from these basic types.

Integer Types

The "integral" types in C form a family of integer types. They all behave like integers and can be mixed together and used in similar ways. The differences are due to the different number of bits ("widths") used to implement each type -- the wider types can store a greater ranges of values.

char ASCII character -- at least 8 bits. Pronounced "car". As a practical matter char is basically always a byte which is 8 bits which is enough to store a single ASCII character. 8 bits provides a signed range of -128..127 or an unsigned range is 0..255. char is also required to be the "smallest addressable unit" for the machine -- each byte in memory has its own address.

short Small integer -- at least 16 bits which provides a signed range of -32768..32767. Typical size is 16 bits. Not used so much.

int Default integer -- at least 16 bits, with 32 bits being typical. Defined to be the "most comfortable" size for the computer. If you do not really care about the range for an integer variable, declare it int since that is likely to be an appropriate size (16 or 32 bit) which works well for that machine.

Long Large integer -- at least 32 bits. Typical size is 32 bits which gives a signed range of about -2 billion..+2 billion. Some compilers support "long long" for 64 bit ints.

The integer types can be preceded by the qualifier unsigned which disallows representing negative numbers, but doubles the largest positive number representable. For example, a 16 bit implementation of short can store numbers in the range -32768..32767, while unsigned short can store 0..65535. You can think of pointers as being a form of unsigned long on a machine with 4 byte pointers. In my opinion, it's best to avoid using unsigned unless you really need to. It tends to cause more misunderstandings and problems than it is worth.

Extra: Portability Problems

Instead of defining the exact sizes of the integer types, C defines lower bounds. This makes it easier to implement C compilers on a wide range of hardware. Unfortunately it occasionally leads to bugs where a program runs differently on a 16-bit-int machine than it runs on a 32-bit-int machine. In particular, if you are designing a function that will be implemented on several different machines, it is a good idea to use typedefs to set up types like Int32 for 32 bit int and Int16 for 16 bit int. That way you can prototype a function Foo(Int32) and be confident that the typedefs for each machine will be set so that the function really takes exactly a 32 bit int. That way the code will behave the same on all the different machines.

char Constants

A char constant is written with single quotes (') like 'A' or 'z'. The char constant 'A' is really just a synonym for the ordinary integer value 65 which is the ASCII value for uppercase 'A'. There are special case char constants, such as '\t' for tab, for characters which are not convenient to type on a keyboard.

'A' uppercase 'A' character

'\n' newline character

'\t' tab character

'\0' the "null" character -- integer value 0 (different from the char digit '0')

'\012' the character with value 12 in octal, which is decimal 10

int Constants

Numbers in the source code such as 234 default to type int. They may be followed by an 'L' (upper or lower case) to designate that the constant should be a long such as 42L. An integer constant can be written with a leading 0x to indicate that it is expressed in hexadecimal -- 0x10 is way of expressing the number 16. Similarly, a constant may be written in octal by preceding it with "0" -- 012 is a way of expressing the number 10.

This page lists C operators in order of *precedence* (highest to lowest). Their *associativity* indicates in what order operators of equal precedence in an expression are applied.

| **Operator** | **Description** | **Associativity** |
|---|---|---|
| ()<br>[]<br>.<br>-><br>++ -- | Parentheses (function call) (see Note 1)<br>Brackets (array subscript)<br>Member selection via object name<br>Member selection via pointer<br>Postfix increment/decrement (see Note 2) | left-to-right |
| ++ --<br>+ -<br>! ~<br>(*type*)<br>*<br>&<br>sizeof | Prefix increment/decrement<br>Unary plus/minus<br>Logical negation/bitwise complement<br>Cast (change *type*)<br>Dereference<br>Address<br>Determine size in bytes | right-to-left |
| * / % | Multiplication/division/modulus | left-to-right |
| + - | Addition/subtraction | left-to-right |
| << >> | Bitwise shift left, Bitwise shift right | left-to-right |
| < <=<br>> >= | Relational less than/less than or equal to<br>Relational greater than/greater than or equal to | left-to-right |

| | | |
|---|---|---|
| == != | Relational is equal to/is not equal to | left-to-right |
| & | Bitwise AND | left-to-right |
| ^ | Bitwise exclusive OR | left-to-right |
| \| | Bitwise inclusive OR | left-to-right |
| && | Logical AND | left-to-right |
| \|\| | Logical OR | left-to-right |
| ?: | Ternary conditional | right-to-left |
| =<br>+= -=<br>*= /=<br>%= &=<br>^= \|=<br><<= >>= | Assignment<br>Addition/subtraction assignment<br>Multiplication/division assignment<br>Modulus/bitwise AND assignment<br>Bitwise exclusive/inclusive OR assignment<br>Bitwise shift left/right assignment | right-to-left |
| , | Comma (separate expressions) | left-to-right |

Type Combination and Promotion

The integral types may be mixed together in arithmetic expressions since they are all basically just integers with variation in their width. For example, char and int can be combined in arithmetic expressions such as ('b' + 5). How does the compiler deal with the different widths present in such an expression? In such a case, the compiler "promotes" the smaller type (char) to be the same size as the larger type (int) before combining the values. Promotions are determined at compile time based purely on the types of the values in the expressions. Promotions do not lose information -- they always convert from a type to compatible, larger type to avoid losing information.

