Unit - I

Introduction to Operating System Concept

Syllabus

- Introduction
- Types of operating systems
- operating systems concepts
- operating systems services
- Introduction to System call
- System Call types.

Introduction :

Operating System Definitions

- An **operating system** is a program that manages the computer hardware.
- An *operating system is software,* acts as an intermediary between the user of a computer and the computer hardware.
- It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware.
- An operating system is software that manages the computer hardware.
- Operating systems for handheld computers are designed to provide an environment in which a user can easily interface with the computer to execute programs.

Goal of O.S

- ***Convenient :*** O.S provides to users easily interface with the computer to execute programs.
- ***Efficient :*** O.S provides system resources use efficiently.

What Operating Systems Do

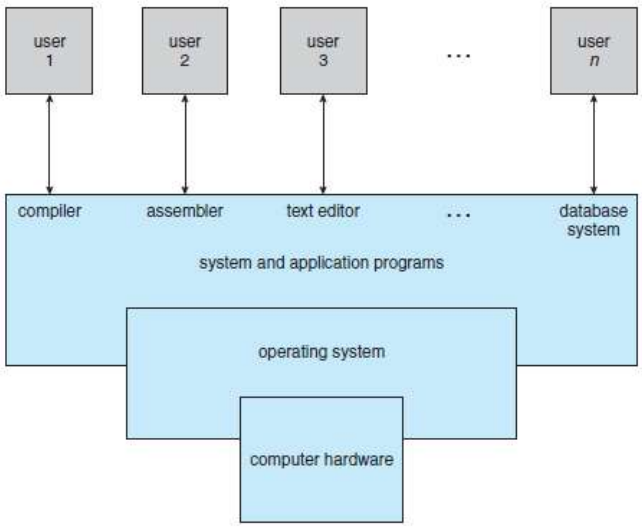


Figure 1.1 Abstract view of the components of a computer system.

A computer system can be divided roughly into four components: the ***hardware,*** the ***operating system,*** the ***application programs,*** and the ***users.***

- **Hardware**—the **central processing unit (CPU)**, the **memory**,
- **input/output (I/O) devices**—provides the basic computing resources for the system.

- **application programs**—such as word processors, spreadsheets, compilers, and Web browsers

- ***User*** **-** use the various application programs in system.

To understand more fully the operating system's role in two views

1. User View
2. System View

User View;

- The operating system is designed mostly for **ease of use**, with some attention paid to performance
- To **resource utilization**—how various hardware and software resources are shared.

System View

- An operating system as a **resource allocator**
- A **control program** manages the execution of user programs to prevent errors and improper use of the computer.

1.2 Computer-System Organization

Computer-System Operation :

A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory.

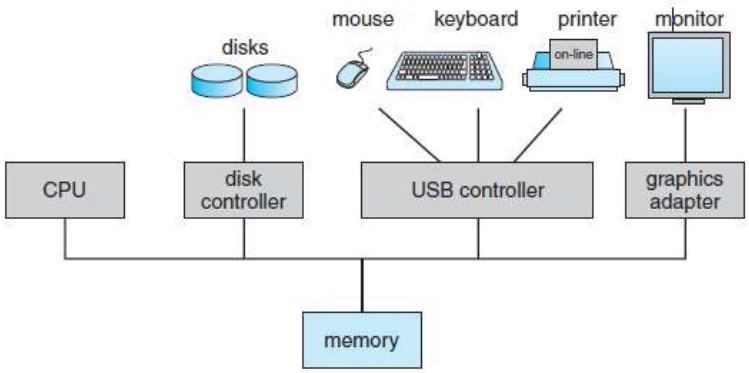


Figure 1.2 A modern computer system.

- For a computer to start running—for instance, when it is powered up or rebooted—it needs to have an initial program to run. This initial program, or **bootstrap program**, tends to be simple.
- Typically, it is stored within the computer hardware in read-only memory (**ROM**)/(**EEPROM**).

➢ The occurrence of an event is usually signalled by an **interrupt** from either the hardware or the software.

➢ **Hardware** may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus.

➢ **Software** may trigger an interrupt by executing a special operation called a **system call**(also called a **monitor call**).

Storage Structure

CPU can load instructions only from memory, so any programs to run must be stored there. General-purpose computers run most of their programs from rewritable memory, called main memory (also called **random-access memory**, or **RAM**). Main memory commonly is implemented in a semiconductor technology called **dynamic random-access memory (DRAM)**.

The top four levels of memory may be constructed using semiconductor memory.

We want the programs and data to reside in main memory permanently. This arrangement usually is not possible for the following two reasons:

1. Main memory is usually too small to store all needed programs and data permanently.

2. Main memory is a **volatile** storage device that loses its contents when power is turned off or otherwise lost.

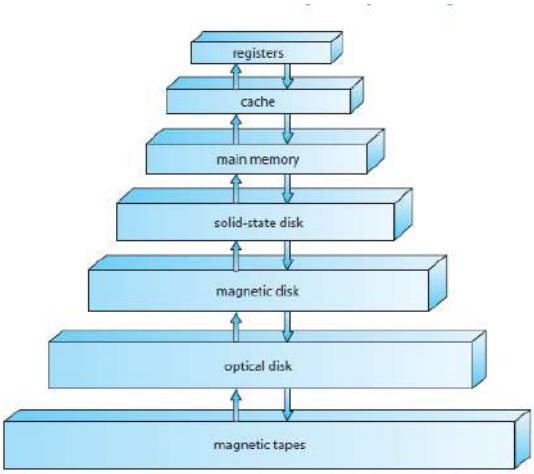


Figure 1.4 Storage-device hierarchy.

- Thus, most computer systems provide **secondary storage** as an extension of main memory.
- **volatile storage** loses its contents when the power to the device is removed.
- **Non-volatile storage** for safekeeping.

