Operating System Overview

UNIT 1

Introduction

- An **operating system** (OS) is a set of programs that control the execution of application programs and act as an intermediary between a user of a computer and the computer hardware.
- OS is software that manages the computer hardware as well as providing an environment for application programs to run.
- Examples of OS are: Windows, Windows/NT, OS/2 and MacOS.

Introduction:

- Computer Software can roughly be divided into two types:

  a). Application Software: Which perform the actual work the user wants.

  b). System Software: Which manage the operation of the computer itself.

- The most fundamental system program is the operating system, whose job is to control all the computer's resources and provide a base upon which the application program can be written.
- Operating system acts as an intermediary between a user of a computer and the computer hardware.

Introduction:

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.

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

Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Coordination between other software and users

Memory Management

- Memory management refers to management of Primary Memory or Main Memory.
- Main memory provides a fast storage that can be accessed directly by the CPU.
- For a program to be executed, it must be in the main memory.

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

Processor Management

- In multiprogramming environment, the OS decides which process gets the processor when and for how much time.
- This function is called **process scheduling**.

An Operating System does the following activities for processor management:

- Keeps tracks of processor and status of process.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

- An Operating System manages device communication via their respective drivers.

It does the following activities for device management:

- Keeps tracks of all devices.
- Decides which process gets the device when and for how much time.
- Allocates the device in the most efficient way.
- De-allocates devices.

File Management

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

An Operating System does the following activities for file management:

- Keeps track of information, location, uses, status etc.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities

Following are some of the important activities that an Operating System performs:

- **Security** -- By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- **Job accounting** -- Keeping track of time and resources used by various jobs .
- **Error detecting** -- Production of error messages, and other debugging and error detecting aids.
- **Coordination between other software and users** -- Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

