UNIT-I

Operating System - Introduction, Structures - Simple Batch, Multiprogrammed, Time-shared, Personal Computer, Parallel, Distributed Systems, Real-Time Systems, System components, Operating System services, System Calls.

An operating System (OS) is an intermediary between users and computer hardware. It provides users an environment in which a user can execute programs conveniently and efficiently.

Definition:

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

What operating system Do?

- An operating system performs the following basic tasks:
- It helps the users interact with the computer system
- Controlling and allocating memory
- Managing file systems
- Controlling input and output devices
- Protecting data
- Facilitating networking

A computer system can be divided roughly into four components:

- ✓ Hardware
- ✓ Operating system
- ✓ Application programs
- ✓ Users

Operating System is designed both by taking user view and system view into consideration.

User View

1. The goal of the Operating System is to maximize the work and minimize the effort of the user.

2. Most of the systems are designed to be operated by single user; however in some systems multiple users can share resources, memory. In these cases Operating System is designed to handle available resources among multiple users and CPU efficiently.
3. Operating System must be designed by taking both usability and efficient resource utilization into view.

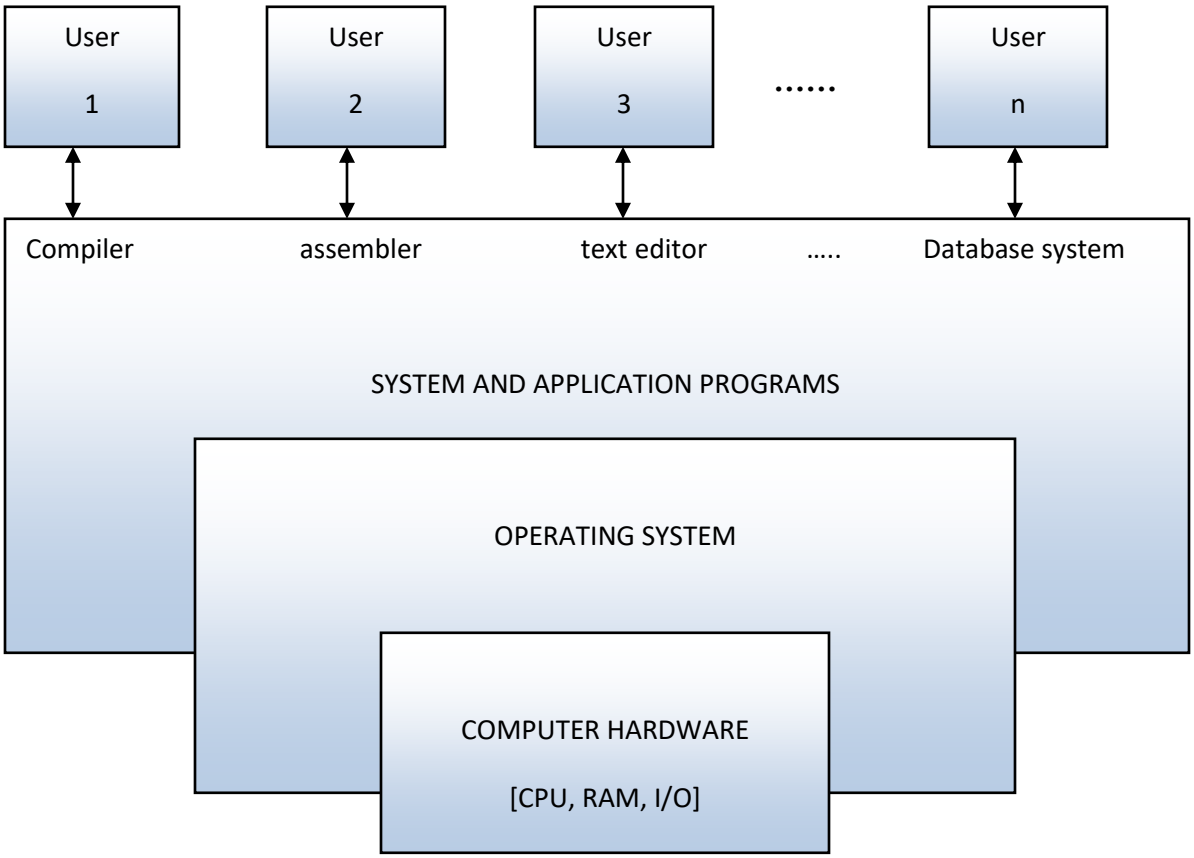


Fig 1: Abstract view of the components of a computer system

4. In embedded systems (Automated systems) user view is not present.
5. Operating System gives an effect to the user as if the processor is dealing only with the current task, but in background processor is dealing with several processes.

System View

1. From the system point of view Operating System is a program involved with the hardware.
2. Operating System is allocator, which allocate memory, resources among various processes. It controls the sharing of resources among programs.
3. It prevents improper usage, error and handles deadlock conditions.
4. It is a program that runs all the time in the system in the form of Kernel.
5. It controls application programs that are not part of Kernel.

Operating System Objectives/Goals:

- **Convenience** – makes computer user friendly.
- **Efficiency**- allows computer to use resources efficiently.
- **Ability to evolve**- constructed in a way to permit effective development, testing and introduction of new functions without interfering with service

Operating System Functions / components

1) Process Management:

- ❖ A program in execution ,it is called a process
- ❖ A system task, such as sending output to a printer, can also be a process.
- ❖ Consider a process to be a job

- A process is the unit of work in a system. Such a system consist of a collection of processes, some of which are operating system processes (those that execute system code) and the rest of which are user processes (those that execute user code).
- A process needs resources requirements (such as CPU time, memory, files and I/O devices) of a process during its execution are managed by the operating system.

Operating System does the following activities for processor management.

- ➢ Creating and deleting both user and system processes
- ➢ Scheduling processes and threads on the CPUs
- ➢ Suspending and resuming processes
- ➢ Providing mechanisms for process synchronization
- ➢ Providing mechanism for deadlock handling

2) Memory Management:

- Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.
- Main memory provides a fast storage that can be access directly by the CPU. So for a program to be executed, it must in the main memory.

Operating System does the following activities for memory management.

- ➢ Keeps tracks of primary memory i.e. what part of it are in use by whom, what part are not in use.

- In multiprogramming, OS decides which process will get memory when and how much.
- Allocates the memory when the process requests it to do so.
- De-allocates the memory when the process no longer needs it or has been terminated.

3) Storage Management:

a) File-System Management:

- Data on a computer is saved in the form of files. File system normally organized into directories for easy navigation and usage. These directories may contains files and other directories

An OS is responsible for the following tasks with regards to file system management:

- Creating and deleting files and directories
- Supporting primitives for manipulating files and directories. (open, flush, etc.)
- Mapping files onto secondary storage.
- Backing up files onto stable permanent storage media.

b) 2. Mass-Storage Management:

An OS is responsible for the following tasks with regards to mass-storage management:

- Free disk space management
- Storage allocation
- Disk scheduling

Note the trade-offs regarding size, speed, longevity, security, and re-writability between different mass storage devices, including floppy disks, hard disks, tape drives, CDs, DVDs, etc

c) Cache memory

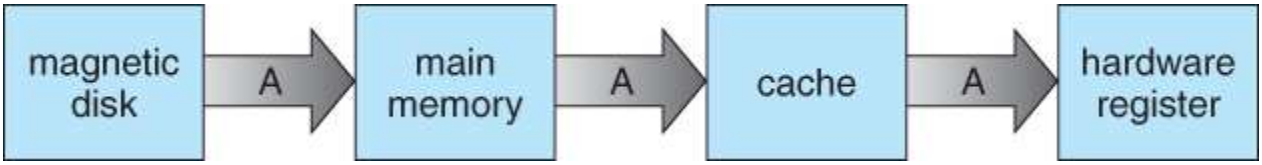
- There are many cases in which a smaller higher-speed storage space serves as a cache, or temporary storage, for some of the most frequently needed portions of larger slower storage areas.
- The hierarchy of memory storage ranges from CPU registers to hard drives and external storage. (See table below.)
- The OS is responsible for determining what information to store in what level of cache, and when to transfer data from one level to another.
- The proper choice of cache management can have a profound impact on system performance.
- Data read in from disk follows a migration path from the hard drive to main memory, then to the CPU cache, and finally to the registers before it can be used, while data being written follows the reverse path.