I/O Structure:

A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus. Each device controller is in charge of a specific type of device. Depending on the controller, more than one device may be attached. For instance, seven or more devices can be attached to the **small computer-systems interface (SCSI)** controller.

A device controller maintains some local buffer storage and a set of special-purpose registers. The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage. Typically, operating systems have a **device driver** for each device controller.

- when used for bulk data movement such as disk I/O used **direct memory access (DMA)**.

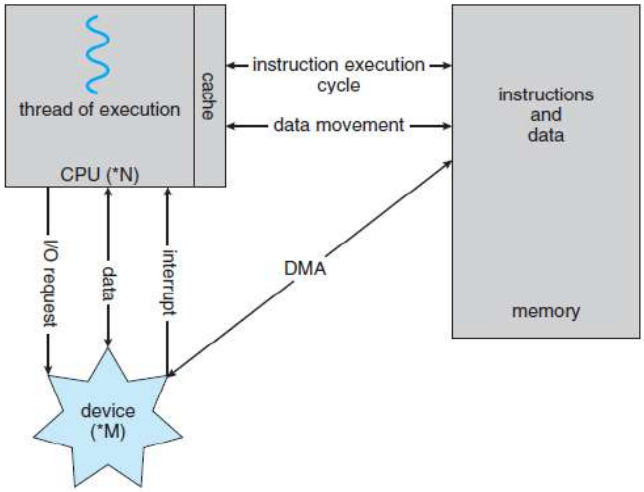


Figure 1.5 How a modern computer system works.

Operating-System Operations:

- A **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.

- Os work in two separate ***modes*** of operation: **user mode** and **kernel mode** (also called **supervisor mode**, **system mode**, or **privileged mode**).

- A bit, called the **mode bit**, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).

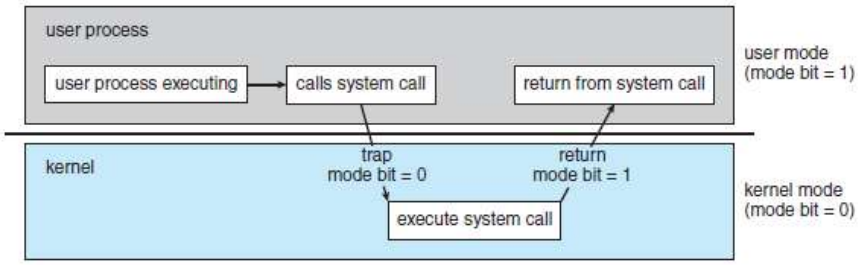


Figure 1.10 Transition from user to kernel mode.

- **Timer :** a user program to get stuck in an infinite loop or to fail to call system services and never return control to the operating system.

Computer-System Architecture

- Categorize roughly into two types according to the number of **general-purpose processors** used.
  1. Single-Processor Systems
  2. Multiprocessor Systems
- **Single-Processor Systems:**
  - On a single-processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.
- **Multiprocessor Systems (parallel systems** or **multi core systems)**
  - This systems have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.

1. Types of operating systems

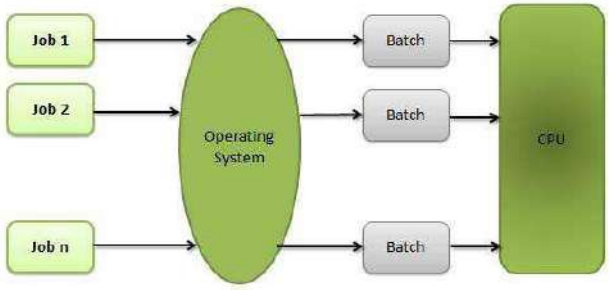
Following are some of the most widely used types of Operating system.

1. Simple Batch System
2. Multiprogramming System
3. Multiprocessor System
4. Clustered System
5. Distributed Operating System
6. Network Operating System
7. Real-time Operating System
8. Special Purpose System

1. Batch Processing

Batch processing is a technique in which an Operating System collects the programs and data together in a batch before processing starts. An operating system does the following activities related to batch processing:

- The OS defines a job which has predefined sequence of commands, programs and data as a single unit.
- Jobs are processed in the order of submission, i.e., first come first served fashion.
- When a job completes its execution, its memory is released and the output for the job gets copied into an output spool for later printing or processing.
- To speed up processing, jobs with similar needs are batched together and run as a group.
- The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.



The problems with Batch Systems are as follows –

1. Lack of interaction between the user and the job.
2. CPU is often **idle**, because the speed of the mechanical I/O devices is slower than the CPU.
3. Difficult to provide the desired priority.

2. **Multiprogramming** : Increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

- The operating system keeps several jobs in memory simultaneously in general, main memory is too small to accommodate all jobs, the jobs are kept initially on the disk in the job pool.
- This pool consists of all processes residing on disk awaiting allocation of main memory.

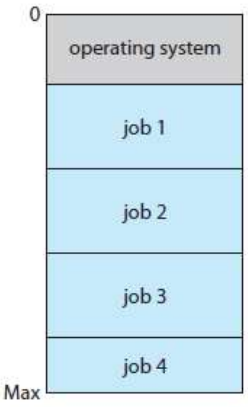
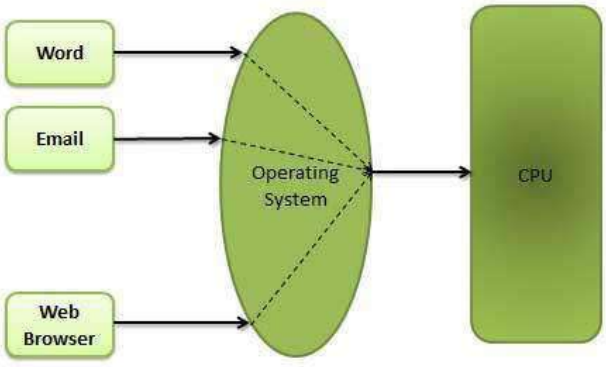


Figure 1.9 Memory layout for a multiprogramming system.

