UNIT-1

Operating Systems Overview:

- ✓ Operating system functions
- ✓ Operating system structure
- ✓ Operating systems operations
- ✓ Computing environments
- ✓ Open-Source Operating Systems.

System Structures:

- ✓ Operating System Services
- ✓ User and Operating-System Interface
- ✓ Systems calls
- ✓ Types of System Calls
- ✓ System programs
- ✓ Operating system Design and Implementation
- ✓ Operating system structure
- ✓ Operating system debugging
- ✓ Building and Booting an Operating System

- **Hardware** refers to the physical and visible components of the system such as a monitor, CPU, keyboard and mouse.
- Hardware is further divided into four main categories:
    - Input Devices
    - Output Devices
    - Secondary Storage Devices
    - Internal Components

Software

- Software is a set of instructions, data or programs used to operate computers and execute specific tasks.
- Computer software is divided into two broad categories:
    - System Software
    - Application Software

System Software

- System software consists of programs that manage the hardware resources of a computer and perform required information processing tasks.
- The best example for system software is OS
- **Operating system**:
    - Examples of popular modern operating systems include Android, iOS, Linux, Mac OS X and Microsoft Windows
- **Utility programs:**
    - **Virus scanner** - to protect your system from trojans and viruses
    - **Disk defragmenter** - to speed up your hard disk
    - **System monitor** - to look at your current system resources
    - **File managers** - to add, delete, rename and move files and folders
- **Library programs:**
- Library programs contain code and data that provide services to other programs such as interface (look and feel), printing, network code and even the graphic engines of computer games.
- **Translator software:**
    - **Assembler** - converts assembly code into machine code
    - **Interpreter** - converts 3rd generation languages such as javascript into machine code one line at a time

- **Compiler** - converts 3rd generation languages such as C++ into machine code all at once
- **Application Software:**
  - Application software is used by user to perform specific task.
  - Some examples of application softwares are word processor, web browser, media player, etc.

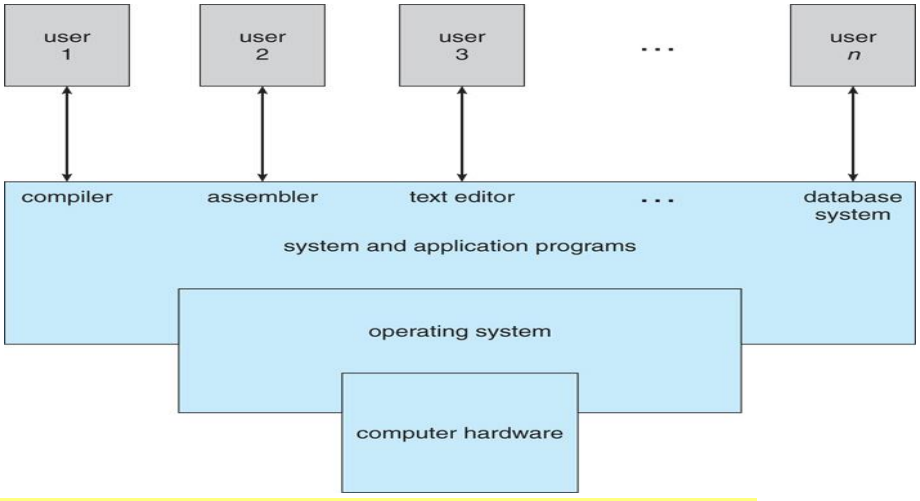
Differences between system software and application software

- The **system software** is designed to manage system resources such as memory management, process management, security, etc. and also provides a platform to run application software. On the other hand, **application software** is designed to meet user requirements to perform specific tasks.
- The **system software** is written in a low-level language, like assembly language. Whereas, **application software** is written in high-level languages like Java, C ++, .net, VB, etc.
- As soon as the **system software** is started, the system starts and continues until the system is shut down. **Application software** starts when the user starts it and the user closes it.
- A system cannot run without system software, while the **application software** is user-specific. They are not required to run the system, they are only for the user.
- Where system software is general-purpose software, while **application software** is a specific purpose software.
- The best example of system software is the operating system, while examples of application software are Microsoft office, photo shop, etc.

What is an Operating System?

- An Operating System (OS) is program that manages the computer hardware.
- It also provides a basis for Application Programs and acts as a intermediary between a user of a computer and the computer hardware.
- An operating system controls the overall activities of a computer.

- An Operating System (OS) is program that manages the computer hardware.
- It also provide a basis for Application Programs and acts as a intermediary between a user of a computer and the computer hardware



Abstract view of the components of a computer system

1.1 Operating system functions

- **The main functions of an operating system are:**

✓ **Device Management (Input / Output):-** An operating system controls the working of all input and output devices.

✓ **Memory Management:-** An operating system assigns memory to various programs whenever required. It also frees the memory when it is not in use.

✓ **Runs Software:-** An operating system runs the applications software like Paint, MS Word, and MS Power Point etc.

✓ **Processor Management:-** An operating system manages the working of the processor by allocating various jobs to it.

✓ **File Management:-** An operating system keeps track of information regarding creating, deletion, transfers, copy and storage of files in an organized way.

✓ **Security:-** It provides security by means of passwords to prevent misuse of a computer.

What Operating Systems Do

- To understand more fully the operating system's role, we next explore operating systems from two viewpoints:
    - user
    - System.

- **The user's view**
    - The user's view of the computer varies according to the interface being used.
- **Personal Computer:**
    - PC, consisting of a monitor, keyboard, mouse, and system unit.
    - Such a system is designed for one user to monopolize its resources.
    - In this case, the operating system is designed mostly for **ease of use,** with some attention paid to **performance** and **none paid** to **resource utilization**
- **Workstations And Servers:**
    - In still other cases, users sit at **workstations** connected to **networks of** other workstations and **servers.**
    - These users have dedicated resources **at** their disposal, but they also share resources such as networking and servers, including file, compute, and print servers. Therefore, their operating system is designed to compromise between individual usability and resource utilization.
- **Mobile Computers:**
    - Many varieties of mobile computers, such as smart phones and tablets, have come into fashion. Most mobile computers are standalone units for individual users. Quite often, they are connected to networks through cellular or other wireless technologies. The user interface for mobile computers generally features a **touch screen, where the user interacts with the** system by pressing and swiping fingers across the screen rather than using a physical keyboard and mouse.
- **Mainframe Or A Minicomputer:**
    - A user sits at a terminal connected to a **mainframe or a minicomputer.** Other users are accessing the same computer through other terminals.
    - These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization— to assure that all available CPU time, memory, and I/O are used efficiently and that no individual user takes more than her fair share.
    - Some computers have little or no user view.
    - For example, **embedded computers** in home devices and automobiles may have numeric keypads and may turn indicator lights on or off to show status, but they and their operating systems are designed primarily to run without user intervention.