Computer System

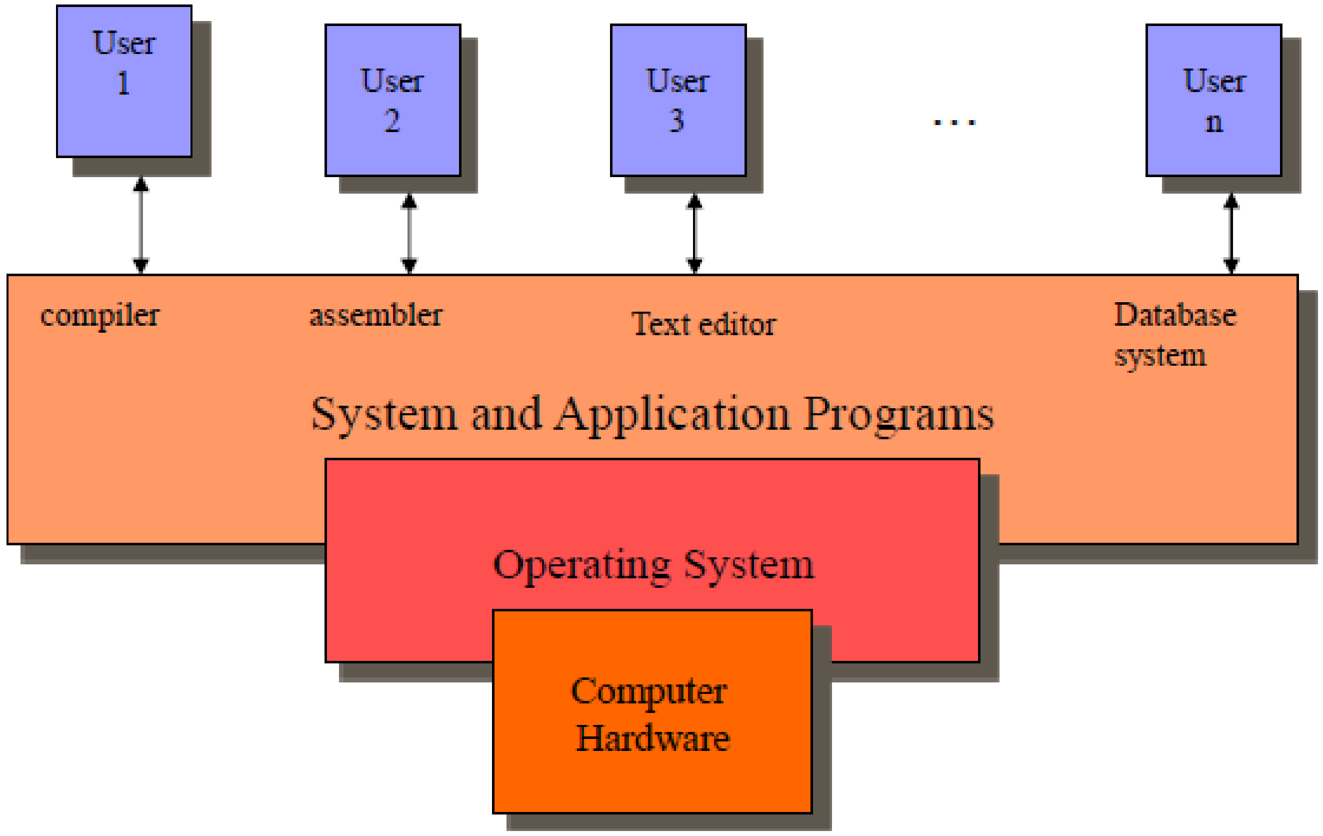


Figure 1: computer system

Computer System

A computer system can be divided into four components: the hardware, the operating system, the application programs and the users. The abstract view of system components is shown in figure1.

1. **Hardware**: such as CPU, memory and I/O devices.
2. **Operating system**: provides the means of proper use of the hardware in the operations of the computer system, it is similar to government.
3. **Application programs**: solve the computing problems of the user, such as : compilers, database systems and web browsers.
4. **Users:** peoples, machine, or other computer.