- ✓ The operating system picks and begins to execute one of the jobs in memory.
- ✓ Eventually, the job may have to wait for some task, such as an I/O operation, to complete.
- ✓ In a multiprogrammed system, the operating system simply switches to, and executes, another job.
- ✓ When that job needs to wait, the CPU switches to another job, and so on.
- ✓ Eventually, the first job finishes waiting and gets the CPU back. As long as at least one job needs to execute, the CPU is never idle.

Time sharing (or multitasking)

- ✓ **Multiprogrammed** systems provide an environment in which the various system resources (for example, CPU, memory, and peripheral devices) are utilized effectively, but they do **not provide for user interaction** with the computer system.
- ✓ **Time sharing** (or multitasking) is a **logical extension** of **multiprogramming.**



- ✓ In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.
- ✓ Time sharing requires an interactive computer system, which provides direct communication between the user and the system.

- ✓ The user gives instructions to the operating system or to a program directly, using a input device such as a keyboard, mouse, touch pad, or touch screen.

- ✓ A time-shared operating system allows many users to share the computer simultaneously

3. Multiprocessor systems

Three main advantages:

1. **Increased throughput:** By increasing the number of processors, we expect to get more work done in less time.
2. **Economy of scale:** Multiprocessor systems can cost less than equivalent multiple single-processor systems, because they can share peripherals, mass storage, and power supplies
3. **Increased reliability:** If we have ten processors and one fails, then each of the remaining nine processors can pick up a share of the work of the failed processor.

The ability to continue providing service proportional to the level of surviving hardware is called **graceful degradation**. Some systems go beyond graceful degradation and are called **fault tolerant**.

The multiple-processor systems are two types:

1. **asymmetric multiprocessing**
2. **symmetric multiprocessing**

1. Asymmetric Multiprocessing: A master processor controls the system; the other processors either look to the master for instruction or have predefined tasks. This scheme defines a master-slave relationship. The master processor schedules and allocates work to the slave processors.

Eg : Sun's operating system SunOS Version 4

2. Symmetric Multiprocessing :

In which each processor performs all tasks within the operating system. SMP means that all processors are peers; no master-slave relationship exists between processors.

Ex : Windows, Mac OS X, and Linux

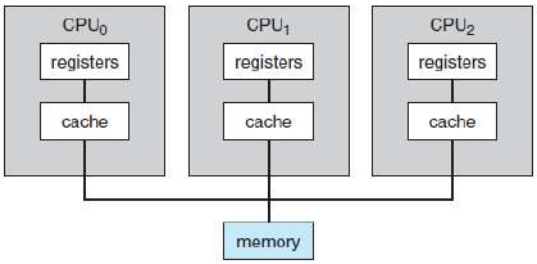


Figure 1.6 Symmetric multiprocessing architecture.

Another type of multiprocessor system is a **clustered system.**

4. Clustered System: which gathers together multiple CPUs.

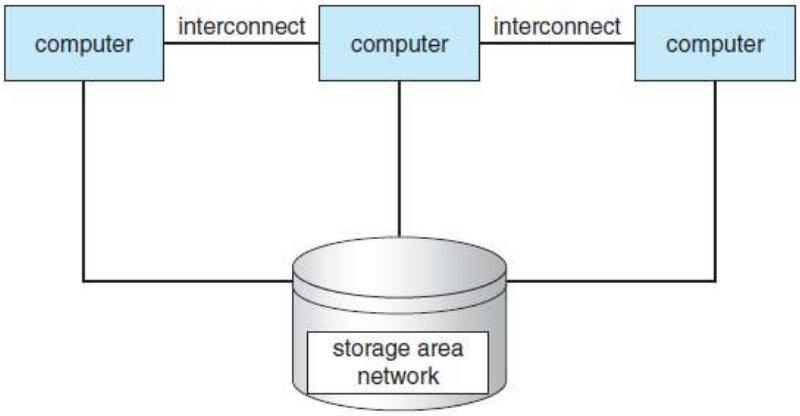


Figure 1.8 General structure of a clustered system.

- ✓ This are composed of two or more individual systems—or nodes—joined together. Such systems are considered **loosely coupled**. Each node may be a single processor system or a multi-core System.
- ✓ Each node may be a single processor system or a multicore system.
- ✓ Clustering is usually used to provide **high-availability** service—that is, service will continue even if one or more systems in the cluster fail.

Clustering can be structured asymmetrically or symmetrically.

- In **asymmetric clustering**, one machine is in **hot-standby mode** while the other is running the applications.
- In **symmetric clustering**, two or more hosts are running applications and are monitoring each other.

Since a cluster consists of several computer systems connected via a network, clusters can also be used to provide **high-performance computing** environments.

The application must have been written specifically to take advantage of the cluster, however.

This involves a technique known as **parallelization**, which divides a program into separate components that run in parallel on individual computers in the cluster.

5. Distributed Systems :

- A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide users with access to the various resources that the system maintains.
- Access to a shared resource increases computation speed, functionality, data availability, and reliability.
- A **distributed system** is a collection of loosely coupled nodes interconnected by a communication network.