- Each step (other than the registers) will typically fetch more data than is immediately needed, and cache the excess in order to satisfy future requests faster. For writing, small amounts of data are frequently buffered until there is enough to fill an entire "block" on the next output device in the chain.

- The issues get more complicated when multiple processes (or worse multiple computers) access common data, as it is important to ensure that every access reaches the most up-to-date copy of the cached data (amongst several copies in different cache levels.)

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Figure 2 - Performance of various levels of storage



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

Operating system provides protection and security at level such as:

- Users
- Application programs
- Hardware

Users: operating system provides passwords and access controls to the users to avoid unauthorized users from accessing the data.

Application program: operating system provides firewall as part of virus management, alert messages.

Hardware: alert messages, access control of devices for users through GroupID, passwords, etc..

- ***Protection*** involves ensuring that no process access or interfere with resources to which they are not entitled, either by design or by accident. (E.g. "protection faults" when pointer variables are misused.)
- ***Security*** involves protecting the system from deliberate attacks, either from legitimate users of the system attempting to gain unauthorized access and privileges, or external attackers attempting to access or damage the system.

5. Resource Management

Operating System manages the resources such as I/O, RAM etc.

An OS is responsible for the following tasks with regards to Resource management:

- Installing drivers required for I/O and memory
- Communicates with devices through devices controls
- Coordinates among peripherals

6. Communication Management

- Operating systems acts as a three way communicator (command interpreter) between the users, application programs and the hardware.
- It provides an interface for a computer to communicate with other computer via a network.

Evolution of operating systems

Batch operating system:

- The users of batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator.
- To speed up processing, jobs with similar needs are batched together and run as a group.

The problems with Batch Systems are following.

- Lack of interaction between the user and job.
- CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU.
- Difficult to provide the desired priority.

Time-sharing operating systems:

- Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time.
- Processor's time which is shared among multiple users simultaneously is termed as time-sharing.
- The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of multiprogrammed batch systems, objective is to maximize processor use, whereas in Time-Sharing Systems objective is to minimize response time.
- Time sharing is also called as multitasking.

Advantages of Timesharing operating systems are following

- Provide advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Timesharing operating systems are following.

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

Distributed operating System:

- Distributed systems are a collection of physically separate computers connected via a network, sharing various resources the system maintains.
- Distributed systems use multiple central processors to serve multiple real time application and multiple users.
- The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as loosely coupled systems or distributed systems.
- Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, and computers and so on.

The advantages of distributed systems are following.

- With resource sharing facility user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

Network operating System;

- Network Operating System runs on a server and provides server the capability to manage data, users, groups, security, applications, and other networking functions.
- The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.
- **Examples:** Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD.

The advantages of network operating systems are following.

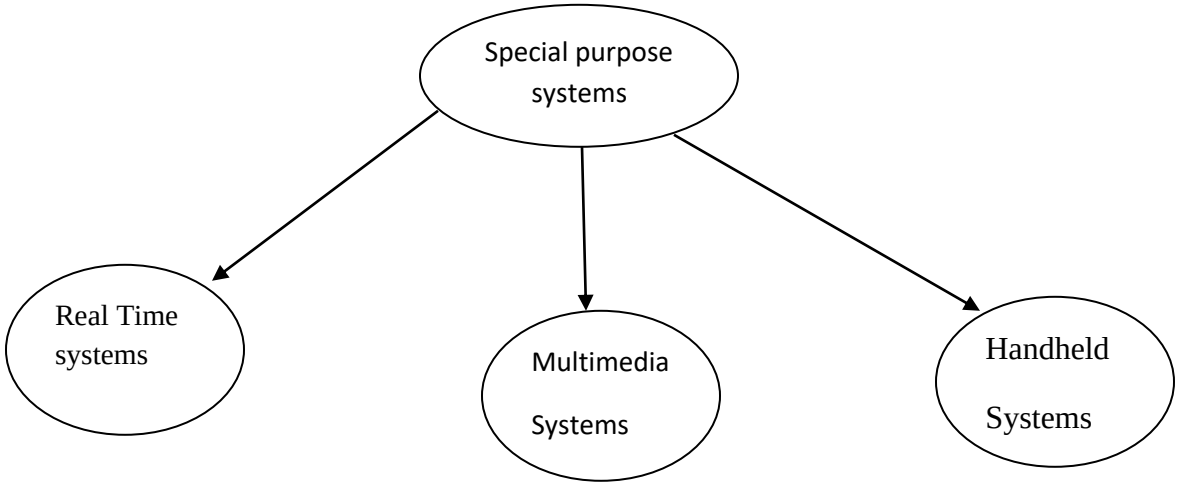
- Centralized servers are highly stable.
- Security is server managed.
- Upgrades to new technologies and hardware can be easily integrated into the system.
- Remote access to servers is possible from different locations and types of systems.

The disadvantages of network operating systems are following.

- High cost of buying and running a server.
- Dependency on a central location for most operations.
- Regular maintenance and updates are required.

Special purpose systems

Definition: "systems whose functions are limited and their objective is to deal with limited computational domains, are called special purpose systems."



Real Time operating System:

- Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application.
- Real-time operating system has well-defined, fixed time constraints otherwise system will fail.
- **Example**: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-appliance controllers, Air traffic control system etc.

Handheld Systems:

- Handheld systems include personal digital assistant (PDAs), such as palm and packet PCs, and cellular telephones.
- Handheld systems provide some features of a computer with small size, less weight and portability.
- Memory should be managed efficiently due to limited size and memory.

Interactivity Systems:

Interactivity refers that a User is capable to interact with computer system.

An OS is responsible for the following activities related to interactivity

- Operating system provides user an interface to interact with system.
- Operating system manages input devices to take inputs from the user. Example: keyboard.
- Operating system manages output devices to show outputs to the users. Example, Monitor

Parallel System

- A system is said to be a parallel system in which multiple processor have direct access to shared memory which forms a common address space.

- Usually tightly-coupled systems are referred to as parallel system.
- In these systems, there is a single system wide primary memory (address space) that is shared by all the processors.

Computer system architecture

Computer architecture means design or construction of a computer. A computer sustem may be organized in different ways. Some computer systems have single processor and other have multiprocessors. So computer systems categorized in these ways.

1. Single Processor Systems
2. Multiprocessor Systems
3. Clustered Systems

1. Single processor

- Some computers use only one processor such as microcomputers. On a single processor system, there is only one CPU that perform all the activities in the computer system, However, most of these systems have other special purpose processors, such as I/O Processor that move data rapidly among different components of the computer system.
- These processors execute only a limited system programs and do not run the user program. So we define single processor systems as “A system that has only one general purpose CPU, is considered as single processor system.

2. Multiprocessor systems

- Some systems have two or more processors. These systems are also known as parallel systems or tightly coupled systems.
- Mostly the processors of these systems share the common system bus (electronic path) memory and peripheral (input/output) devices.
- These systems are fast in data processing and have capability to execute more than one program simultaneously on different processors.
- This type of processing is known as multiprogramming or multiprocessing. Multiple processors further divided in two types.

    i. Asymmetric Multiprocessing Systems (AMS)

    ii. Symmetric Multiprocessing Systems (SYS)

i. Asymmetric multiprocessing systems

- The multiprocessing system, in which each processor is assigned a specific task, is known as Asymmetric Multiprocessing Systems.
- In this system there exists master slave relationship like one processor defined as master and others are slave.
- The master processor controls the system and other processor executes predefined tasks. The master processor also defined the task of slave processors.

ii.Symmetric multiprocessing system

- The multiprocessing system in each processor performs all types of task within the operating system.
- All processors are peers and no master slave relationship exits. In SMP systems many programs can run simultaneously.
- But I/O must control to ensure that data reach the appropriate processor because all the processor shares the same memory.