Computer System Organization

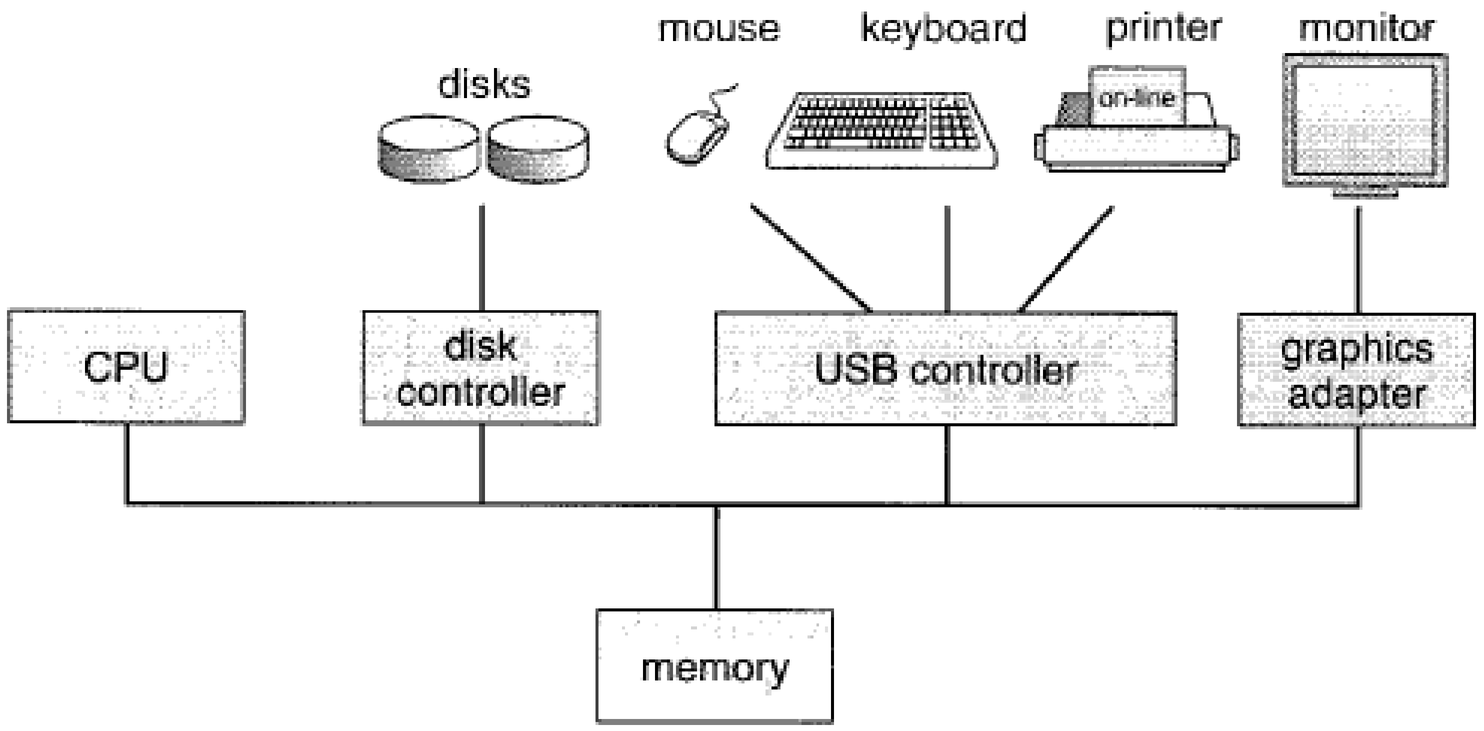
1. Computer-System Operation

- 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). Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, and video displays). The CPU and the device controllers can execute concurrently, competing for memory cycles. To ensure orderly access to the shared memory, a memory controller is synchronizing access to the memory.

Computer System Organization..

1. Computer-System Operation …

- 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 in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM),known by the general term **firmware**, within the computer hardware. It initializes all aspects of the system, from CPU registers to device controllers to memory contents. The bootstrap program must know how to load the operating system and to start executing that system. To accomplish this goal, the bootstrap program must locate and load into memory the operating system kernel. The operating system then starts executing the first process, such as "init," and waits for some event to occur.



A modern computer system.

Figure 2

Computer System Organization

2. Storage Structure

- Computer programs must be in main memory (also called RAM) to be executed. Main memory is the only large storage area that the processor can access directly. It forms an array of memory words. Each word has its own address. Interaction is achieved through a sequence of load or store instructions to specific memory addresses. The load instruction moves a word from main memory to an internal register within the CPU, whereas the store instruction moves the content of a register to main memory.
- The **instruction-execution cycle** includes:
- 1) Fetches an instruction from memory and stores that instruction in the instruction register. And increment the PC register.
- 2) Decode the instruction and may cause operands to be fetched from memory and stored in some internal register.
- 3) Execute the instruction and store the result in memory.

Computer System Organization…

2. Storage Structure ….

The programs and data are not resided in main memory permanently for the following two reasons:

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

Thus, most computer systems provide secondary storage as an extension of main memory to hold large quantities of data permanently.

The wide variety of storage systems in a computer system can be organized in a hierarchy (figure 2). **The main differences among the various storage systems lie in speed, cost, size, and volatility**. The higher levels are expensive, but they are fast.