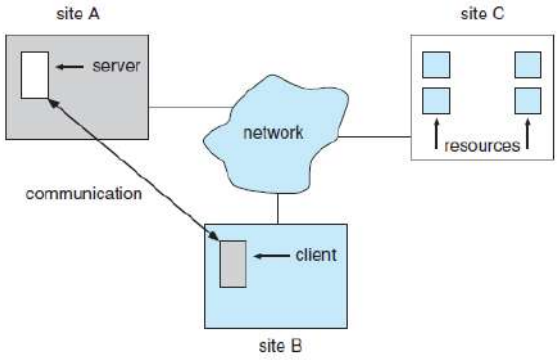


Figure 17.1 A distributed system.

- A **network**, in the simplest terms, is a communication path between two or more systems. Distributed systems depend on networking for their functionality.
- Networks are characterized based on the distances between their nodes.
    - A **local-area network (LAN)** connects computers within a room, a building, or a campus.
    - A **wide-area network (WAN)** usually links buildings, cities, or countries.
    - A **metropolitan-area network (MAN)** could link buildings within a city.
    - A **personal-area network (PAN)** between a phone and a headset or a Smartphone and a desktop computer.

Advantages of Distributed Systems:

- **Resource Sharing**: in a distributed system provides mechanisms for sharing files at remote sites, processing information in a distributed database, using remote specialized hardware devices

- **Computation Speedu**p if a particular site is currently overloaded with jobs, some of them can be moved to other, lightly loaded sites. This movement of jobs is called **load sharing** or **job migration**

- **Reliability** If one site fails in a distributed system, the remaining sites can continue operating, giving the system better reliability
- **Communication** When several sites are connected to one another by a communication network, users at the various sites have the opportunity to exchange information.

Types of Distributed Operating Systems

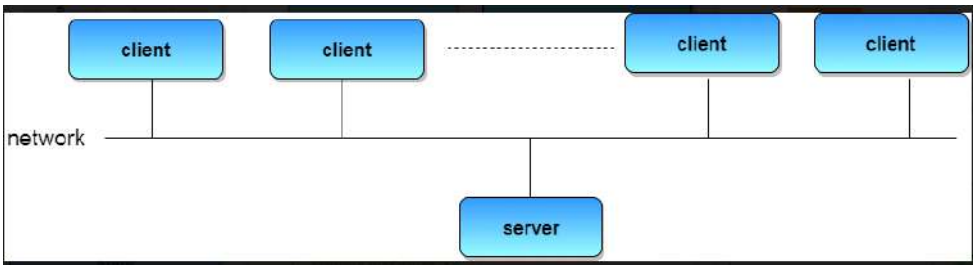
Following are the two types of distributed operating systems used:

1. Client-Server Systems
2. Peer-to-Peer Systems

Client-Server Systems

Centralized systems today act as **server systems** to satisfy requests generated by **client systems**.

The general structure of a client-server system is depicted in the figure below:



Server Systems can be broadly categorized as: **Compute Servers** and **File Servers**.

- **Compute Server systems**, provide an interface to which clients can send requests to perform an action, in response to which they execute the action and send back results to the client.
- **File Server systems**, provide a file-system interface where clients can create, update, read, and delete files.

Peer-to-Peer Systems

- The growth of computer networks - especially the Internet and World Wide Web (WWW) – has had a profound influence on the recent development of operating systems.
- In contrast to the **Tightly Coupled** systems, the computer networks used in these applications consist of a collection of processors that do not share memory or a clock.
- Instead, each processor has its own local memory.
- The processors communicate with one another through various communication lines, such as high-speed buses or telephone lines.

These systems are usually referred to as loosely coupled systems (or distributed systems).

The general structure of a **Peer-to-Peer** system is depicted in the figure below:

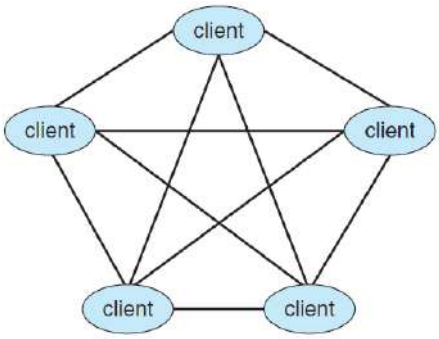


Fig : Peer-to-peer system with no centralized service

6. Network Operating Systems

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

- A computer running a network operating system acts autonomously from all other computers on the network, although it is aware of the network and is able to communicate with other networked computers.Adistributed operating system provides a less autonomous environment.

- The different computers communicate closely enough to provide the illusion that only a single operating system controls the network.

7. Special-Purpose Systems

i) Real-Time Embedded Systems

Embedded computers are the most prevalent form of computers in existence.

EX : MICROWAVE OVENS, WASHING MACHINES

- Embedded systems almost always run real-time operating systems.
- **A real-time system** is used when rigid time requirements have been placed on the operation of a processor or the flow of data.

Real-time computing is of two types: **Hard and Soft**.

- **Hard Real-Time System** has the most stringent requirements, guaranteeing that critical real time tasks be completed within their deadlines.
- **Soft Real-Time System** is less restrictive, simply providing that a critical real-time task will receive priority over other tasks and that it will retain that priority until it completes.

Characteristics of real-time systems are:

• Single purpose

• Small size

• Specific timing requirements

Multimedia Systems

- **Multimedia** data consist of audio and video files as well as conventional files. These data differ from conventional data in that multimedia data—such as frames of video
  - EX : MP3 DVD movies, video conferencing, and short video clips

Handheld Systems

Handheld systems include personal digital assistants (PDAs), such as Palm and Pocket-PCs, and cellular telephones, many of which use special-purpose embedded operating systems.

- Developers of handheld systems and applications face many challenges
  a. Limited size ,
  b. Slow processors,
  c. Small display screens.
  d. Faster processors require **more power**.
  e. Currently, many handheld devices do **not use virtual memory** techniques

- web page is delivered and displayed on the handheld device is called "**web clipping**"

2. O.S Concepts / Functions

1. Process Management :

- A program does nothing unless its instructions are executed by a CPU.