3. Clustered systems

- Clustered systems are another form of multiprocessor system. This system also contains multiple processors but it differs from multiprocessor system.
- The clustered system is composed of multiple individual systems that connected together.
- In clustered system, also individual systems or computers share the same storage and liked to gather via local area network.
- A special type of software is known as cluster to control the node the systems.

Operating System properties:

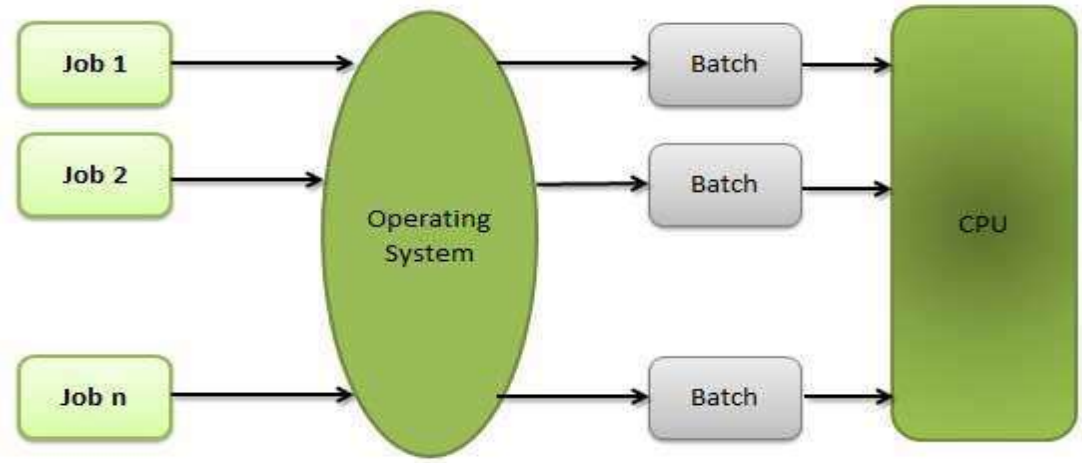
Operating systems are there from the very first computer generation. Operating systems keep evolving over the period of time. Following are few of the important types of operating system which are most commonly used.

Batch processing:

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

- OS defines a job which has predefined sequence of commands, programs and data as a single unit.
- OS keeps a number of jobs in memory and executes them without any manual information.
- Jobs are processed in the order of submission i.e. first come first served fashion.

- When 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.



Advantages

1. Batch processing takes much of the work of the operator to the computer.
2. Increased performance as a new job gets started as soon as the previous job finished without any manual intervention.

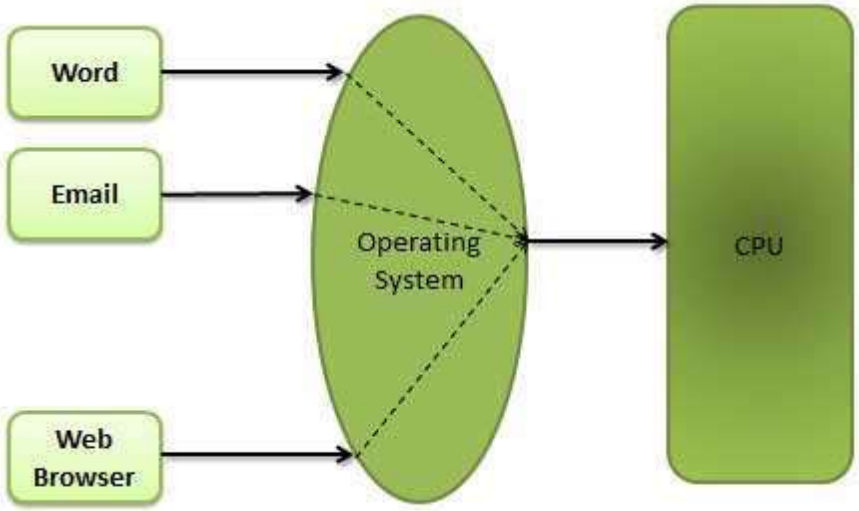
Disadvantages

1. Difficult to debug program.
2. A job could enter an infinite loop.
3. Due to lack of protection scheme, one batch job can affect pending jobs.

Multitasking:

Multitasking refers to term where multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running. Operating system does the following activities related to multitasking.

- The user gives instructions to the operating system or to a program directly, and receives an immediate response.
- Operating System handles multitasking in the way that it can handle multiple operations / executes multiple programs at a time.
- Multitasking Operating Systems are also known as Time-sharing systems.
- These Operating Systems were developed to provide interactive use of a computer system at a reasonable cost.
- A time-shared operating system uses concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU. Each user has at least one separate program in memory.

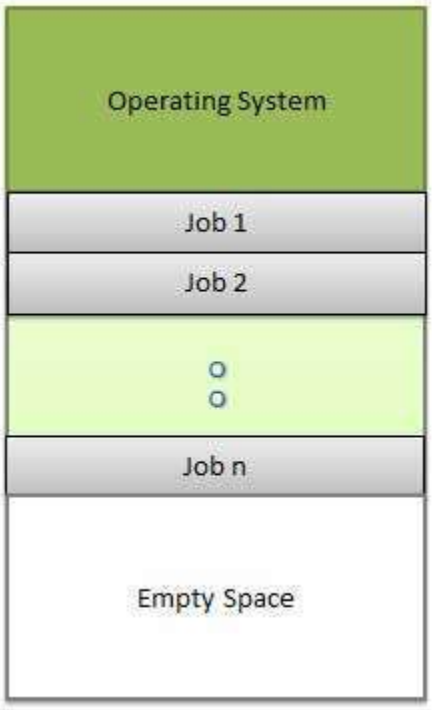


- A program that is loaded into memory and is executing is commonly referred to as a process.
- When a process executes, it typically executes for only a very short time before it either finishes or needs to perform I/O.
- Since interactive I/O typically runs at people speeds, it may take a long time to complete. During this time a CPU can be utilized by another process.
- Operating system allows the users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.
- As the system switches CPU rapidly from one user/program to the next, each user is given the impression that he/she has his/her own CPU, whereas actually one CPU is being shared among many users.

Multiprogramming:

When two or more programs are residing in memory at the same time, then sharing the processor is referred to the multiprogramming. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Following figure shows the memory layout for a multiprogramming system.



Operating system does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the job in the memory.
- Multiprogramming operating system monitors the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle unless there are no jobs

Advantages

1. High and efficient CPU utilization.
2. User feels that many programs are allotted CPU almost simultaneously.

Disadvantages

1. CPU scheduling is required.
2. To accommodate many jobs in memory, memory management is required.

Distributed Environment

Distributed environment refers to multiple independent CPUs or processors in a computer system. Operating system does the following activities related to distributed environment.

- OS Distributes computation logics among several physical processors.
- The processors do not share memory or a clock.
- Instead, each processor has its own local memory.

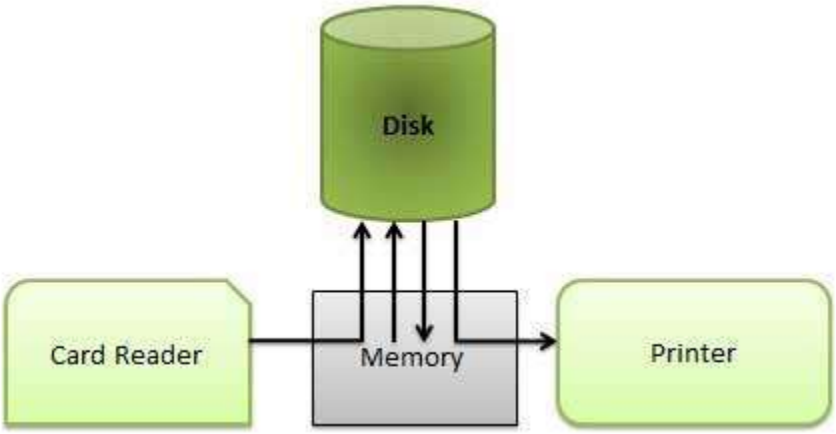
- OS manages the communications between the processors. They communicate with each other through various communication lines.