Pitfall -- int Overflow

A piece of code which tried to compute the number of bytes in a buffer with the expression (k * 1024) where k was an int representing the number of kilobytes wanted. Unfortunately this was on a machine where int happened to be 16 bits. Since k and 1024 were both int, there was no promotion. For values of k >= 32, the product was too big to fit in the 16 bit int resulting in an overflow. The compiler can do whatever it wants in overflow situations -- typically the high order bits just vanish. One way to fix the code was to rewrite it as (k * 1024L) -- the long constant forced the promotion of the int. This was not a fun bug to track down -- the expression sure looked reasonable in the source code. Only stepping past the key line in the debugger showed the overflow problem. "Professional Programmer's Language." This example also demonstrate s the way that C only promotes based on the types in an expression. The compiler does not consider the values 32 or 1024 to realize that the operation will overflow (in general, the values don't exist until run time anyway). The compiler just looks at the compile time types, int and int in this case, and thinks everything is fine.

Floating point Types

float Single precision floating point number typical size: 32 bits double Double precision floating point number typical size: 64 bits long double Possibly even bigger floating point number (somewhat obscure) Constants in the source code such as 3.14 default to type double unless the are suffixed with an 'f' (float) or 'l' (long double). Single precision equates to about 6 digits of precision and double is about 15 digits of precision. Most C programs use double for their computations. The main reason to use float is to save memory if many numbers need to be stored. The main thing to remember about floating point numbers is that they are inexact. For example, what is the value of the following double expression?

(1.0/3.0 + 1.0/3.0 + 1.0/3.0) // is this equal to 1.0 exactly?

The sum may or may not be 1.0 exactly, and it may vary from one type of machine to another. For this reason, you should never compare floating numbers to eachother for equality (==) -- use inequality (<) comparisons instead. Realize that a correct C program run on different computers may produce slightly different outputs in the rightmost digits of its floating point computations.

Comments

Comments in C are enclosed by slash/star pairs: /* .. comments .. */ which may cross multiple lines. C++ introduced a form of comment started by two slashes and extending to the end of the line: // comment until the line end

The // comment form is so handy that many C compilers now also support it, although it is not technically part of the C language.

Along with well-chosen function names, comments are an important part of well written code. Comments should not just repeat what the code says. Comments should describe what the code accomplishes which is much more interesting than a translation of what each statement does. Comments should also narrate what is tricky or non-obvious about a section of code.

Variables

As in most languages, a variable declaration reserves and names an area in memory at run time to hold a value of particular type. Syntactically, C puts the type first followed by the name of the variable. The following declares an int variable named "num" and the 2nd line stores the value 42 into num.

```
int num;

num = 42;
```

num 42

A variable corresponds to an area of memory which can store a value of the given type.

Making a drawing is an excellent way to think about the variables in a program. Draw each variable as box with the current value inside the box. This may seem like a "beginner" technique, but when I'm buried in some horribly complex programming problem, invariably resort to making a drawing to help think the problem through.

Variables, such as num, do not have their memory cleared or set in any way when they are allocated at run time. Variables start with random values, and it is up to the program to set them to something sensible before depending on their values.

Names in C are case sensitive so "x" and "X" refer to different variables. Names can contain digits and underscores (_), but may not begin with a digit. Multiple variables

can be declared after the type by separating them with commas. C is a classical "compiletime" language -- the names of the variables, their types, and their implementations are all flushed out by the compiler at compile time (as opposed to figuring such details out at run time like an interpreter).

float x, y, z, X;

Assignment Operator =

The assignment operator is the single equals sign (=).

i = 6;

i = i + 1;

The assignment operator copies the value from its right hand side to the variable on its left hand side. The assignment also acts as an expression which returns the newly assigned value. Some programmers will use that feature to write things like the following.

y = (x = 2 * x); // double x, and also put x's new value in y

Truncation

The opposite of promotion, truncation moves a value from a type to a smaller type. In that case, the compiler just drops the extra bits. It may or may not generate a compile time warning of the loss of information. Assigning from an integer to a smaller integer (e.g.. long to int, or int to char) drops the most significant bits. Assigning from a floating point type to an integer drops the fractional part of the number.

char ch;

int i;

i = 321;

ch = i; // truncation of an int value to fit in a char

// ch is now 65

The assignment will drop the upper bits of the int 321. The lower 8 bits of the number 321 represents the number 65 (321 - 256). So the value of ch will be (char)65 which happens to be 'A'.

The assignment of a floating point type to an integer type will drop the fractional part of the number. The following code will set i to the value 3. This happens when assigning a floating point number to an integer or passing a floating point number to a function which takes an integer.

double pi;

int i;

pi = 3.14159;

i = pi; // truncation of a double to fit in an int

// i is now 3

Pitfall -- int vs. float Arithmetic

Here's an example of the sort of code where int vs. float arithmetic can cause problems. Suppose the following code is supposed to scale a homework score in the range 0..20 to be in the range 0..100.