- A program in execution, as mentioned, is a process.

  Ex:

  - A time-shared user program such as a **compiler** is a process.
  - A **word-processing** program being run by an individual user on a PC is a process.

- A process needs certain resources—including CPU time, memory, files, and I/O devices—to accomplish its task.
- These resources are either given to the process when it is created or allocated to it while it is running.

- The operating system is responsible for the following activities in connection with process management:
  - Scheduling processes and threads on the CPUs
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for deadlock handling
  - Providing mechanisms for process communication

2. Memory Management

- Main memory is a large array of bytes, ranging in size from hundreds of thousands to billions.
- Each byte has its own address.
- Main memory is a repository of quickly accessible data shared by the CPU and I/O devices.
- To improve both the utilization of the CPU and the speed of the computer's response to its users, general-purpose computers must keep several programs in memory, creating a need for memory management.
- Many different memory management schemes are used.

- The operating system is responsible for the following activities in connection with memory management:
  - Keeping track of which parts of memory are currently being used and who is using them
  - Deciding which processes (or parts of processes) and data to move into and out of memory
  - Allocating and deallocating memory space as needed.

3. Storage Management

- The operating system provides a uniform, logical view of information storage.

File-System Management:

The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, the **file**.

File management is one of the most visible components of an operating system.

- The operating system is responsible for the following activities in connection with file management:
  - Creating and deleting files
  - Creating and deleting directories to organize files
  - Supporting primitives for manipulating files and directories
  - Mapping files onto secondary storage
  - Backing up files on stable (nonvolatile) storage media.

Mass-Storage Management

- Main memory is too small to accommodate all data and programs, and because the data that it holds are lost when power is lost, the computer system must provide secondary storage to back up main memory.

- The operating system implements the abstract concept of a file by managing mass-storage media, such as tapes and disks, and the devices that control them.

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

- Free-space management
- Storage allocation
- Disk scheduling

I/O Systems:

The I/O subsystem consists of several components:

- A memory-management component that includes buffering, caching, and spooling
- A general device-driver interface
- Drivers for specific hardware devices

4. Protection and Security

- **Protection**, then, is any mechanism for controlling the access of processes or users to the resources defined by a computer system. This mechanism must provide means to specify the controls to be imposed and to enforce the controls.

- Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.

- **Security** to defend a system from external and internal attacks. Such attacks spread across a huge range and include viruses and worms, denial-of service attacks (which use all of a system's resources and so keep legitimate users out of the system), identity theft, and theft of service (unauthorized use of a system).

- Protection and security require the system to be able to distinguish among all its users. Most operating systems maintain a list of user names and associated **user identifiers (user IDs)**. In Windows parlance, this is a **security ID (SID)**.

3. Operating-System Services

An operating system provides an environment for the execution of programs.
It provides certain services to programs and to the users of those programs

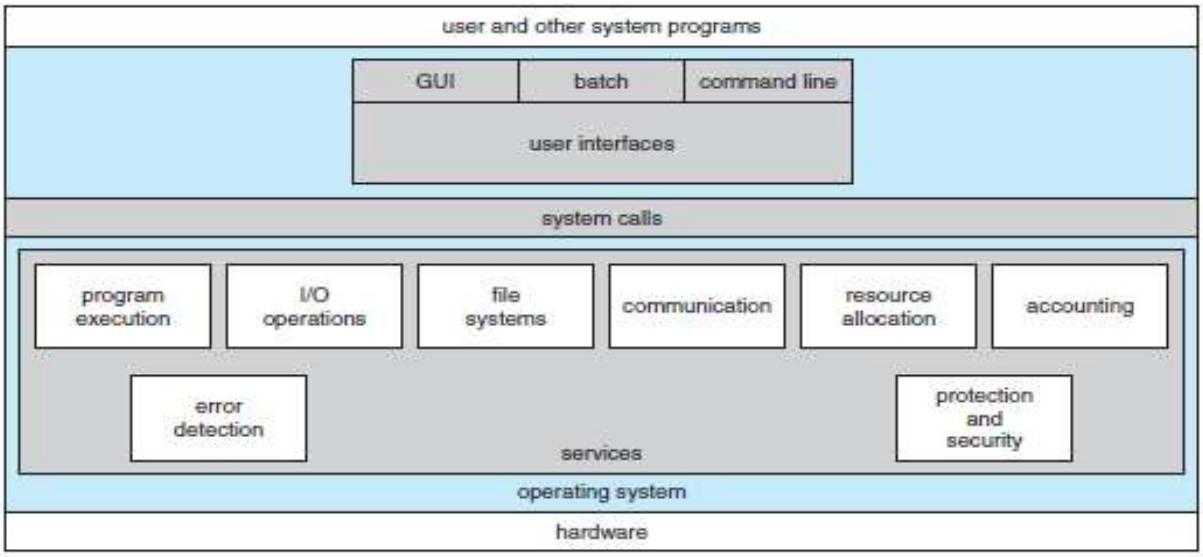


Figure 2.1 A view of operating system services.

Set of operating system services provides functions that are helpful to the user

USER VIEW:

- **User interface**. Almost all operating systems have a **user interface (UI)**. This interface can take several forms.
    - One is a **command-line interface (CLI)**, which uses text commands and a method for entering them (say, a keyboard for typing in commands in a specific format with specific options).
    - Another is a **batch interface**, in which commands and directives to control those commands are entered into files, and those files are executed.
    - Most commonly, a **graphical user interface (GUI)** is used.

- **Program execution**. The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally.

- **I/O operations**. A running program may require I/O, which may involve a file or an I/O device.

- **File-system manipulation**. programs need to read and write files and directories. They also need to create and delete them by name, search for a given file, and list file information.