Spooling:

Spooling is an acronym for simultaneous peripheral operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices.

Operating system does the following activities related to distributed environment.

- OS handles I/O device data spooling as devices have different data access rates.
- OS maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.
- OS maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashion. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task.



Advantages

1. The spooling operation uses a disk as a very large buffer.
2. Spooling is capable of overlapping I/O operation for one job with processor operations for another job.

Operating System services

- An Operating System provides services to both the users and to the programs.
- It provides programs, an environment to execute. It provides users, services to execute the programs in a convenient manner.

Following are few common services provided by operating systems.

- User interface
- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

User interface:

User interface is used to interact with the computer to perform various tasks. User gives commands to computer and enters the data into computer. These are examples of user interfacing. The operating system plays the main role for interfacing between user and computer.

The input devices like keyboard and mouse are commonly used for giving commands to computer.

Types of Operating Systems

Based on the user interface, there are two types of operating systems.

1. Graphical User Interface Operating System
2. Command Line Operating System

1. Graphical User Interface Operating System:

- Graphical User Interface (GUI) operating system presents commands in graphical form.
- For example, application programs, commands, disk drives, files etc. are presented in the form of icons.
- Usually a command is given to the computer by clicking with mouse on the icon. GUI also provides menus, buttons and other graphical objects to the user to perform different tasks.
- GUI is very easy to interact with the computer.

Example

Examples of GUI operating systems are Windows, Linux, and Solaris. Today Windows is commonly used in PCs. In Windows, mouse is used as input device.

Features of Graphical user interface OS:

Interfacing

It provides commands in graphical form on the computer screen. The user gives commands to computer by checking with mouse on the icon. The users have not to memorize commands. Usually mouse is used for interfacing with computer.

Control

Although a GUI offers a better control of a file system and computer resources but often users have to use command line to complete a specific tasks.

Ease

It is easy to learn and use.

Multitasking

GUI provides facility to open multiple programs each in a separate window. So it enables a user to view, and to manipulate multiple things at a time on computer screen.

Speed

A GUI is easier to use. However, it is slower to perform different tasks.

Scripting

Although a GUI enables a user to create shortcuts, or other similar actions to complete a task. However, GUI does not provide the facility of scripting a sequence of commands to perform a task.

2. Command Line Operating System:

- A command line operating system provides a command prompt on the computer screen.
- The commands are given to the computer by typing on the keyboard. The commands are typed according to the predefined format.
- The users have to memorize commands and rules of writing these commands. It is not an easy way to interface with the computer.

Example

Examples of Command line operating systems are DOS (Disk Operating System), and UNIX etc**.**

Features of Command Line OS:

Interfacing

It provides commands prompt on the computer screen. The user gives commands to computer by typing on the keyboard. The users have to memorize commands and rules of writing these commands. Usually keyboard is used for interfacing with computer.

Control

It provides full access to computer resources.

Ease

It is difficult to learn and use.

Multitasking

Although many command line operating systems allow multitasking, but it is difficult in these operating systems to view multiple things at a time on computer screen.

Speed

The command line interface is faster than GUI to perform different tasks.

Scripting

A command line interface enables a user to easily script a sequence of commands to perform a tasks.

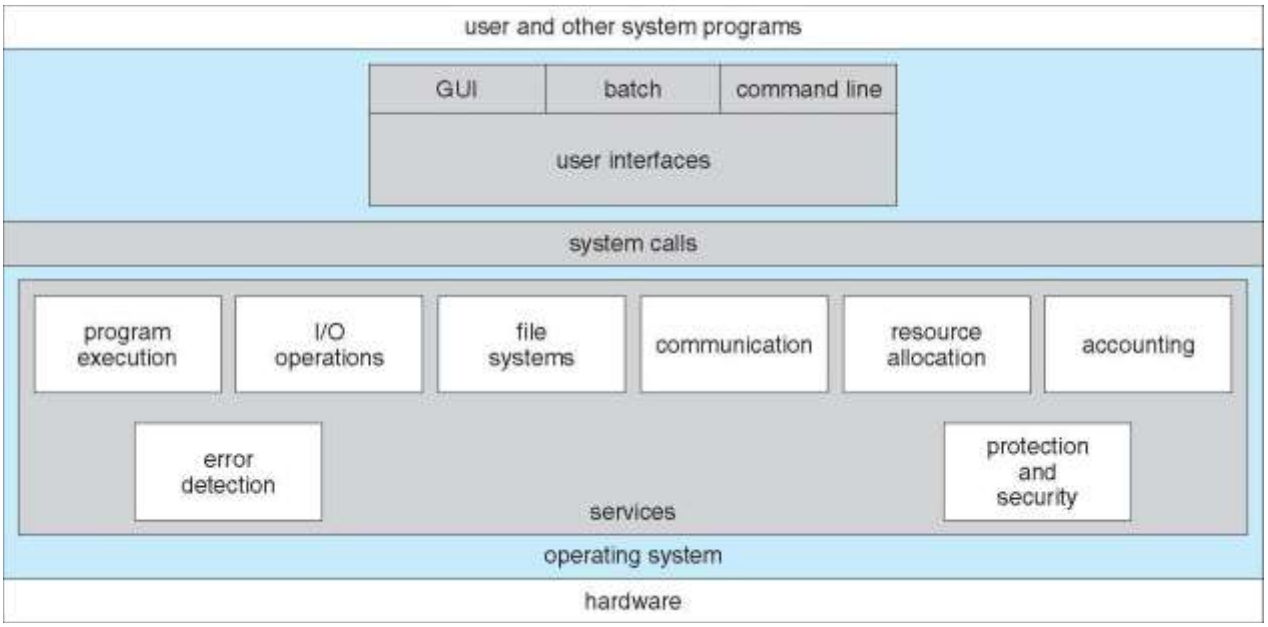


Figure 3: a view of operating system services

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|(indicating error).

I/O Operation:

- A running program may require I/O, which may involve a file or an I/O device.
- For specific devices, special functions may be desired (such as recording to a CD or DVD drive or banking a display screen).
- For efficiency and protection, users usually cannot control I/O devices directly.

File system manipulation:

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.

Following are the major activities of an operating system with respect to file management.

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

Communication:

- In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, operating system manages communications between processes. Multiple processes with one another through communication lines in the network.

Following are the major activities of an operating system with respect to communication.

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

Error Detection:

- Error can occur anytime and anywhere. Error may occur in CPU, in I/O devices or in the memory hardware.

Following are the major activities of an operating system with respect to error handling.

- OS constantly remains aware of possible errors.
- OS takes the appropriate action to ensure correct and consistent computing.

Resource allocation:

In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job.

Following are the major activities of an operating system with respect to resource management.

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

Protection and security:

- Protection refers to mechanism or a way to control the access of programs, processes, or users to the resources defined by computer systems.

Following are the major activities of an operating system with respect to protection.

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

System calls

"System calls provide an interface to the services made available by an operating system."

These calls are generally available as routine written in C and C++,

An example to illustrate how system calls are used:

Writing a simple program to read data from one file and copy them to another file -

a) First input required is names of two files – input file and output file. Names can be specified in many ways-

- One approach is for the program to ask the user for the names of two files.
- In an interactive system, this approach will require a sequence of system calls, to write a prompting message on screen and then read from the keyboard the characters that define the two files.
- On mouse based and icon based systems, a menu of file names is displayed in a window where the user can use the mouse to select the source names and a window can be opened for the destination name to be specified.

b) Once the two file names are obtained, program must open the input file and create the output file. Each of these operations requires another system call.