System View

- **OS is a resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- **OS is a control program**
  - Controls execution of programs to prevent errors and improper use of the computer

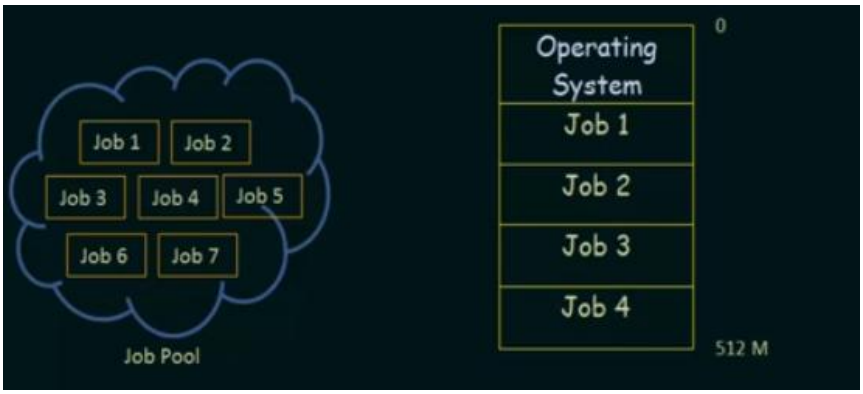
1.2 Operating system structure

- Operating System vary greatly in their makeup internally.

  COMMONALITIES:
  - 1.Multiprogramming
  - 2.Time Sharing (Multitasking)

1. Multiprogramming

- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming increases CPU utilization by organizes jobs (code and data) so that the CPU always has one to execute
- Multiprogrammed systems provide an environment in which the various system resources(for example, CPU, memory, and peripheral devices) are utilized effectively, but they don't provide for user interaction with the computer system



- **Advantages**
  - 1)CPU utilization maximum
  - 2)Short time jobs get executed faster
  - 3)Resources are used efficiently
  - 4)Response time is shorter
- **Disadvantages**
  - 1)Difficult to program system because of complicated schedule handling
  - 2)Due to high load of tasks ,long time jobs have to wait long

2.Time Sharing (Multitasking)

- Time sharing (or multitasking) is a logical extension of multiprogramming.
- CPU executes multiple jobs by switching among them
- Switches occur so frequently that the users can interact with each program while it is running
- Time sharing requires an interactive (or hands-on) computer system, which provides direct communication between the user and the system.
- A time-shared operating system allows many users to share the computer simultaneously
- A time-shared operating system allows many users to share the computer
- Uses CPU scheduling and Multiprogramming to provide With a small portion of a time-shared computer
- Each User has at least one separate program in memory
- A program loaded into memory and executing is called a **"PROCESS"**

Terminology

- Each user has at least one program executing in memory ⇨process
- If several jobs ready to run at the same time ⇨ CPU scheduling
- If processes don't fit in memory, swapping moves them in and out to run
- Virtual memory allows execution of processes not completely in memory

1.3 Operating systems operations

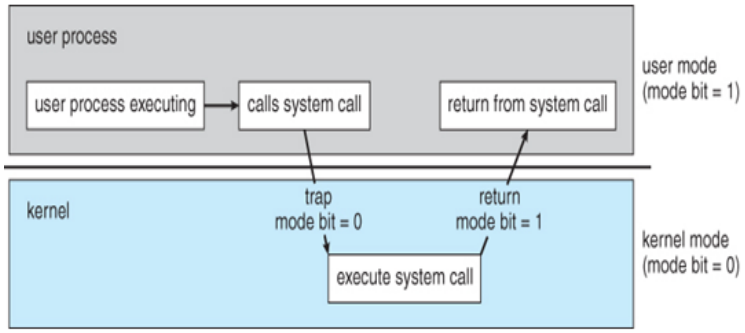
- **Interrupt driven** is modern operating systems.
- **Trap (or** an **exception)** is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed.
- The interrupt-driven nature of an operating system defines that system's general structure.
- For each type of interrupt, separate segments of code in the operating system determine what action should be taken.
- An interrupt service routine is provided that is responsible for dealing with the interrupt.
- Since the operating system and the users share the hardware and software resources of the computer system, we need to make sure that an error in a user program could cause problems only for the one program that was running.

Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.

  1. User mode – execution done on behalf of a user.
  2. **Monitor mode** (also **kernel mode** or **system mode**) – execution done on behalf of operating system.

  A bit, called the **mode bit,** is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).
- The dual mode of operation provides us with the means for protecting the operating system from errant users—and errant users from one another.



Transition from user to Kernel mode

- We accomplish this protection by designating some of the machine instructions that may cause harm as **privileged instructions.** The hardware allows privileged instructions to be executed only in kernel mode.
- If an attempt is made to execute a privileged instruction in user mode, the hardware does not execute the instruction but rather treats it as illegal and traps it to the operating system.

Timer

- A **timer** can be set to interrupt the computer after a specified period.

- The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second).
- **Variable timer** is generally implemented by a fixed-rate clock and a counter. The operating system sets the counter.
- Every time the clock ticks, the counter is decremented.
- We can use the timer to prevent a user program from running too long. A simple technique is to initialize a counter with the amount of time that a program is allowed to run.