- **Communications**. There are many circumstances in which one process needs to exchange information with another process.

    - Communications may be implemented via **shared memory**, in which two or more processes read and write to a shared section of memory**.**
    - **message passing**, in which packets of information in predefined formats are moved between processes by the operating system.

- **Error detection**. The operating system needs to be detecting and correcting errors constantly . Errors may occur in the CPU and memory hardware , in I/O devices, and in the user program

SYSTEM VIEW:

- **Resource allocation**. When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them. The operating system manages many different types of resources

- **Accounting**. We want to keep track of which users use how much and what kinds of computer resources. This record keeping may be used for accounting

- **Protection and security**. The owners of information stored in a multiuser or networked computer system may want to control use of that information

4. System Calls

- **System calls** provide an interface to the services made available by an operating system.
- These calls are generally available as routines written in **C** and **C++,** although certain low-level tasks (for example, tasks where hardware must be accessed directly) may have to be written using assembly-language instructions.

This system-call sequence is

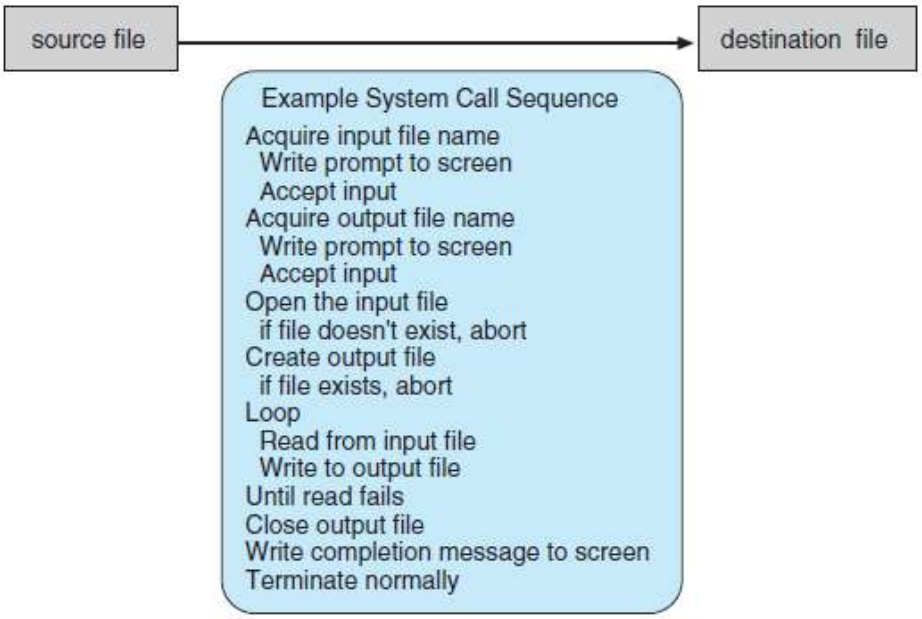


Figure 2.5 Example of how system calls are used.

- A **system call interface** that serves as the link to system calls made available by the operating system.

- The system-call interface intercepts function calls in the API and invokes the necessary system calls within the operating system.

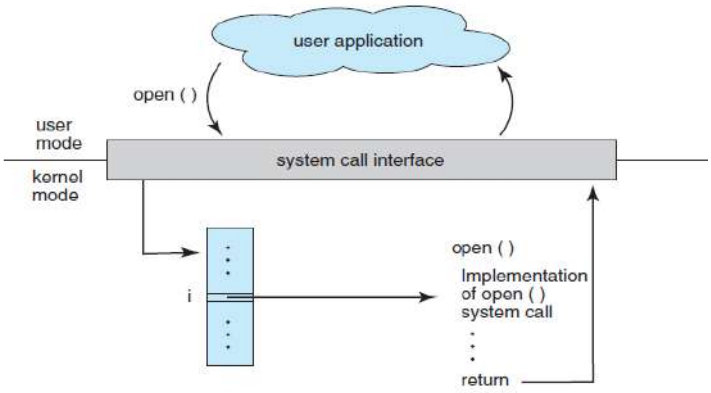
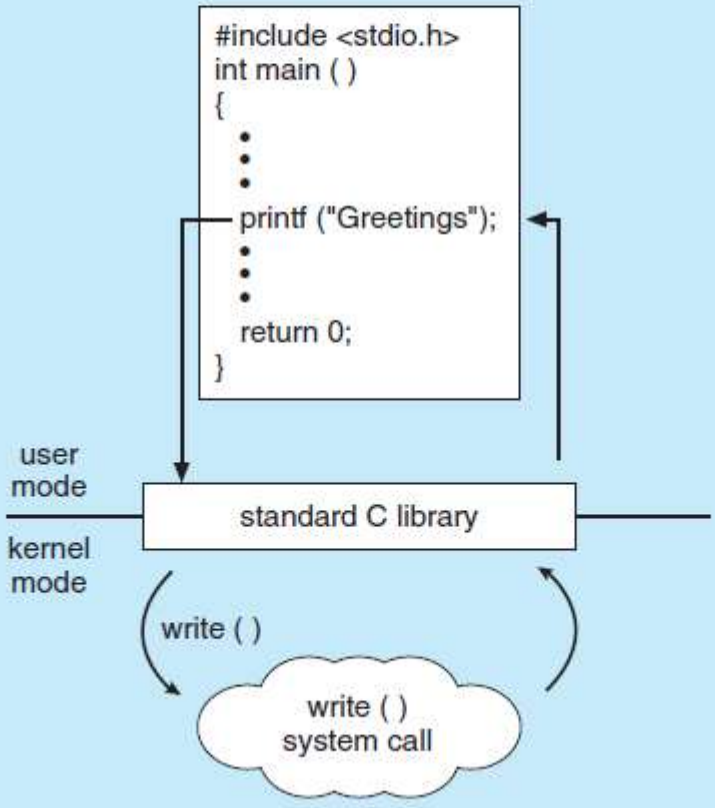


Figure 2.6 The handling of a user application invoking the `open()` system call.