- When the program tries to open input file, no file of that name may exist or file is protected against access. Program prints a message on console and terminates abnormally.
- If input file exists, we must create a new output file. If the output file with the same name exists, the situation caused the program to abort or delete the existing file and create a new one. Another option is to ask the user (via a sequence of system calls) whether to replace the existing file or to abort the program.
- When both files are set up, a loop reads from the input file and writes to the output file (system calls respectively). Each read and write must return status information regarding various possible error conditions. After entire file is copied, program closes both files, write a message to the console or window and finally terminate normally.
- Application developers design programs according to application programming interface (API). API specifies set of functions that are available to an application programmer.

- Three of the most common API's available to application programmers are the Win32API for Windows Systems; POSIX API for POSIX based systems (which include all versions of UNIX, Linux and Mac OS X) and Java API for designing programs that run on Java virtual machine.

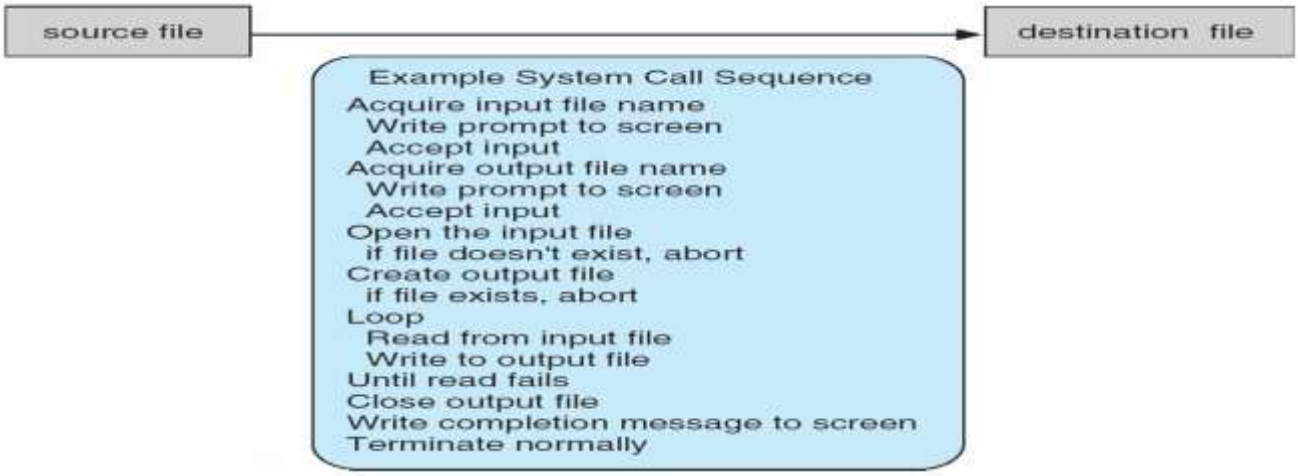


Figure 5: Example of how system calls are used

Types of System Calls

System calls can be grouped roughly into five major categories:

1. Process control
2. File manipulation
3. Device manipulation
4. Information maintenance
5. Communication

We discuss briefly the types of system calls that may be provided by an operating system.

1. Process Control

- A running program needs to be able to halt its execution either normally (end) or abnormally (abort). If a system call is made to terminate the currently running program abnormally, or if the program runs into a problem and causes an error trap, a dump of memory is sometimes taken and an error message generated and it can be examined with a debugger.
- Under either normal or abnormal circumstances, the operating system must transfer control to the invoking command interpreter. The command interpreter then reads the next command.

Process control provides system calls

– end, abort

– load, execute

– create process, terminate process

– get process attributes, set process attributes

– wait for time

– wait event, signal event

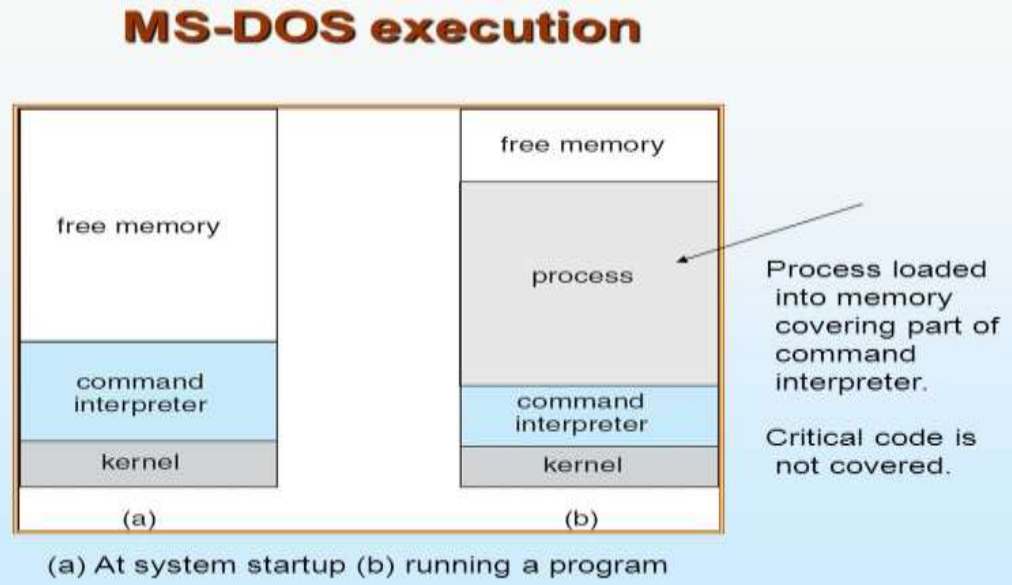
– allocate and free memory

- There are so many variations in process and job control that we next use two examples-one involving a single-tasking system and the other a multitasking system -to clarify these concepts.

MS-DOS Execution:

❖ The MS-DOS operating system is an example of a single-tasking system.

- It has a command interpreter that is invoked when the computer is started (Figure (a)). Because MS-DOS is single-tasking, it uses a simple method to run a program and does not create a new process.



(a) At system startup (b) running a program

- It loads the program into memory, writing over most of itself to give the program as much memory as possible (Figure (b)).
- Next, it sets the instruction pointer to the first instruction of the program. The program then runs, and either an error cause a trap, or the program executes a system call to terminate.
- In either case, the error code is saved in the system memory for later use. Then the command interpreter makes the previous error code available to the user or to the next program.

FreeBSD Running Multiple Programs:

- FreeBSD is a free Unix-like operating system descended from Research Unix via the Berkeley Software Distribution (BSD).
- FreeBSD is an example of a multitasking system. When a user logs on to the system, the shell of the user's choice is run.
- This shell is similar to the MS-DOS shell in that it accepts commands and executes programs that the user requests.
- However, since FreeBSD is a multitasking system, the command interpreter may continue running while another program is executed (Figure C).

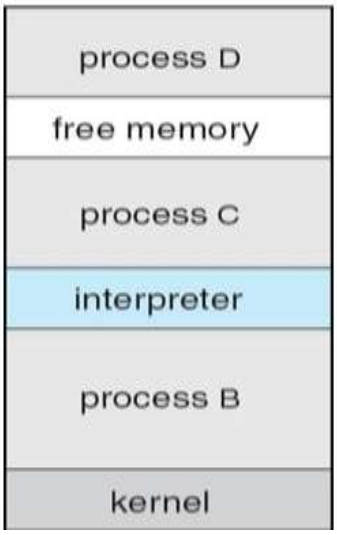


Figure C FreeBSD running multiple programs.

- To start a new process, the shell executes a fork ( ) system call. Then, the selected program is loaded into memory via an exec() system call, and the program is executed.
- Depending on the way the command was issued, the shell then either waits for the process to finish or runs the process "in the background." In the latter case, the shell immediately requests another command.
- When a process is running in the background, it cannot receive input directly from the keyboard, because the shell is using this resource.
- I/O is therefore done through files or through a GUI interface. Meanwhile, the user is free to ask the shell to run other programs, to monitor the progress of the running process, to change that program's priority, and so on.
- When the process is done, it executes an exit ( ) system calls to terminate, returning to the invoking process a status code of 0 or a nonzero error code. This status or error code is then available to the shell or other programs.