1.4 Computing environments

- **Traditional Computing:**
  - ✓ Stand-alone general purpose machines
  - ✓ But blurred as most systems interconnect with others (i.e., the Internet)
  - ✓ Portals provide web access to internal systems
  - ✓ Network computers (thin clients) are like Web terminals
  - ✓ Mobile computers interconnect via wireless networks
  - ✓ Networking becoming ubiquitous
  - ✓ even home systems use firewalls to protect home computers from Internet attacks
- **Mobile Computing**
  - ✓ Handheld smartphones, tablets, etc
  - ✓ What is the functional difference between them and a "traditional" laptop?
  - ✓ These devices share the distinguishing physical features of being portable and lightweight.
  - ✓ Allows new types of apps like augmented reality
  - ✓ Use IEEE 802.11 wireless, or cellular data networks for connectivity
  - ✓ Two operating systems currently dominate mobile computing:
    - Apple iOS
    - Google Android
- **Distributed Systems**
  - ✓ Distributed computing is a field of computer science that studies distributed systems.
  - ✓ A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages.
  - Two modes:
    - **FTP – File transfer protocol** – is a standard network protocol used for the transfer of computer files between a client and server on a computer network.
      - FileZilla Client, **FTP** Voyager, WinSCP, CoffeeCup Free **FTP**, and Core **FTP**
    - **NFS – Network File System** - is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed

Network- is a communication path between two or more systems.

TCP/IP is the most common network protocol, and it provides the fundamental architecture of the Internet. Most operating systems support TCP/IP, including all general-purpose ones. Some systems support proprietary protocols to suit their needs.

- Types of Network
  - **Local-Area Network (LAN)** connects computers within a room, a building, or a campus.
  - **Wide-Area Network (WAN)** usually links buildings, cities, or countries. A global company may have a WAN to connect its offices worldwide.
  - **Metropolitan-Area Network (MAN)** – links buildings within the city.
  - **Personal-Area Network (PAN)** between a phone and a headset or a smartphone and a desktop computer.

- **Network Operating System** is an operating system that provides features such as file sharing across the network, along with a communication scheme that allows different processes on different computers to exchange messages.

Client-Server Computing

- ✓ **Compute-server system** provides an interface to which a client can send a request to perform an action (for example, read data).
- ✓ **File-server system** provides a file-system interface where clients can create, update, read, and delete files.

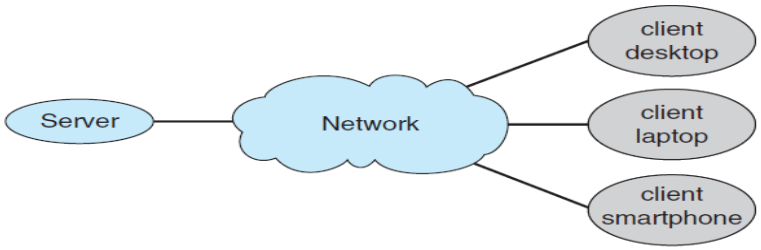


Figure 1.18 General structure of a client–server system.

Peer – to – Peer Computing

- Clients and servers are not distinguished from one another. Instead, all nodes within the system are considered peers, and each may act as either a client or a server, depending on whether it is requesting or providing a service.

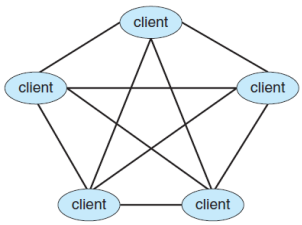


Figure 1.19 Peer-to-peer system with no centralized service.

- Skype is another example of peer-to-peer computing. It allows clients to make voice calls and video calls and to send text messages over the Internet using a technology known as voice over IP (VoIP).

Virtualization

- Virtualization is a technology that allows operating systems to run as applications within other operating systems.
- Emulation is used when the source CPU type is different from the target CPU type.
  - which allowed applications compiled for the IBM CPU to run on the Intel CPU.

Cloud Computing

- Cloud Computing is a type of computing that delivers computing, storage, and even applications as a service across a network.
- **Types of Cloud Computing**
  - **Software as a service (SaaS)—**one or more applications (such as word processors or spreadsheets) available via the Internet.
  - Example
    - Google Slides
    - Google Docs
    - Google Sheets
    - Calendar

- **Platform as a service (PaaS)—**often simply referred to as PaaS, is a category of cloud computing that provides a platform and environment to allow developers to build applications and services over the internet.
- **Infrastructure as a service (IaaS)—**servers or storage available over the Internet (for example, storage available for making backup copies of production data).
- **Examples:**Dropbox,google drive

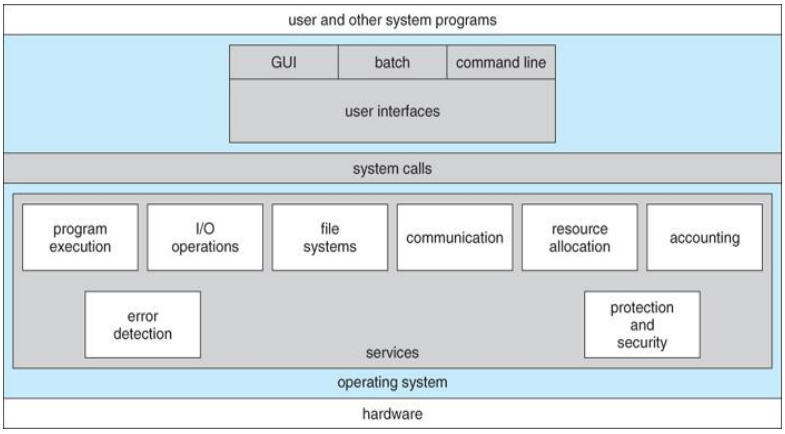
Real-Time Embedded Systems

- An embedded system is a **computer system** with a **dedicated** function within a larger **mechanical** or **electrical system**, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts.
- Embedded systems **control** many devices in common use today.
- Examples of **Real-Time Embedded System**
  - Alarm Systems
  - Washing machine
  - Vendo Machine
- Devices in Embedded System
  - Arduino
  - Raspberry PI