EXAMPLE OF STANDARD C LIBRARY

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the `printf()` statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance, the `write()` system call. The C library takes the value returned by `write()` and passes it back to the user program. This is shown below:



Three general methods are used to pass parameters to the operating system.

1. The simplest approach is to pass the parameters in **registers.**
2. In some cases, however, there may be more parameters than registers. In these cases, the parameters are generally stored in **a block,** or **table**, in memory, and the address of the block is passed as a parameter in a register (Figure 2.7).

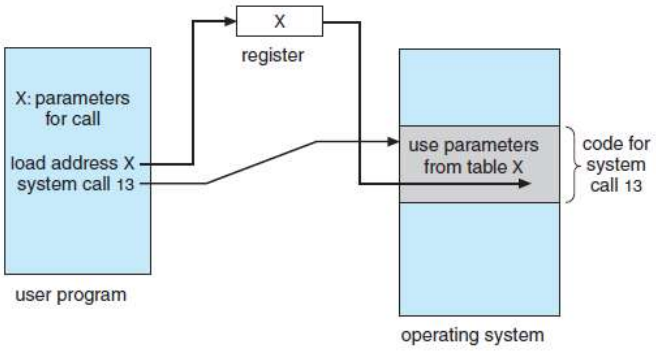


Figure 2.7 Passing of parameters as a table.

3. Parameters also can be placed, or pushed, onto the **stack** by the program and popped off the stack by the operating system.

6 . Types of System Calls

System calls can be grouped roughly into six major categories:

1. **process control**, 2. **file manipulation**, 3. **device manipulation**, 4. **information maintenance**, 5. **communications**, and 6. **protection**.

- **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**
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes
- **Device management**
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- **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
- **Communications**
  - create, delete communication connection
  - send, receive messages
  - Transfer status information
  - attach or detach remote devices
- **Protection**
  - set permission, get permission
  - allow user, deny user

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

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

Extra information and important in R13 not in R16

Operating-System Structure

- A system as large and complex as a modern operating system must be engineered carefully if it is to function properly and be modified easily

- A common approach is to partition the task into small components, or modules, rather than have one **monolithic** system. Each of these modules should be a well-defined portion of the system, with carefully defined inputs, outputs, and functions.

1. Simple Structure

Many operating systems do not have well-defined structures. Frequently, such systems started as small, simple, and limited systems and then grew beyond their original scope.

- MS-DOS is an example of such a system.

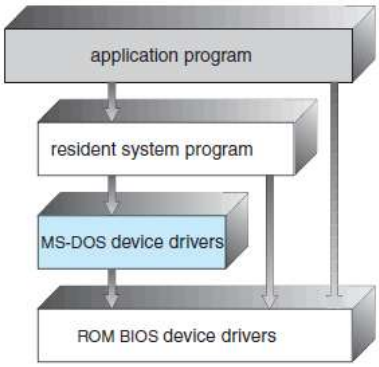


Figure 2.11 MS-DOS layer structure.

Advantages

- Small, simple, and limited systems and then grew beyond their original scope.
- It was written to provide the most functionality in the least space, so it was not carefully divided into modules.

Disadvantages :

- The interfaces and levels of functionality are not well separated
- Application programs are able to access the basic I/O routines to write directly to the display and disk drives.
- Vulnerable to errant (or malicious) programs, causing entire system crashes when user programs fail.

- Limited by the hardware.

Another example of limited structuring is the original UNIX operating system.

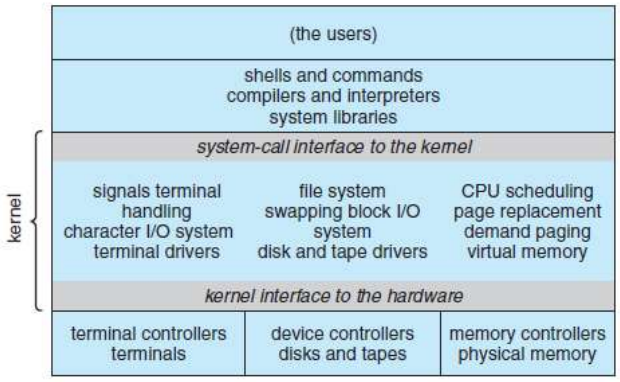


Figure 2.12 Traditional UNIX system structure.

Advantages

- This simple, monolithic structure in the UNIX, Linux, and Windows operating systems.
- It consists of two separable parts: the kernel and the system programs.
- The kernel is further separated into a series of interfaces and device drivers
- The kernel provides the file system, CPU scheduling, memory management, and other operating-system functions through system calls.

Disadvantages :

- This structure was difficult to implement and maintain.
- There is very little overhead in the system call interface or in communication within the kernel

Layered Approach

- A system can be made modular in many ways. One method is the **layered approach**, in which the operating system is broken into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer *N*) is the user interface.

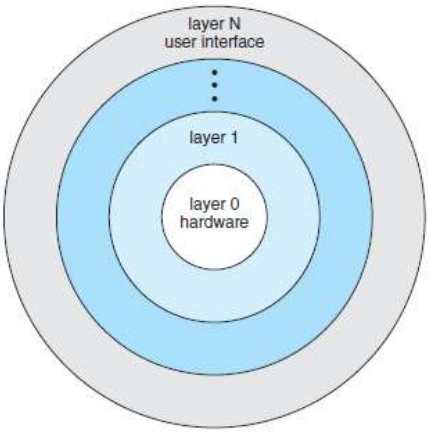


Figure 2.13 A layered operating system.