2. File Management

- We first need to be able to create and delete files. Either system call requires the name of the file and perhaps some of the file's attributes.
- Once the file is created, we need to open it and to use it. We may also read, write, or reposition (rewinding or skipping to the end of the file, for example).
- Finally, we need to close the file, indicating that we are no longer using it.
- We may need these same sets of operations for directories if we have a directory structure for organizing files in the file system.
- File attributes include the file name, file type, protection codes, accounting information, and so on. At least two system calls, get file attribute and set file attribute, are required for this function. Some operating systems provide many more calls, such as calls for file move and copy.

File management provides system calls

– create file, delete file

– open, close file

– read, write, reposition

– get and set file attributes

3. Device Management

- A process may need several resources to execute-main memory, disk drives, access to files, and so on.
- If the resources are available, they can be granted, and control can be returned to the user process. Otherwise, the process will have to wait until sufficient resources are available.
- The various resources controlled by the operating system can be thought of as devices. Some of these devices are physical devices (for example, disk drives), while others can be thought of as abstract or virtual devices (for example, files).
- A system with multiple users may require us to first request the device, to ensure exclusive use of it. After we are finished with the device, we release it.
- These functions are similar to the open and close system calls for files

Device management provides system calls

– request device, release device

– read, write, reposition

– get device attributes, set device attributes

4. Information Maintenance

- Many system calls exist simply for the purpose of transferring information between the user program and the operating system.
- For example, most systems have a system call to return the current time and date. Other system calls may return information about the system, such as the number of current users, the version number of the operating system, the amount of free memory or disk space, and so on.

Information maintenance provides system calls

– get time or date, set time or date

– get system data, set system data

– get and set process, file, or device attributes

5. Communication

- There are two common models of interprocess communications:
  - i). Message passing model
  - ii). shared –memory model.
- **In the message-passing model**, the communicating processes exchange messages with one another to transfer information.
- Messages can be exchanged between the processes either directly or indirectly through a common mailbox.
- Before communications can take place, a connection must be opened. The name of the other communicator must be known, be it another process on the same system or a process on another computer connected by a communications network.
- Each computer in a network has a host name by which it is commonly known. A host also has a network identifier, such as an IP address.
- Similarly, each process has a process name, and this name is translated into an identifier by which the operating system can refer to the process. The gethostid and getprocessid system calls do this translation.
- The identifiers are then passed to the general purpose open and close calls provided by the file system or to specific open connection and close connection system calls, depending on the system's model of communication

Communications provides system calls

– create, delete communication connection

– send, receive messages

– transfer status information

– attach and detach remote devices

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()

System programs:

System programs, also known as **system utilities**, provide a convenient environment for program development and execution.

Some of them are simply user interfaces to system calls; others are considerably more complex. They can be divided into these categories:

- **File management**. These programs create, delete, copy, rename, print, dump, list, and generally manipulate files and directories.
- **Status information**. Some programs simply ask the system for the date, time, amount of available memory or disk space, number of users, or similar status information. Others are more complex, providing detailed performance, logging, and debugging information. Typically, these programs format and print the output to the terminal or other output devices or files or display it

in a window of the GUI. Some systems also support a registry, which is used to store and retrieve configuration information.

- **File modification**. Several text editors may be available to create and modify the content of files stored on disk or other storage devices. There may also be special commands to search contents of files or perform transformations of the text.
- **Programming-language support**. Compilers, assemblers, debuggers, and interpreters for common programming languages (such as C, C++, Java, Visual Basic, and PERL) are often provided to the user with the operating system.
- **Program loading and execution**. Once a program is assembled or compiled, it must be loaded into memory to be executed. The system may provide absolute loaders, relocatable loaders, linkage editors, and overlay loaders. Debugging systems for either higher-level languages or machine language are needed as well.
- **Communications.** These programs provide the mechanism for creating virtual connections among processes, users, and computer systems. They allow users to send rnessages to one another's screens, to browse Web pages, to send electronic-mail messages, to log in remotely, or to transfer files from one machine to another.

In addition to systems programs, most operating systems are supplied with programs that are useful in solving common problems or performing common operations. Such as application programs include web browsers, word processors and text editors, spreadsheets, database systems, compilers, plotting and statistical -analysis packages, and games.

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.

1. Simple Structure

- Many commercial 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. It was originally designed and implemented by a few people who had no idea that it would become so popular. It was written to provide the most functionality in the least space, so it was not divided into modules carefully. Figure A shows its structure
- Another example of limited structuring is the original UNIX operating system. Like MS-DOS, UNIX initially was limited by hardware functionality

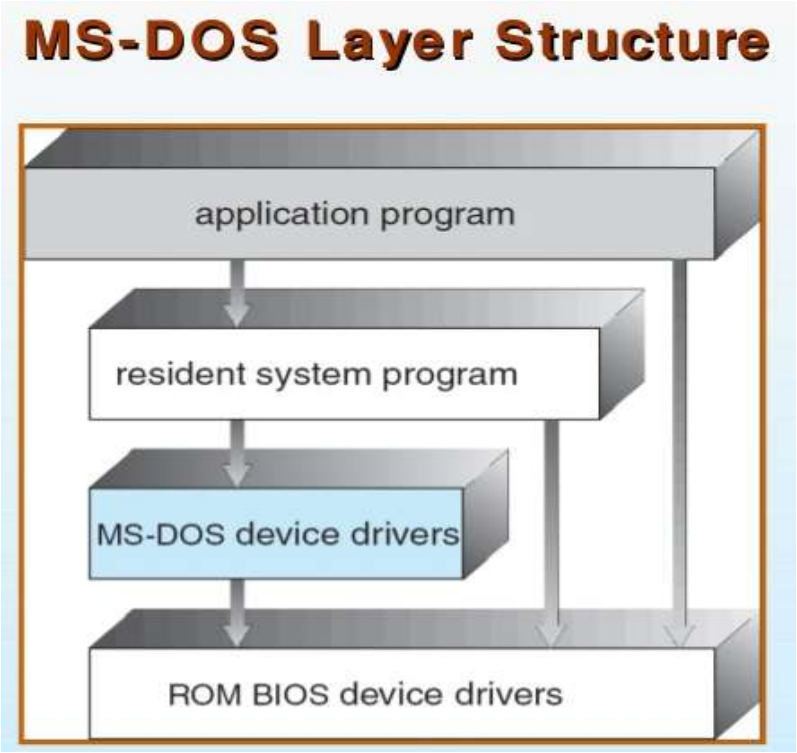


Fig A: MS-DOS layer structure.

- 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, which have been added and expanded over the years as UNIX has evolved.
- We can view **the traditional UNIX operating system** as being layered, as shown in Figure B.