1.5 Open-Source Operating Systems.

- Open source is a term that originally referred **to open source software (OSS)**.
- Open source software is code that is designed to be publicly accessible—anyone can see, modify, and distribute the code as they see fit.
- Some widely used programs, platforms, and languages which are considered open source are:
  - Linux operating system
  - Android by Google

1.6 Operating System Services

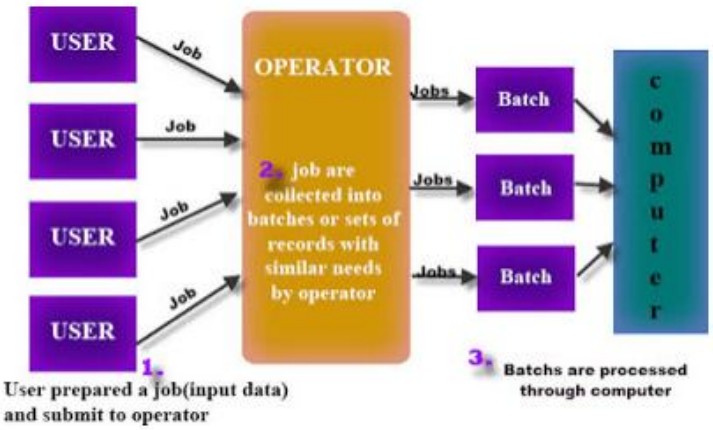


A view of operating system services

- OS provide environments in which programs run, and services for the users of the system, including:
  - User Interfaces
  - Program Execution
  - I/O Operations
  - File-System Manipulation
  - Communications
  - Error Detection
- **User Interfaces –**

- Means by which users can issue commands to the system. Depending on the system these may be a

- **Command-Line Interface**
  - (e.g. sh, csh, ksh, tcsh, etc.),
- **GUI interface**
  - (e.g. Windows, X-Windows, KDE, Gnome, etc.),
- The latter are generally older systems using punch cards of job-control language, JCL, but may still be used today for specialty systems designed for a single purpose.
- **Batch operating system**
- "The operating system is termed as "batch operating" because the input data (job) are collected into batches or sets of records with similar needs and each batch is processed as a unit(group). The output is another batch that can be reused for computation."



- **Program Execution** - The OS must be able to load a program into RAM, run the program, and terminate the program, either normally or abnormally.
- **I/O Operations -** The OS is responsible for transferring data to and from I/O devices, including keyboards, terminals, printers, and storage devices.
- **Communications -** Inter-process communications, IPC, either between processes running on the same processor, or between processes running on separate processors or separate machines. May be implemented as either shared memory or message passing, (or some systems may offer both)
- **File-System Manipulation** - the OS is also responsible for maintaining directory and subdirectory structures, mapping file names to specific blocks of data storage, and providing tools for navigating and utilizing the file system.
- **Error Detection** - Both hardware and software errors must be detected and handled appropriately, with a minimum of harmful repercussions.
- Some systems may include complex error avoidance or recovery systems, including backups, RAID drives, and other redundant systems. Debugging and diagnostic tools aid users and administrators in tracing down the cause of problems.
- Other systems aid in the efficient operation of the OS itself:
- **Resource Allocation** - E.g. CPU cycles, main memory, storage space, and peripheral devices. Some resources are managed with generic systems and others with very carefully designed and specially tuned systems, customized for a particular resource and operating environment.
- **Accounting** - Keeping track of system activity and resource usage, either for billing purposes or for statistical record keeping that can be used to optimize future performance.
- **Protection and Security** - Preventing harm to the system and to resources, either through wayward internal processes or malicious outsiders. Authentication, ownership, and restricted access are obvious parts of this system. Highly secure systems may log all process activity down to excruciating detail, and security regulation dictate the storage of those records on permanent non-erasable medium for extended times in secure (off-site) facilities.

1.7 User and Operating-System Interface

- The **user interface** (UI) is the point of human-computer interaction and communication in a device.

(OR)

- A **user interface** (**UI**) is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end
- There are several ways for users to interface with the operating system.
- **Two fundamental approaches.**
    - 1.command-line interface, or **command interpreter,**
    - 2. Graphical User Interface

Command Interpreters

- ✓ Command Interpreter that allows users to directly enter commands to be performed by the operating system.
- ✓ Some operating systems include the **command interpreter** in the **kernel**.
- ✓ Others such as Windows and UNIX, treat the **command interpreter** as a **special program** that is running when a job is initiated or when a user first logs on (on interactive systems).
- ✓ On systems with multiple command interpreters to choose from, the interpreters are known as **shells.**
    - **For example, on UNIX and Linux systems, a** user may choose among several different shells, including the ***Bourne shell, C shell, Bourne-Again shell, Korn shell, and others***

Advantages of using a command-line interface are:

- ✓ It is faster than the other types of user interfaces.
- ✓ Fast data transfer in CLI.
- ✓ Memory (RAM) is used lesser than GUI.
- ✓ It is cheaper to use as a lesser resolution screen can be used.
- ✓ Lesser CPU processing time is needed.
- ✓ It does not need Windows to run.

Disadvantages of using a command-line interface are

- ✓ Without the known commands, you cannot work on the CLI
- ✓ A lot of commands have to be learned to use this interface.
- ✓ One needs to be very specific and careful when typing the commands. Even a single spelling mistake may lead to instruction failure.
- ✓ There are no images and graphics on the screen.

Graphical User Interfaces

- A GUI (Graphic User Interface) is a graphical representation in which the users can interact with software or devices through graphical icons.
- A second strategy for interfacing with the operating system is through a user friendly graphical user interface, or GUI.
- Here, rather than entering commands directly via a command-line interface, users employ a mouse-based window and- menu system characterized by a **desktop metaphor.**
- **The user moves the** mouse to position its pointer on images, or **icons, on the screen (the desktop)** that represent programs, files, directories, and system functions.

History of GUI

- The first GUI appeared on the Xerox Alto computer in 1973. However, graphical interfaces became more widespread with the advent of **Apple Macintosh** computers in the 1980s.
- The user interface for the Macintosh operating system (Mac OS) has undergone various changes over the years, the most significant being the adoption of the ***Aqua interface that appeared with Mac OS X.***
- ***Microsoft's first version of*** Windows—Version 1.0—was based on the addition of a GUI interface to the MS-DOS operating system.