{

int score;

...// suppose score gets set in the range 0..20 somehow

7

score = (score / 20) * 100; // NO -- score/20 truncates to 0

...

Unfortunately, score will almost always be set to 0 for this code because the integer division in the expression (score/20) will be 0 for every value of score less than 20.

The fix is to force the quotient to be computed as a floating point number...

score = ((double)score / 20) * 100; // OK -- floating point division from cast

score = (score / 20.0) * 100; // OK -- floating point division from 20.0

score = (int)(score / 20.0) * 100; // NO -- the (int) truncates the floating

// quotient back to 0

No Boolean -- Use int

C does not have a distinct boolean type-- int is used instead. The language treats integer

0 as false and all non-zero values as true. So the statement...

i = 0;

while (i - 10) {

...

will execute until the variable i takes on the value 10 at which time the expression (i - 10) will become false (i.e. 0). (we'll see the while() statement a bit later)

Mathematical Operators

C includes the usual binary and unary arithmetic operators. See the appendix for the table of precedence. Personally, I just use parenthesis liberally to avoid any bugs due to a misunderstanding of precedence. The operators are sensitive to the type of the operands.

So division (/) with two integer arguments will do integer division. If either argument is a float, it does floating point division. So (6/4) evaluates to 1 while (6/4.0)

evaluates to 1.5 -- the 6 is promoted to 6.0 before the division.

+ Addition

- Subtraction

/ Division

* Multiplication

% Remainder (mod)

Unary Increment Operators: ++ --

The unary ++ and -- operators increment or decrement the value in a variable. There are "pre" and "post" variants for both operators which do slightly different things (explained below)

*var*++ increment "post" variant

++*var* increment "pre" variant

8

*var*-- decrement "post" variant

--*var* decrement "pre" variant

int i = 42;

i++; // increment on i

// i is now 43

i--; // decrement on i

// i is now 42

Pre and Post Variations

The Pre/Post variation has to do with nesting a variable with the increment or decrement operator inside an expression -- should the entire expression represent the value of the variable before or after the change? I never use the operators in this way (see below), but an example looks like...

int i = 42;

int j;

j = (i++ + 10);

// i is now 43

// j is now 52 (NOT 53)

j = (++i + 10)

// i is now 44

// j is now 54

Relational Operators

These operate on integer or floating point values and return a 0 or 1 boolean value.

== Equal

!= Not Equal

> Greater Than

< Less Than

>= Greater or Equal

<= Less or Equal

To see if x equals three, write something like:

if (x == 3) ...

Pitfall = ==

An absolutely classic pitfall is to write assignment (=) when you mean comparison (==).

This would not be such a problem, except the incorrect assignment version compiles fine because the compiler assumes you mean to use the value returned by the assignment. This is rarely what you want

if (x = 3) ...

This does not test if x is 3. This sets x to the value 3, and then returns the 3 to the if for testing. 3 is not 0, so it counts as "true" every time. This is probably the single most common error made by beginning C programmers. The problem is that the compiler is no help -- it thinks both forms are fine, so the only defense is extreme vigilance when coding. Or write "= □□==" in big letters on the back of your hand before coding. This mistake is an absolute classic and it's a bear to debug. Watch Out! And need I say:

"Professional Programmer's Language."

Logical Operators

The value 0 is false, anything else is true. The operators evaluate left to right and stop as soon as the truth or falsity of the expression can be deduced. (Such operators are called "short circuiting") In ANSI C, these are furthermore guaranteed to use 1 to represent true, and not just some random non-zero bit pattern. However, there are many C programs out there which use values other than 1 for true (non-zero pointers for example), so when programming, do not assume that a true boolean is necessarily 1 exactly.

! Boolean not (unary)

&& Boolean and

|| Boolean or

Bitwise Operators

C includes operators to manipulate memory at the bit level. This is useful for writing low level hardware or operating system code where the ordinary abstractions of numbers, characters, pointers, etc... are insufficient -- an increasingly rare need. Bit manipulation code tends to be less "portable". Code is "portable" if with no programmer intervention it compiles and runs correctly on different types of computers. The bitwise operations are typically used with unsigned types. In particular, the shift operations are guaranteed to shift 0 bits into the newly vacated positions when used on unsigned values.

~ Bitwise Negation (unary) – flip 0 to 1 and 1 to 0 throughout

& Bitwise And

| Bitwise Or

^ Bitwise Exclusive Or

>> Right Shift by right hand side (RHS) (divide by power of 2)

<< Left Shift by RHS (multiply by power of 2)

Do not confuse the Bitwise operators with the logical operators. The bitwise connectives are one character wide (&, |) while the boolean connectives are two characters wide (&&, ||). The bitwise operators have higher precedence than the boolean operators. The compiler will never help you out with a type error if you use &

when you meant &&. As far as the type checker is concerned, they are identical--they both take and produce integers since there is no distinct boolean type.

Other Assignment Operators

In addition to the plain = operator, C includes many shorthand operators which represents variations on the basic =. For example "+=" adds the right hand side to the left hand side.

x = x + 10; can be reduced to x += 10;. This is most useful if x is a long expression such as the following, and in some cases it may run a little faster. person->relatives.mom.numChildren += 2; // increase children by 2

Here's the list of assignment shorthand operators...