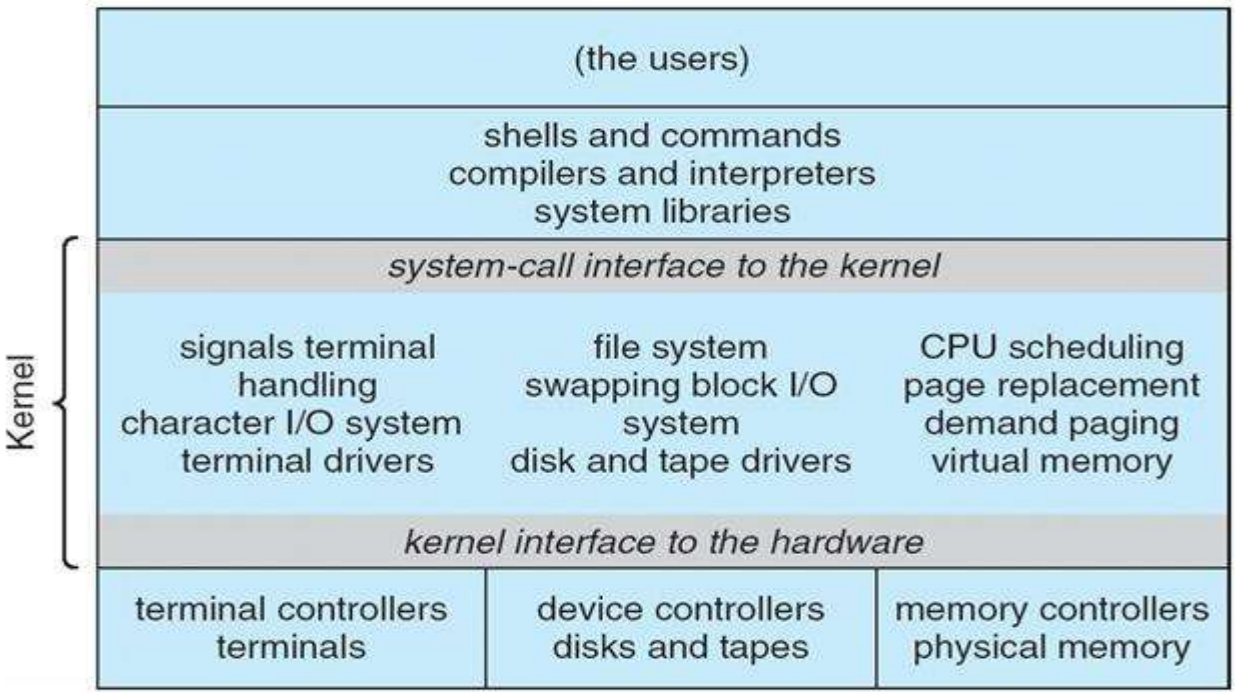


Figure B: Traditional UNIX system structure.

- Everything below the system-call interface and above the physical hardware is the kernel.

- The kernel provides the file system, CPU scheduling, memory management, and other operating-system functions through system calls.

2. 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 (layer N) is the user interface. This layering structure is depicted in Figure D.

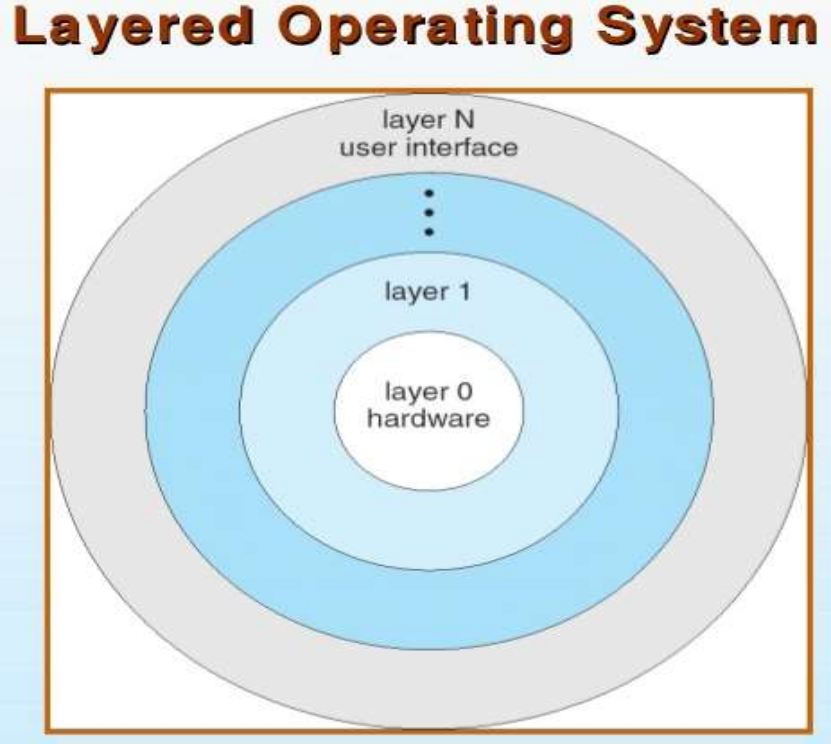


Figure D: A layered operating system.

- An operating-system layer is an implementation of an abstract object made up of data and the operations that can manipulate those data.
- A typical operating-system layer-say, layer *M* - consists of data structures and a set of routines that can be invoked by higher-level layers. Layer *M*, in turn, can invoke operations on lower-level layers.
- The main advantage of the layered approach is simplicity of construction and debugging. The layers are selected so that each uses functions (operations) and services of only lower-level layers. This approach simplifies debugging and system verification.
- Each layer is implemented with only those operations provided by lower level layers. A layer does not need to know how these operations are implemented; it needs to know only what these operations do. Hence, each layer hides the existence of certain data structures, operations, and hardware from higher-level layers.

- The major difficulty with the layered approach involves appropriately defining the various layers. Because a layer can use only lower-level layers, careful planning is necessary.
- **For example,** the device driver for the backing store (disk space used by virtual-memory algorithms) must be at a lower level than the memory-management routines, because memory management requires the ability to use the backing store.

3. Micro kernels

- We know that as UNIX expanded, the kernel became large and difficult to manage. In the mid-1980s, developed an operating system called **Mach** that modularized the kernel using the micro kernel approach.
- This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel. There is little consensus regarding which services should remain in the kernel and which should be implemented in user space.
- Typically, however, micro kernels provide minimal process and memory management, in addition to a communication facility.
- The main function of the micro kernel is to provide a communication facility between the client program and the various services that are also running in user space. Communication is provided by message passing.
- **For example,** if the client program wishes to access a file, it must interact with the file server. The client program and service never interact directly. Rather, they communicate indirectly by exchanging messages with the micro kernel.
- One benefit of the microkernel approach is ease of extending the operating system. All new services are added to user space and consequently do not require modification of the kernel.

4. Modules

- The best current methodology for operating-system design involves using object-oriented programming techniques to create a modular kernel. Here, the kernel has a set of core components and links in additional services either during boot time or during run time.
- Such a strategy uses dynamically loadable modules and is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X.
- For example, the Solaris operating system structure, shown in Figure E, is organized around a core kernel with seven types of loadable kernel modules:
  - ✓ Scheduling classes
  - ✓ File systems
  - ✓ Loadable system calls

- ✓ Executable formats
- ✓ STREAMS modules
- ✓ Miscellaneous
- ✓ Device and bus drivers

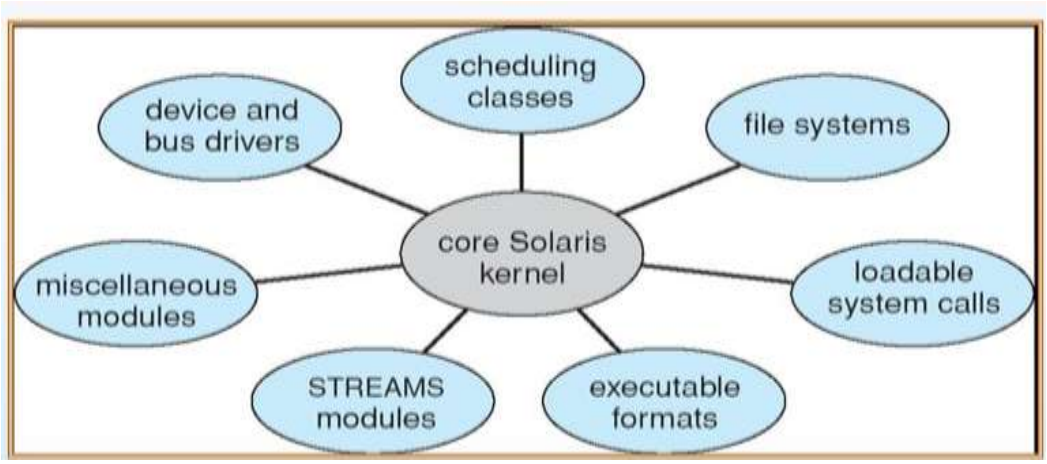


Figure E: Solaris loadable modules

- Such a design allows the kernel to provide core services yet also allows certain features to be implemented dynamically.
- For example, device and bus drivers for specific hardware can be added to the kernel, and support for different file systems can be added as loadable modules.