Advantages of a graphic user interface are:

- ✓ This type of user interface is easy to use.
- ✓ It is easy to explore and find your way around the system using a GUI interface
- ✓ You do not have to learn complicated commands.
- ✓ It is based on a graphical representation
- ✓ Users do not need to know any programming languages.
- ✓ It is also known as a multi-user operating system.

Disadvantages of a graphic user interface are:

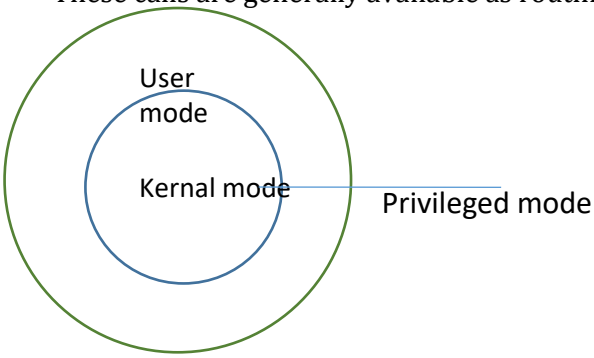
- ✓ GUIs take up a much larger amount of hard disk space than other interfaces.
- ✓ They use more processing power than other types of interface.
- ✓ They need significantly more memory (RAM) to run than other interface types. They let you exchange data between different software applications

Choice of Interface

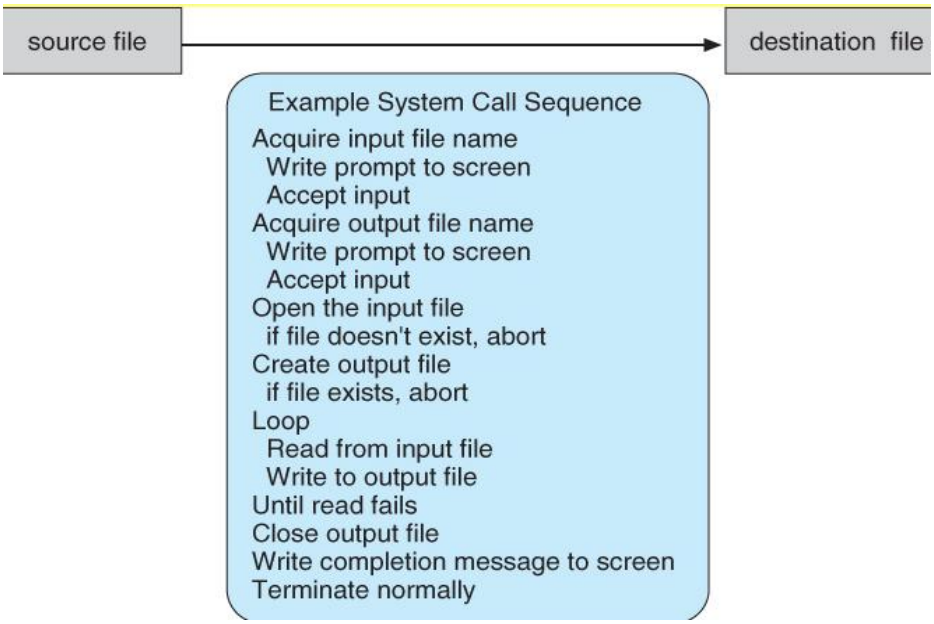
- One of personal preference.
  - System administrators who manage computers and power users who have deep knowledge of a system frequently use the command-line interface.
- For them, it is more efficient, giving them faster access to the activities they need to perform.
- Most Windows users are happy to use the Windows GUI environment and almost never use the MS-DOS shell interface.
- The various changes undergone by the Macintosh operating systems provide a nice study in contrast.

1.8 Systems calls

- "System calls provide the interface between a process (User program or application software) and the operating system.
- In other word System call is how a program requests a service from an operating system's kernel.
- These calls are generally available as routines written in C or C++



Example of a System call sequence for writing a simple program to read data from one file and copy them to another file



illustrates the sequence of system calls required to copy a file:

- Most programmers do not use the low-level system calls directly, but instead use an "Application Programming Interface", API. The following sidebar shows the read( ) call available in the API on UNIX based systems::

EXAMPLE OF STANDARD API

As an example of a standard API, consider the read( ) function that is available in UNIX and Linux systems. The API for this function is obtained from the man page by invoking the command

man read

on the command line. A description of this API appears below:

```
#include <unistd.h>

ssize_t     read(int fd, void *buf, size_t count)
```

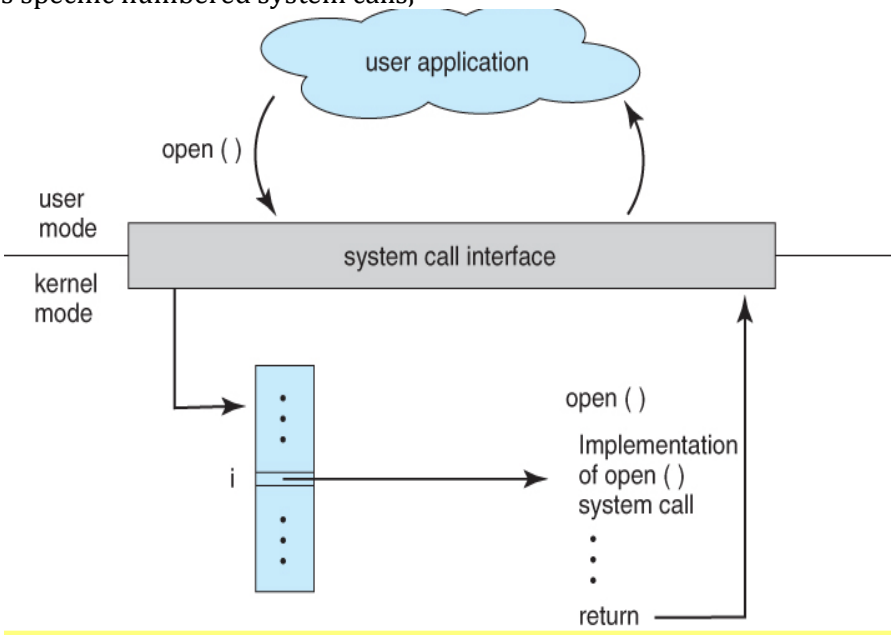
return value — function name — parameters

A program that uses the read( ) function must include the unistd.h header file, as this file defines the ssize_t and size_t data types (among other things.) The parameters passed to read( ) are as follows:

- int fd - The file descriptor to be read
- void *buf - A buffer where the data will be read into
- size_t count - The maximum number of bytes to be read into the buffer.