+=, -= Increment or decrement by RHS

*=, /= Multiply or divide by RHS

%= Mod by RHS

>>= Bitwise right shift by RHS (divide by power of 2)

<<= Bitwise left shift RHS (multiply by power of 2)

&=, |=, ^= Bitwise and, or, xor by RHS

Control Structures

C uses curly braces ({}) to group multiple statements together. The statements execute in order. Some languages let you declare variables on any line (C++). Other languages insist that variables are declared only at the beginning of functions (Pascal). C takes the middle road -- variables may be declared within the body of a function, but they must follow a '{'. More modern languages like Java and C++ allow you to declare variables on any line, which is handy.

If Statement

Both an if and an if-else are available in C. The <*expression*> can be any valid expression. The parentheses around the expression are required, even if it is just a single variable.

if (<expression>) <statement> // simple form with no {}'s or else clause if (<expression>) { // simple form with {}'s to group statements

<statement>

<statement>

}

if (<expression>) { // full then/else form

<statement>

}

else {

<statement>

}

Conditional Expression -or- The Ternary Operator

The conditional expression can be used as a shorthand for some if-else statements. The general syntax of the conditional operator is:

<expression1> ? <expression2> : <expression3>

This is an expression, not a statement, so it represents a value. The operator works by evaluating expression1. If it is true (non-zero), it evaluates and returns expression2 .

Otherwise, it evaluates and returns expression3.

The classic example of the ternary operator is to return the smaller of two variables. Every once in a while, the following form is just what you needed. Instead of...

if (x < y) {

min = x;

}

else {

min = y;

}

12

You just say...

min = (x < y) ? x : y;

Switch Statement

The switch statement is a sort of specialized form of if used to efficiently separate different blocks of code based on the value of an integer. The switch expression is evaluated, and then the flow of control jumps to the matching const-expression case. The case expressions are typically int or char constants. The switch statement is probably the single most syntactically awkward and error-prone features of the C language.

switch (<expression>) {

case <const-expression-1>:

<statement>

break;

case <const-expression-2>:

<statement>

break;

case <const-expression-3>: // here we combine case 3 and 4

case <const-expression-4>:

<statement>

break;

default: // optional

<statement>

}

Each constant needs its own case keyword and a trailing colon (:). Once execution has jumped to particular case, the program will keep running through all the cases from that point down -- this so called "fall through" operation is used in the above example so that expression-3 and expression-4 run the same statements. The explicit break statements are necessary to exit the switch. Omitting the break statements is a common error – it compiles, but leads to inadvertent fall-through behavior.

Why does the switch statement fall-through behavior work the way it does? The best explanation I can think of is that originally C was developed for an audience of assembly language programmers. The assembly language programmers were used to the idea of a jump table with fall-through behavior, so that's the way C does it (it's also relatively easy to implement it this way.) Unfortunately, the audience for C is now quite different, and the fall-through behavior is widely regarded as a terrible part of the language.

While Loop

The while loop evaluates the test expression before every loop, so it can execute zero times if the condition is initially false. It requires the parenthesis like the if.

while (<expression>) {

<statement>

}

Do-While Loop

Like a while, but with the test condition at the bottom of the loop. The loop body will always execute at least once. The do-while is an unpopular area of the language, most everyone tries to use the straight while if at all possible.

do {

<statement>

} while (<expression>)

For Loop

The for loop in C is the most general looping construct. The loop header contains three parts: an initialization, a continuation condition, and an action.

for (<initialization>; <continuation>; <action>) {

<statement>

}

The initialization is executed once before the body of the loop is entered. The loop continues to run as long as the continuation condition remains true (like a while). After every execution of the loop, the action is executed. The following example executes 10 times by counting 0..9. Many loops look very much like the following...

for (i = 0; i < 10; i++) {

<statement>

}

C programs often have series of the form 0..(some_number-1). It's idiomatic in C for the above type loop to start at 0 and use < in the test so the series runs up to but not equal to the upper bound. In other languages you might start at 1 and use <= in the test. Each of the three parts of the for loop can be made up of multiple expressions separated by commas. Expressions separated by commas are executed in order, left to right, and represent the value of the last expression. (See the string-reverse example below for a demonstration of a complex for loop.)

Break

The break statement will move control outside a loop or switch statement. Stylistically speaking, break has the potential to be a bit vulgar. It's preferable to use a straight while with a single test at the top if possible. Sometimes you are forced to use a break because the test can occur only somewhere in the midst of the statements in the loop body. To keep the code readable, be sure to make the break obvious -- forgetting to account for the action of a break is a traditional source of bugs in loop behavior.

while (<expression>) {

<statement>

<statement>

if (<condition which can only be evaluated here>)

break;

<statement>

<statement>

}

// control jumps down here on the break

The break does not work with if. It only works in loops and switches. Thinking that a break refers to an if when it really refers to the enclosing while has created some high quality bugs. When using a break, it's nice to write the enclosing loop to iterate in the most straightforward, obvious, normal way, and then use the break to explicitly catch the exceptional, weird cases.