Operating System Design and implementation

Design and implementation of operating system not solvable, no complete solutions to such problems, but some approaches have proven successful

a) 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 shared, single user, multiuser, distributed, real time, or general purpose.
- Beyond this highest design level, the requirements may be much harder to specify. The requirements can, however, be divided into two basic groups: **user goals and system goals.**
- Users desire certain obvious properties in a system.
- The system should be convenient to use, easy to learn and to use, reliable, safe, and fast. Of course, these specifications are not particularly useful in the system design, since there is no general agreement on how to achieve them.
- The system should be easy to design, implement, and maintain; and it should be flexible, reliable, error free, and efficient.

b) <u>Mechanisms and Policies:</u> One important principle is the separation of **policy** from **mechanism.**

- Mechanisms determine→ how to do something,
- policies decide→ what will be done

– The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

c) <u>Implementation</u>

Once an operating system is designed, it must be implemented. Traditionally, operating systems have been written in assembly language. Now, however, they are most commonly written in higher-level languages such as C or C++.

- The advantages of using a higher-level language, or at least a systems implementation language, for implementing operating systems are the same as those accrued when the language is used for application programs: the code can be written faster, is more compact, and is easier to understand and debug,
- Finally, an operating system is far easier to port-to move to some other hardware-if it is written in a higher-level language. For example, MS-DOS was written in Intel 8088 assembly language. Consequently, it runs natively only on the Intel X86 family of CPUs.
- The Linux operating system, in contrast, is written mostly in C and is available natively on a number of different CPUs, including Intel X86, Sun SPARC, and IBMPowerPC.
- The only possible disadvantages of implementing an operating system in a higher-level language are reduced speed and increased storage requirements.
- This however is no longer a major issue in today's systems. Although an expert assembly-language programmer can produce efficient small routines, for large programs a modern compiler can perform complex analysis and apply sophisticated optimizations that produce excellent code.
- Modern processors have deep pipelining and multiple functional units that can handle the details of complex dependencies much more easily than can the human mind.

Virtual machine

- The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.
- By using CPU scheduling and virtual-memory techniques, an operating system host can create the illusion that a process has its own processor with its own (virtual) memory. The virtual machine provides an interface that is identical to the underlying hardware.

- Each guest process is provided with a (virtual) copy of the underlying computer (Figure F). Usually, the guest process is in fact an operating system, and that is how a single physical machine can run multiple operating systems concurrently, each in its own virtual machine.

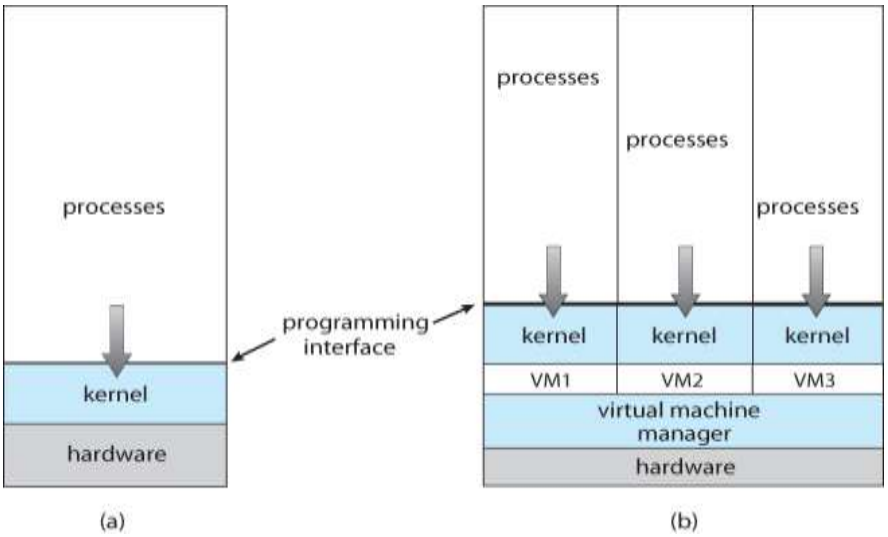


Figure F: System models. (a) Nonvirtual machine. (b) Virtual machine.

- Virtual machines first appeared commercially on IBM mainframes via the VM operating system.
- A major difficulty with the VM virtual machine approach involved disk systems. Suppose that the physical machine had three disk drives but wanted to support seven virtual machines. Clearly, it could not allocate a disk drive to each virtual machine, because the virtual machine software itself needed substantial disk space to provide virtual memory and spooling.
- The solution was to provide virtual disks-termed mini disks in IBM's VM operating system -that are identical in all respects except size. The system implemented each minidisk by allocating as many tracks on the physical disks as the minidisk needed.
- Once these virtual machines were created, users could run any of the operating systems or software packages that were available on the underlying machine.

Advantages:

- One important advantage is that the host system is protected from the virtual machines, just as the virtual machines are protected from each other.
- A virus inside a guest operating system might damage that operating system but is unlikely to affect the host or the other guests. Because each virtual machine is completely isolated from all other virtual machines, there are no protection problems. At the same time, however, there is no direct sharing of resources.
- Two approaches to provide sharing have been implemented. First, it is possible to share a file-system volume and thus to share files. Second, it is possible to define a network of virtual machines, each of which can send information over the virtual communications network.

The Java Virtual Machine

- Java is a popular object-oriented programming language introduced by Sun Microsystems in 1995. In addition to a language specification and a large API library, Java also provides a specification for a Java virtual machine-or JVM.
- Java objects are specified with the class construct; a Java program consists of one or more classes. For each Java class, the compiler produces an architecture-neutral byte code output (.class) file that will run on any implementation of the JVM.

The JVM is a specification for an abstract computer. It consists of---

- ❖ class loader
- ❖ class verifier
- ❖ runtime interpreter

- ➢ **Class loader** and a Java interpreter that executes the architecture-neutral byte codes, as diagrammed in Figure G. The class loader loads the compiled **.**class files from both the Java program and the Java API for execution by the Java interpreter.
- ➢ After a class is loaded, the **verifier** checks that the **.**class file is valid Java byte code and does not overflow or underflow the stack. It also ensures that the byte code does not perform pointer arithmetic, which could provide illegal memory access.
- ➢ If the class passes verification, it is run by the Java interpreter.
- ➢ The **JVM** may be implemented in software on top of a host operating system, such as Windows, Linux, or Mac OS X, or as part of a Web browser. Alternatively, the JVM may be implemented in hardware on a chip specifically designed to nm Java programs. If the JVM is implemented in. software, the Java interpreter interprets the byte code operations one at a time.
- ➢ A faster software technique is to use a just-in-time (JIT) compiler

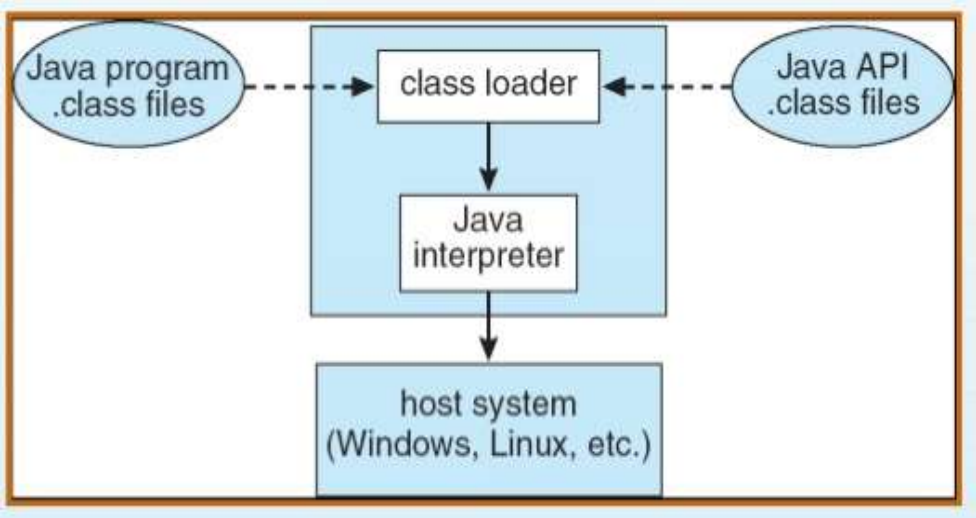


Figure G: The Java virtual machine.