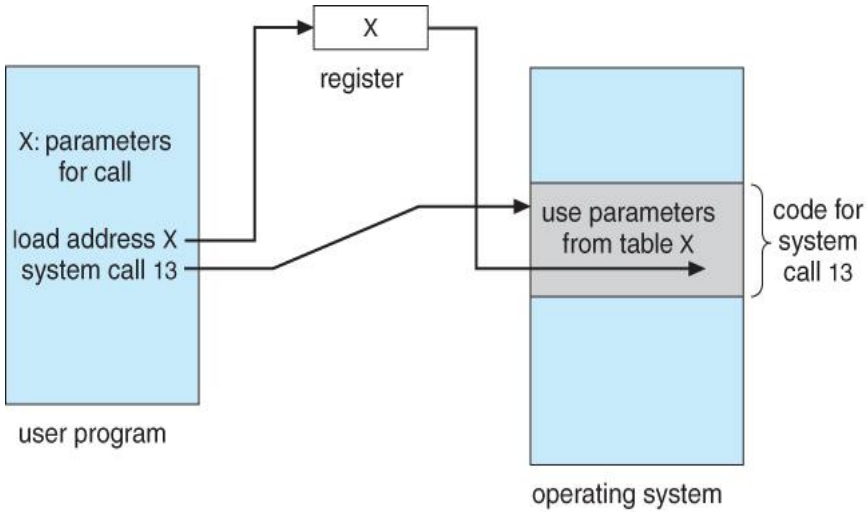
On a successful read, the number ofbytes read is returned. A return value of 0 indicates end of file. If an error occurs, read( ) returns -1.

- The use of APIs instead of direct system calls provides for greater program portability between different systems. The API then makes the appropriate system calls through the **system call interface**, using a table lookup to access specific numbered system calls,



The handling of a user application invoking the open( ) system call

- Parameters are generally passed to system calls via registers, or less commonly, by values pushed onto the stack. Large blocks of data are generally accessed indirectly, through a memory address passed in a register or on the stack



Passing of parameters as a table

1.9 Types of System Calls

- Six major categories,
  - **Process Control**
  - **File Management**
  - **Device Management**
  - **Information Maintenance**
  - **Communication**
  - **Protection**
- **Process Control**
  - A running program needs to be able to stop execution either normally or abnormally. When execution is stopped abnormally, often a dump of memory is taken and can be examined with a debugger.

- Process control
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory

- **File Management**
  - ✓ File management system calls include create file, delete file, open, close, read, write, reposition, get file attributes, and set file attributes.
  - ✓ These operations may also be supported for directories as well as ordinary files.

- File management
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes

- **Device Management**

✓ Device management system calls include request device, release device, read, write, reposition, get/set device attributes, and logically attach or detach devices.

- ✓ Devices may be physical (e.g. disk drives), or virtual / abstract (e.g. files, partitions, and RAM disks).
  - ✓ Some systems represent devices as special files in the file system, so that accessing the "file" calls upon the appropriate device drivers in the OS.

- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices

- **Information Maintenance**

✓ Information maintenance system calls include calls to get/set the time, date, system data, and process, file, or device attributes.

✓ Systems may also provide the ability to dump memory at any time, single step programs pausing execution after each instruction, and tracing the operation of programs, all of which can help to debug programs.

- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes

- **Communication**

✓ Communication system calls create/delete communication connection, send/receive messages, transfer status information, and attach/detach remote devices.

✓ The **message passing** model must support calls to:
  - Identify a remote process and/or host with which to communicate.
  - Establish a connection between the two processes.
  - Open and close the connection as needed.
  - Transmit messages along the connection.
  - Wait for incoming messages, in either a blocking or non-blocking state.
  - Delete the connection when no longer needed.

✓ The **shared memory** model must support calls to:
  - Create and access memory that is shared amongst processes (and threads.)
  - Provide locking mechanisms restricting simultaneous access.
  - Free up shared memory and/or dynamically allocate it as needed.

✓ Message passing is simpler and easier, (particularly for inter-computer communications), and is generally appropriate for small amounts of data.

✓ Shared memory is faster, and is generally the better approach where large amounts of data are to be shared, (particularly when most processes are reading the data rather than writing it, or at least when only one or a small number of processes need to change any given data item.)

- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices

Figure 2.8 Types of system calls.

- **Protection**

✓ Protection provides mechanisms for controlling which users / processes have access to which system resources.

✓ System calls allow the access mechanisms to be adjusted as needed, and for non-privileged users to be granted elevated access permissions under carefully controlled temporary circumstances.

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

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

1.10 System programs

- ✓ An important aspect of a modern system is the collection of system programs
- ✓ System programs provide a convenient environment for program development and execution
- **File management** -
  - programs to create,
  - delete,
  - copy,
  - rename,
  - print,
  - list, and generally manipulate files and directories.
- **Status information** -
  - Utilities to check on the date,
  - time,
  - number of users,
  - processes running,
  - data logging, etc.
  - System **registries** are used to store and recall configuration information for particular applications.
- **File modification** -
  - e.g. text editors and other tools which can change file contents.
- **Programming-language support** -
  - E.g. Compilers,
  - linkers,
  - debuggers,
  - profilers,
  - assemblers,
  - library archive management,
  - interpreters for common languages, and support for make.
- **Background services** - System daemons are commonly started when the system is booted, and run for as long as the system is running, handling necessary services.
  - Examples include network daemons, print servers, process schedulers, and system error monitoring services.