Continue

The continue statement causes control to jump to the bottom of the loop, effectively skipping over any code below the continue. As with break, this has a reputation as being vulgar, so use it sparingly. You can almost always get the effect more clearly using an if inside your loop.

while (<expression>) {

...

if (<condition>)

continue;

...

...

// control jumps here on the continue

}

# CHAPTER - 7

## Complex Data Types

C has the usual facilities for grouping things together to form composite types—arrays and records (which are called "structures"). The following definition declares a type called "struct fraction" that has two integer sub fields named "numerator" and "denominator". If you forget the semicolon it tends to produce a syntax error in whatever thing follows the struct declaration.

```
struct fraction {
int numerator;
int denominator;
}; // Don't forget the semicolon!
```

This declaration introduces the type struct fraction (both words are required) as a new type. C uses the period (.) to access the fields in a record. You can copy two records of the same type using a single assignment statement, however == does not work on structs.

```
struct fraction f1, f2; // declare two fractions
f1.numerator = 22;
f1.denominator = 7;
f2 = f1; // this copies over the whole struct
```

## Arrays

The simplest type of array in C is one which is declared and used in one place. There are more complex uses of arrays which I will address later along with pointers. The following declares an array called scores to hold 100 integers and sets the first

and last elements. C arrays are always indexed from 0. So the first int in scores array is scores[0] and the last is scores[99].

int scores[100];

scores[0] = 13; // set first element

scores[99] = 42; // set last element

0

scores

Index 1 2 99

Someone else’s memory off either end of the array — do not read or write this memory. There is space for each int element in the scores array — this element is referred to as scores[0].

-5673 22541 42

These elements have random values because the code has not yet initialized them to anything. The name of the array refers to the whole array. (implementation) it works by representing a pointer to the start of the array.

It's a very common error to try to refer to non-existent scores[100] element. C does not do any run time or compile time bounds checking in arrays. At run time the code will just access or mangle whatever memory it happens to hit and crash or misbehave in some unpredictable way thereafter. "Professional programmer's language." The convention of numbering things 0..(number of things - 1) pervades the language. To best integrate with C and other C programmers, you should use that sort of numbering in your own data structures as well.

## Multidimensional Arrays

The following declares a two-dimensional 10 by 10 array of integers and sets the first and last elements to be 13.

```
int board [10][10];

board[0][0] = 13;

board[9][9] = 13;
```

The implementation of the array stores all the elements in a single contiguous block of memory. The other possible implementation would be a combination of several distinct one dimensional arrays -- that's not how C does it. In memory, the array is arranged with the elements of the rightmost index next to each other. In other words, board[1][8] comes right before board[1][9] in memory. (highly optional efficiency point) It's typically efficient to access memory which is near other recently accessed memory. This means that the most efficient way to read through a chunk of the array is to vary the rightmost index the most frequently since that will access elements that are near each other in memory.

## Array of Structs

The following declares an array named "numbers" which holds 1000 struct fraction's.

```
struct fraction numbers[1000];

numbers[0].numerator = 22; /* set the 0th struct fraction */

numbers[0].denominator = 7;
```

Here's a general trick for unraveling C variable declarations: look at the right hand side and imagine that it is an expression. The type of that expression is the left hand side. For the above declarations, an expression which looks like the right hand side (numbers[1000], or really anything of the form numbers[...]) will be the type on the left hand side (struct fraction).

## Pointers

A pointer is a value which represents a reference to another value sometimes known as the pointer's "pointee". Hopefully you have learned about pointers somewhere else, since the preceding sentence is probably inadequate explanation. This discussion will concentrate on the syntax of pointers in C -- for a much more complete discussion of pointers and their use see http://cslibrary.stanford.edu/102/, Pointers and Memory.

**Syntax**

Syntactically C uses the asterisk or "star" (*) to indicate a pointer. C defines pointer types based on the type pointee. A char* is type of pointer which refers to a single char.

A struct fraction* is type of pointer which refers to a struct fraction.

int* intPtr; // declare an integer pointer variable intPtr

char* charPtr; // declares a character pointer --

// a very common type of pointer

// Declare two struct fraction pointers

// (when declaring multiple variables on one line, the *

// should go on the right with the variable)

struct fraction *f1, *f2;

**The Floating "*"**

In the syntax, the star is allowed to be anywhere between the base type and the variable name. Programmer's have their own conventions-- I generally stick the * on the left with the type. So the above declaration of intPtr could be written equivalently...

int *intPtr; // these are all the same

int * intPtr;

int* intPtr;

**Pointer Dereferencing**

We'll see shortly how a pointer is set to point to something -- for now just assume the pointer points to memory of the appropriate type. In an expression, the unary * to the left of a pointer dereferences it to retrieve the value it points to.

There's an alternate, more readable syntax available for dereferencing a pointer to a struct. A "->" at the right of the pointer can access any of the fields in the struct. So the reference to the numerator field could be written f1->numerator.