Advantage:

- Layered approach is simplicity of construction and debugging.
- Each layer is implemented only with operations provided by lower-level layers.
- Each layer hides the existence of certain data structures, operations, and hardware from higher-level layers.

Disadvantages:

- A layer can use only lower-level layers, careful planning is necessary.
- Other requirements may not be so obvious.
- Layered implementations are that they tend to be less efficient than other types.

Microkernels

- In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called **Mach** that modularized the kernel using the **microkernel** approach.

- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs.

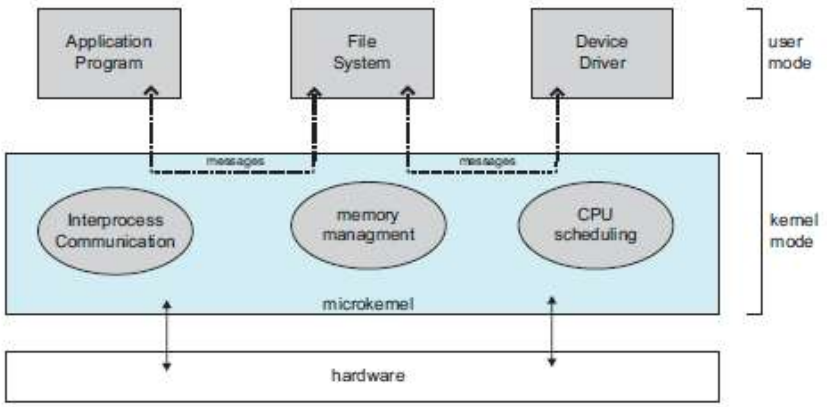


Figure 2.14 Architecture of a typical microkernel.

Advantages

- ✓ microkernel is to provide communication between the client program and the various services that are also running in user space.
- ✓ it makes extending the operating system easier

Disadvantages

- ✓ The performance of microkernels can suffer due to increased system-function overhead.

Modules

- ✓ Operating-system design involves using object-oriented programming techniques to create a modular kernel.
- ✓ The kernel has a set of core components and dynamically links in additional services either during boot time or during run time.

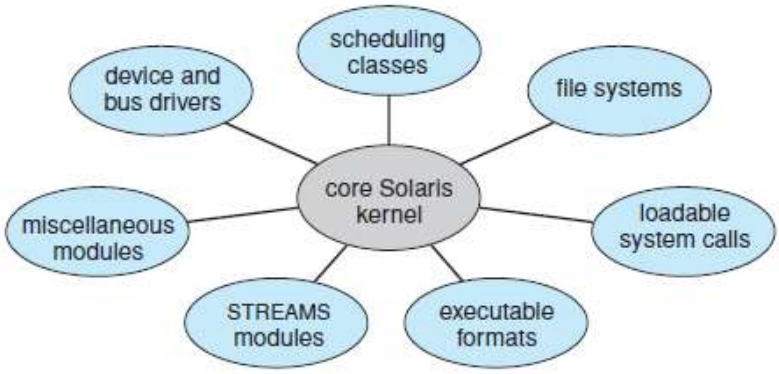


Figure 2.15 Solaris loadable modules.

- ✓ 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 Structure

The Apple Macintosh Mac OS X operating system uses a **hybrid structure**. Mac OS X (also known as ***Danvin)*** structures the operating system using a layered technique where one layer consists of the Mach microkernel

- ➢ The top layers include application environments and a set of services providing a graphical interface to applications.
- ➢ Below these layers is the kernel environment, which consists primarily of the Mach microkernel and the BSD kernel.
- ➢ Mach provides
  - o memory management
  - o for remote procedurecalls (RPCs) and interprocess communication (IPC) facilities,
  - o including message passing; and thread scheduling
- ➢ BSD kernel
  - o The BSD component provides a BSD command line interface, support for networking and file systems, and an implementation of POSIX APIs, including Pthreads.

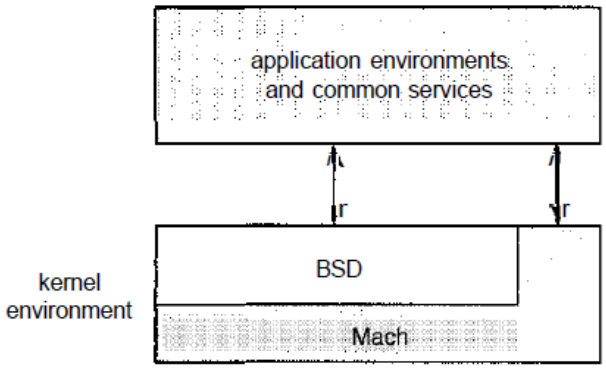


Figure 2.14 The Mac OS X structure.

9. Operating-System Generation

- ✓ Operating systems are designed to run on any of a class of machines at a variety of sites with a variety of peripheral configurations.
- ✓ The system must then be configured or generated for each specific computer site, a process sometimes known as **system generation** (SYSGEN).

The SYSGEN program reads from a given file, or asks the operator of the system for information.

The following kinds of information

- ✓ What CPU is to be used?
- ✓ How much memory is available?
- ✓ What devices are available?
- ✓ What operating-system options are desired.

HISTORY OF OPERATING SYSTEMS

The first true digital computer was designed by the English mathematician Charles Babbage (1792–1871).

- **The First Generation (1945–55): Vacuum Tubes**
- **The Second Generation (1955–65): Transistors and Batch Systems**
- **The Third Generation (1965–1980): ICs and Multiprogramming**
- **The Fourth Generation (1980–Present): Personal Computers**
- **The Fifth Generation (1990–Present): Mobile Computers**