1.11 Operating system Design and Implementation

Design Goals

The first problem in designing a system is to define goals and specifications. At the highest level, the design of the system will be affected by the choice of hardware and the type of system: batch, time sharing, single user, multiuser, distributed, real time, or general purpose.

Beyond this highest design level, the requirements may be much harder to specify. The requirements can, however, be divided into two basic groups: user goals and system goals.

- User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
- System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

- Specifying and designing an OS is highly creative task of **software engineering**

Mechanisms and Policies

One important principle is the separation of policy from mechanism. Mechanisms determine *how* to do something; policies determine *what* will be done.

For example, the timer construct is a mechanism for ensuring CPU protection, but deciding how long the timer is to be set for a particular user is a policy decision.

- **Policy: What** needs to be done?
  - Example: Interrupt after every 100 seconds
- **Mechanism: How** to do something?
  - Example: timer
- Important principle: separate policy from mechanism
- The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later.
  - Example: change 100 to 200

Implementation

- Much variation
  - Early OSes in assembly language
  - Then system programming languages like Algol, PL/1
  - Now C, C++
- Actually usually a mix of languages
  - Lowest levels in assembly
  - Main body in C
  - Systems programs in C, C++, scripting languages like PERL, Python, shell scripts
- More high-level language easier to **port** to other hardware
  - But slower
- **Emulation** can allow an OS to run on non-native hardware

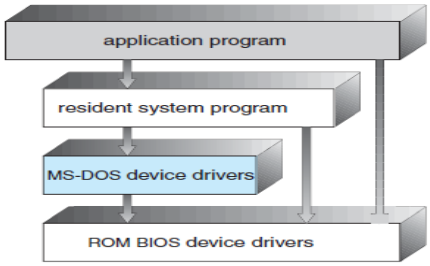
1.12 Operating system structure

- An operating system structure (Also referred as **OS Architecture**) is a term that specifies the overall internal structure of the operating system.
- It includes the logical components and the logical interrelationships of various components and the functions of these structural components that constitute an operating system
- Depending upon the internal structure , the Operating Systems can be grouped into of four major types :
  - Simple Structure OS
  - Layered Approach OS
  - Microkernel Approach OS
  - Module based OS

Simple Structure OS

- The simple structure OS has only four layers although it is not a layered OS .
- The MS DOS (Disk Operating System) was based on the simple structure .

- However , the simple structure OS had some major limitations
- The Simple Structure OS did not have a well defined structure which made these OS vulnerable for system crash.



- **Figure 2.11** MS-DOS layer structure.

Monolithic Kernel Operating Systems

Kernel:

- The OS consist of number of components and the core software components of an operating system are collectively called as the kernel.
- The kernel components include system software components that manages file system , Inter process communication , Input and output , Device Management and process Management.
- The kernel has unrestricted access and also responsible to manage all of the resources of the computer system.
- In monolithic architecture operating systems , each layer of the operating system communicates only with the layer immediately placed above and below it .
- In this architecture , the lower-level layers provide services to higher-level ones using an interface that hides their implementation.

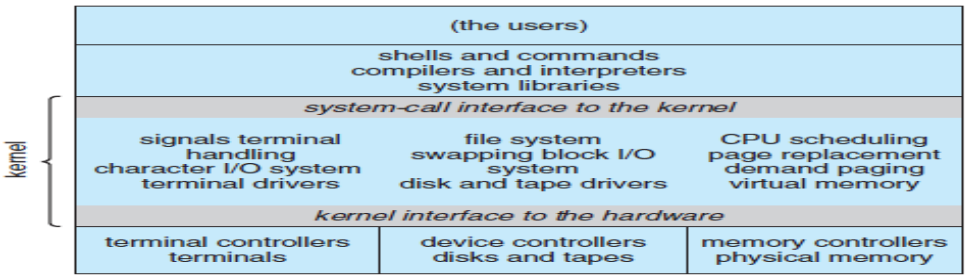
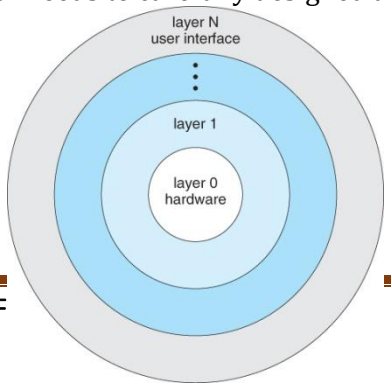


Figure 2.12 Traditional UNIX system structure.

- And therefore , the Applications had access and could communicate directly with any internal component of the operating system which gave unrestricted system access .
- This kind of layered design was perceived to be very efficient but at the same time more vulnerable for system crash due to unrestricted system access.

Layered Approach OS

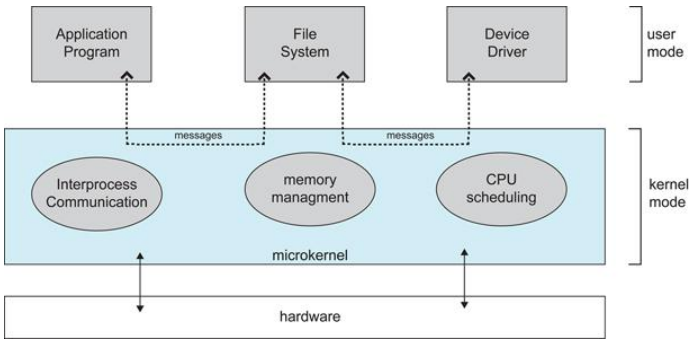
- The lack of well defined layers and the modules was problem area for simple structure OS . To overcome these limitations, the next generation of operating systems adopted a layered Architecture approach of operating system developed in 1960's.
- In layered approach based OS , the operating system components are grouped in layers based on the functionality of these components . The bottom layer (layer 0) is the hardware layer and the topmost layer (layer n).
- In layered approach OS architecture , each top layer can use immediate bottom layer . And therefore , the functionality of the each layer needs to carefully designed and implemented in each layer



- In case of layered OS structure , the OS is divided into number of layers depending upon the functionality of layers .
- It is relatively much easier to implement the layered architecture and that is the main advantage.
- On the other hand , the layered structure OS are relatively less efficient .
- The MS Windows NT which is a network OS and OS2 were based on the layered approach OS .