Here are some more complex declarations...

struct fraction** fp; // a pointer to a pointer to a struct fraction

struct fraction fract_array[20]; // an array of 20 struct fractions

struct fraction* fract_ptr_array[20]; // an array of 20 pointers to

// struct fractions

One nice thing about the C type syntax is that it avoids the circular definition problems which come up when a pointer structure needs to refer to itself. The following definition defines a node in a linked list. Note that no preparatory declaration of the node pointer type is necessary.

struct node {

int data;

struct node* next;

};

**The & Operator**

The & operator is one of the ways that pointers are set to point to things. The & operator computes a pointer to the argument to its right. The argument can be any variable which takes up space in the stack or heap (known as an "LValue" technically). So &i and &(f1->numerator) are ok, but &6 is not. Use & when you have some memory, and you want a pointer to that memory.

```
void foo() {
int* p; // p is a pointer to an integer
int i; // i is an integer
p = &i; // Set p to point to i
*p = 13; // Change what p points to -- in this case i -- to 13
// At this point i is 13. So is *p. In fact *p is i.
}
```

p

i 13

When using a pointer to an object created with &, it is important to only use the pointer so long as the object exists. A local variable exists only as long as the function where it is declared is still executing (we'll see functions shortly). In the above example, i exists only as long as foo() is executing. Therefore any pointers which were initialized with &i are valid only as long as foo() is executing. This "lifetime" constraint of local memory is standard in many languages, and is something you need to take into account when using the & operator.

**NULL**

A pointer can be assigned the value 0 to explicitly represent that it does not currently have a pointee. Having a standard representation for "no current pointee" turns out to be very handy when using pointers. The constant NULL is defined to be 0 and is typically used when setting a pointer to NULL. Since it is just 0, a NULL pointer will behave like a boolean false when used in a boolean context. Dereferencing a NULL pointer is an error which, if you are lucky, the computer will detect at runtime -- whether the computer detects this depends on the operating system.

**Pitfall -- Uninitialized Pointers**

When using pointers, there are two entities to keep track of. The pointer and the memory it is pointing to, sometimes called the "pointee". There are three things which must be done for a pointer/pointee relationship to work...

(1) The pointer must be declared and allocated

(2) The pointee must be declared and allocated

(3) The pointer (1) must be initialized so that it points to the pointee (2)

The most common pointer related error of all time is the following: Declare and allocate the pointer (step 1). Forget step 2 and/or 3. Start using the pointer as if it has been setup to point to something. Code with this error frequently compiles fine, but the runtime results are disastrous. Unfortunately the pointer does not point anywhere good unless (2) and (3) are done, so the run time dereference operations on the pointer with * will misuse and trample memory leading to a random crash at some point.

{

int* p;

*p = 13; // NO NO NO p does not point to an int yet

// this just overwrites a random area in memory

}

-14346

p

i

Of course your code won't be so trivial, but the bug has the same basic form: declare a pointer, but forget to set it up to point to a particular pointee.

**Using Pointers**

Declaring a pointer allocates space for the pointer itself, **but it does not allocate space for the pointee**. The pointer must be set to point to something before you can dereference it.

Here's some code which doesn't do anything useful, but which does demonstrate (1) (2)

(3) for pointer use correctly...

```
int* p;                 // (1) allocate the pointer

int i;                  // (2) allocate pointee

struct fraction f1;     // (2) allocate pointee

p = &i;                     // (3) setup p to point to i

*p = 42;                    // ok to use p since it's setup

p = &(f1.numerator);        // (3) setup p to point to a different int

*p = 22;

p = &(f1.denominator);      // (3)

*p = 7;
```

So far we have just used the & operator to create pointers to simple variables such as i. Later, we'll see other ways of getting pointers with arrays and other techniques.

**C Strings**

C has minimal support of character strings. For the most part, strings operate as ordinary arrays of characters. Their maintenance is up to the programmer using the standard facilities available for arrays and pointers. C does include a standard library of functions which perform common string operations, but the programmer is responsible for the managing the string memory and calling the right functions. Unfortunately computations involving strings are very common, so becoming a good C programmer often requires becoming adept at writing code which manages strings which means managing pointers and arrays.

A C string is just an array of char with the one additional convention that a "null" character ('\0') is stored after the last real character in the array to mark the end of

the string. The compiler represents string constants in the source code such as "binky" as arrays which follow this convention. The string library functions (see the appendix for a partial list) operate on strings stored in this way. The most useful library function is strcpy(char dest[], const char source[]); which copies the bytes of one string over to another. The order of the arguments to strcpy() mimics the arguments in of '=' -- the right is assigned to the left. Another useful string function is strlen(const char string[]); which returns the number of characters in C string not counting the trailing '\0'.

Note that the regular assignment operator (=) does **not** do string copying which is why strcpy() is necessary. See Section 6, Advanced Pointers and Arrays, for more detail on how arrays and pointers work.

The following code allocates a 10 char array and uses strcpy() to copy the bytes of the string constant "binky" into that local array.

```
{
char localString[10];
strcpy(localString, "binky");
}
```

b i n k y 0 x x x x x

0 1 2 ...

localString

The memory drawing shows the local variable localString with the string "binky" copied into it. The letters take up the first 5 characters and the '\0' char marks the end of the string after the 'y'. The x's represent characters which have not been set to any particular value. If the code instead tried to store the string "I enjoy languages whichh have good string support" into localString, the code would just crash at run time since the 10 character array can contain at most a 9 character string. The large

string will be written passed the right hand side of localString, overwriting whatever was stored there.