Microkernel Approach OS

- In case of micro-kernel architecture , the kernel components includes only a very small number of bare minimum essential services within the kernel space in order to keep the kernel design small and scalable.
- The services within the micro-kernel area typically include low-level memory management, inter-process communication and basic process synchronization to enable processes to cooperate.
- This kind of micro-kernel design was observed to be more stable and at the same time less vulnerable for system crash caused due to unrestricted system access.



- In micro-kernel architecture design based operating systems , only critical components are placed inside kernel area .
- And other operating system components such as process management and device management are placed outside kernel area which execute outside the kernel within a user area with lower level of system access.

Module based OS

- The best current methodology for operating-system design involves using **loadable kernel modules.**
- **Here, the kernel has a set of core components** and links in additional services via modules, either at boot time or during run time.
- This type of design is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X, as well as Windows.

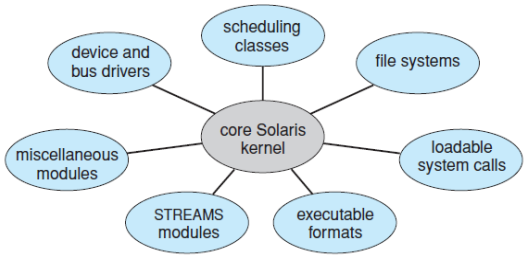


Figure 2.15 Solaris loadable modules.

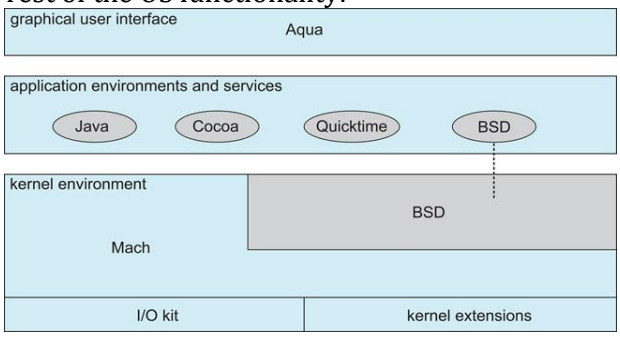
- A layered system in that each kernel section has defined, protected interfaces; but it is more flexible than a layered system, because any module can call any other module.
- The approach is also similar to the microkernel approach in that the primary module has only core functions and knowledge of how to load and communicate with other modules; but it is more efficient, because modules do not need to invoke message passing in order to communicate

Hybrid Systems

- Most OSes today do not strictly adhere to one architecture, but are hybrids of several.
  - **Mac OS X**
  - **iOS**
  - **Android**

Mac OS X

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



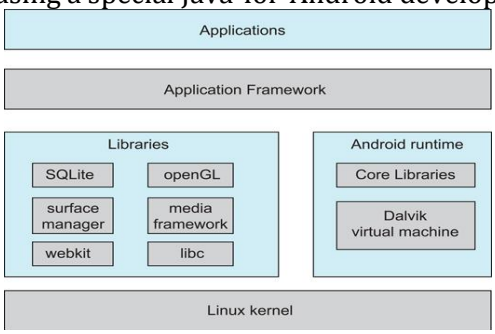
iOS

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

Architecture of Apple's iOS

Android

- The Android OS was developed for Android smart phones and tablets by the Open Handset Alliance, primarily Google.
- Android is an open-source OS, as opposed to iOS, which has lead to its popularity.
- Android includes versions of Linux and a Java virtual machine both optimized for small platforms.
- Android apps are developed using a special Java-for-Android development environment.



Architecture of Google's Android

1.13 Operating system debugging

- ✓ Debugging here includes both error discovery and elimination and performance tuning.
- ✓ ***Kernighan's Law***
  - *"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."*

Failure Analysis

- **Debugging** is the activity of finding and fixing errors in a system, both in hardware and in software.
- Performance problems are considered bugs, so debugging can also include performance tuning.
- If a process fails, most operating systems write the error information to a **log file** to alert system operators or users that the problem occurred.
- The operating system can also take a core dump—a capture of the memory of the process— and store it in a file for later analysis. (Memory was referred to as the "core" in the early days of computing.)
- Running programs and **core dumps** can be used by a debugger, which allows a programmer to explore the code and memory of a process.
- A failure in the kernel is called **a crash**. When a crash occurs, error information is saved to a log file, and the memory state is saved to a **crash dump**.

Performance Tuning

- Performance tuning seeks to improve performance by removing processing bottlenecks.
- To identify bottlenecks, we must be able to monitor system performance.
- In a number of systems, all interesting events are logged with their time and important parameters and are written to a file. Later, an analysis program can process the log file to determine system performance and to identify bottlenecks and inefficiencies.
- Another approach to performance tuning uses single-purpose, interactive tools that allow users and administrators to question the state of various system components to look for bottlenecks.
    - The Windows Task Manager includes information for current applications as well as processes, CPU and memory usage, and networking statistics Improve performance by removing bottlenecks OS must provide means of computing and displaying measures of system behavior For example, "top" program or Windows Task Manager

Dtrace- *Dynamic Tracing*

- DTrace (DTrace.exe) is a command-line tool that displays system information and events.
- DTrace is an open source tracing platform ported to windows.
- DTrace was originally developed for the Solaris operating system.
- It provides dynamic instrumentation of both user/kernel functions, the ability to script using the D-language, speculative tracing.
-

1.14 System Boot

- The procedure of starting a computer by loading the kernel is known as booting the system.
- On most computer systems, a small piece of code known as the bootstrap program or bootstrap loader locates the kernel.
- This program is in the form of read-only-memory(ROM),because the RAM is in an unknown state at system startup.ROM is convenient because it needs no initialization and cannot be infected by a computer virus
- When the full bootstrap program has been loaded, it can traverse the file system to find the os kernel, load it into memory and start its execution. It is only at this point that the system is said to be Running