**String Code Example**

Here's a moderately complex for loop which reverses a string stored in a local array. It demonstrates calling the standard library functions strcpy() and strlen() and demonstrates that a string really is just an array of characters with a '\0' to mark the effective end of the string. Test your C knowledge of arrays and for loops by making a drawing of the memory for this code and tracing through its execution to see how it works.

```
{
char string[1000]; // string is a local 1000 char array
int len;
strcpy(string, "binky");
len = strlen(string);
/*
Reverse the chars in the string:
i starts at the beginning and goes up
j starts at the end and goes down
i/j exchange their chars as they go until they meet
*/
int i, j;
char temp;
for (i = 0, j = len - 1; i < j; i++, j--) {
temp = string[i];
```

```
string[i] = string[j];

string[j] = temp;

}

// at this point the local string should be "yknib"

}
```

**"Large Enough" Strings**

The convention with C strings is that the owner of the string is responsible for allocating array space which is "large enough" to store whatever the string will need to store. Most routines do not check that size of the string memory they operate on, they just assume its big enough and blast away. Many, many programs contain declarations like the following...

```
{

char localString[1000];

...

}
```

The program works fine so long as the strings stored are 999 characters or shorter. Someday when the program needs to store a string which is 1000 characters or longer, then it crashes. Such array-not-quite-big-enough problems are a common source of bugs, and are also the source of so called "buffer overflow" security problems. This scheme has the additional disadvantage that most of the time when the array is storing short strings, 95% of the memory reserved is actually being wasted. A better solution allocates the string dynamically in the heap, so it has just the right size. To avoid buffer overflow attacks, production code should check the size of the data first, to make sure it fits in the destination string. See the strlcpy() function in Appendix A.

**char***

Because of the way C handles the types of arrays, the type of the variable localString above is essentially char*. C programs very often manipulate strings using variables of type char* which point to arrays of characters. Manipulating the actual chars in a string requires code which manipulates the underlying array, or the use of library functions such as strcpy() which manipulate the array for you. See Section 6 for more detail on pointers and arrays.

**TypeDef**

A typedef statement introduces a shorthand name for a type. The syntax is...

typedef <type><name>;

The following defines Fraction type to be the type (struct fraction). C is case sensitive, so fraction is different from Fraction. It's convenient to use typedef to create types with upper case names and use the lower-case version of the same word as a variable.

typedef struct fraction Fraction;

Fraction fraction; // Declare the variable "fraction" of type "Fraction"

// which is really just a synonym for "struct fraction".

The following typedef defines the name Tree as a standard pointer to a binary tree node where each node contains some data and "smaller" and "larger" subtree pointers.

typedef struct treenode* Tree;

struct treenode {

int data;

Tree smaller, larger; // equivalently, this line could say

}; // "struct treenode *smaller, *larger"

**Functions**

All languages have a construct to separate and package blocks of code. C uses the "function" to package blocks of code. This article concentrates on the syntax and peculiarities of C functions. The motivation and design for dividing a computation into separate blocks is an entire discipline in its own.

A function has a name, a list of arguments which it takes when called, and the block of code it executes when called. C functions are defined in a text file and the names of all the functions in a C program are lumped together in a single, flat namespace. The special function called "main" is where program execution begins. Some programmers like to begin their function names with Upper case, using lower case for variables and parameters, Here is a simple C function declaration. This declares a function named Twice which takes a single int argument named num. The body of the function computes the value which is twice the num argument and returns that value to the caller.

/*

Computes double of a number.

Works by tripling the number, and then subtracting to get back to double.

*/

static int Twice(int num) {

int result = num * 3;

result = result - num;

return(result);

}

**Syntax**

The keyword "static" defines that the function will only be available to callers in the file where it is declared. If a function needs to be called from another file, the function cannot be static and will require a prototype -- see prototypes below. The static form is convenient for utility functions which will only be used in the file where they are declared. Next , the "int" in the function above is the type of its return value. Next comes name of the function and its list of parameters. When referring to a function by name in documentation or other prose, it's a convention to keep the parenthesis () suffix, so in this case I refer to the function as "Twice()". The parameters are listed with their types and names, just like variables. Inside the function, the parameter num and the local variable result are "local" to the function -- they get their own memory and exist only so long as the function is executing. This independence of "local" memory is a standard feature of most languages.

The "caller" code which calls Twice() looks like...

```
int num = 13;

int a = 1;

int b = 2;

a = Twice(a); // call Twice() passing the value of a

b = Twice(b + num); // call Twice() passing the value b+num

// a == 2

// b == 30

// num == 13 (this num is totally independent of the "num" local to Twice()
```

**Call by Value vs. Call by Reference**

C passes parameters "by value" which means that the actual parameter values are copied into local storage. The caller and callee functions do not share any memory --

they each have their own copy. This scheme is fine for many purposes, but it has twodisadvantages.

1) Because the callee has its own copy, modifications to that memory are not communicated back to the caller. Therefore, value parameters do not allow the callee to communicate back to the caller. The function's return value can communicate some information back to the caller, but not all problems can be solved with the single return value.

2) Sometimes it is undesirable to copy the value from the caller to the callee because the value is large and so copying it is expensive, or because at a conceptual level copying the value is undesirable.

The alternative is to pass the arguments "by reference". Instead of passing a copy of a value from the caller to the callee, pass a pointer to the value. In this way there is only one copy of the value at any time, and the caller and callee both access that one value through pointers.

Some languages support reference parameters automatically. C does not do this – the programmer must implement reference parameters manually using the existing pointer constructs in the language.

**Swap Example**

The classic example of wanting to modify the caller's memory is a swap() function which exchanges two values. Because C uses call by value, the following version of Swap will not work...

```
void Swap(int x, int y) { // NO does not work

int temp;

temp = x;

x = y; // these operations just change the local x,y,temp
```

```
y = temp; // -- nothing connects them back to the caller's a,b
}
// Some caller code which calls Swap()...
int a = 1;
int b = 2;
Swap(a, b);
```

Swap() does not affect the arguments a and b in the caller. The function above only operates on the copies of a and b local to Swap() itself. This is a good example of how "local" memory such as ( x, y, temp) behaves -- it exists independent of everything else only while its owning function is running. When the owning function exits, its local memory disappears.

**Reference Parameter Technique**

To pass an object X as a reference parameter, the programmer must pass a pointer to X instead of X itself. The formal parameter will be a pointer to the value of interest. The caller will need to use & or other operators to compute the correct pointer actual parameter. The callee will need to dereference the pointer with * where appropriate to access the value of interest. Here is an example of a correct Swap() function.

```
static void Swap(int* x, int* y) { // params are int* instead of int
int temp;
temp = *x; // use * to follow the pointer back to the caller's memory
*x = *y;
*y = temp;
}

// Some caller code which calls Swap()...
```

```
int a = 1;

int b = 2;

Swap(&a, &b);
```

Things to notice...

• The formal parameters are int* instead of int.

• The caller uses & to compute pointers to its local memory (a,b).

• The callee uses * to dereference the formal parameter pointers back to get the caller's memory.

Since the operator & produces the address of a variable -- &a is a pointer to a. In Swap() itself, the formal parameters are declared to be pointers, and the values of interest (a,b) are accessed through them. There is no special relationship between the **names** used for the actual and formal parameters. The function call matches up the actual and formal parameters by their order -- the first actual parameter is assigned to the first formal parameter, and so on. I deliberately used different names (a,b vs x,y) to emphasize that the names do not matter.

**const**

The qualifier const can be added to the left of a variable or parameter type to declare that the code using the variable will not change the variable. As a practical matter, use of const is very sporadic in the C programming community. It does have one very handy use, which is to clarify the role of a parameter in a function prototype...

```
void foo(const struct fraction* fract);
```

In the foo() prototype, the const declares that foo() does not intend to change the struct fraction pointee which is passed to it. Since the fraction is passed by pointer, we could not know otherwise if foo() intended to change our memory or not. Using the const, foo() makes its intentions clear. Declaring this extra bit of information helps to clarify the role of the function to its implementor and caller.

**Bigger Pointer Example**

The following code is a large example of using reference parameters. There are several common features of C programs in this example...Reference parameters are used to allow the functions Swap() and IncrementAndSwap() to affect the memory of their callers.

There's a tricky case inside of IncrementAndSwap() where it calls Swap() -- no additional use of & is necessary in this case since the parameters x, y inside InrementAndSwap() are already pointers to the values of interest. The names of the variables through the program(a, b, x, y, alice, bob) do not need to match up in any particular way for the parameters to work. The parameter mechanism only depends on the types of the parameters and their order in the parameter list -- not their names. Finally this is an example of what multiple functions look like in a file and how they are called from the main() function.

```
static void Swap(int* a, int* b) {

int temp;

temp = *a;

*a = *b;

*b = temp;

}

static void IncrementAndSwap(int* x, int* y) {

(*x)++;

(*y)++;

Swap(x, y); // don't need & here since a and b are already

// int*'s.

}

int main()

{
```

```
int alice = 10;

int bob = 20;

Swap(&alice, &bob);

// at this point alice==20 and bob==10

IncrementAndSwap(&alice, &bob);

// at this point alice==11 and bob==21

return 0;

}
```

**Standard Library Functions**

Many basic housekeeping funcions are available to a C program in form of standard library functions. To call these, a program must #include the appropriate .h file. Most compilers link in the standard library code by default. The functions listed in the next section are the most commonly used ones, but there are many more which are not listed here.

| | |
|---|---|
| stdio.h | file input and output |
| ctype.h | character tests |
| string.h | string operations |
| math.h | mathematical functions such as sin() and cos() |
| stdlib.h | utility functions such as malloc() and rand() |
| assert.h | the assert() debugging macro |
| stdarg.h | support for functions with variable numbers of arguments |
| setjmp.h | support for non-local flow control jumps |
| signal.h | support for exceptional condition signals |

time.h date and time

limits.h, float.h constants which define type range values such as INT_MAX

****** THANK YOU ******