Computer System Organization…

2. Storage Structure ….

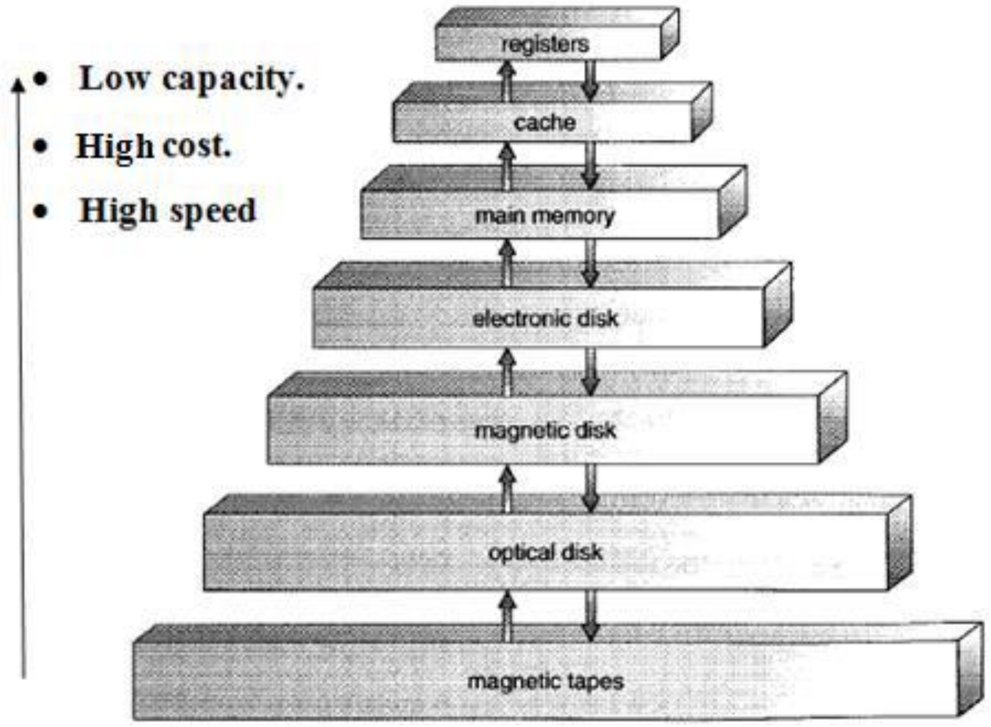


Figure 3: Storage device hierarchy

BASIC ELEMENTS and hardware review

- **Processor**
  - Processor controls the operation of computer and performs its data processing functions like arithmetic, logic and others.
- **Main Memory**
  - Main memory is also called as volatile memory, primary memory, real memory or temporary memory, because it stores data and programs temporarily during the processing time only.
- **Input/Output Modules**
  - Input/Output modules move data between the computer and its external environment like secondary memory, communications equipment and terminals etc.
- **System Inter Connection**
  - System inter connection provide some structure and mechanisms that provide for communication among processors, main memory and Input/Output modules

A Brief History of Operating Systems

Phase 0: No operating system: 1940-1955

- Computers were experimental equipment.
- Program in machine language.
- Use plugboards to direct computer.
- User sits at the console.
- Programs manually loaded via card decks.

Phase 1: 1955-1970

- Computers are expensive; people are cheap
- Make more efficient use of the computer: move the person away from the machine.
- Time of batch processing.
- OS becomes a *batch monitor:*
  - a program that loads a user's job, runs it, and then moves on to the next.
- More efficient use of hardware, but increasingly difficult to debug!

Phase 1 Technology

- Buffering and interrupt handling is done by OS.
- Spool(simultaneous peripheral operations online) jobs onto "high speed" drums (cards are slow)

Problems

- CPU Utilization is low (one job at a time).
- Short jobs wait if they get stuck behind longer jobs.

Solutions

- Multiprogramming: Many programs can share the system.
- Scheduling: Let short jobs finish quickly
- OS/360: first OS designed for a family of computers; one operating system designed to run from smallest to largest machines.

Phase 1 Disasters

- Operating systems didn't really work!
- OS/360 was introduced in 1963; worked in 1968.
- Systems were enormously complicated.
- They were written in assembly code.

Phase 2: 1970-1980

- Interactive timesharing: let many users use the same machine at once.
- Terminals are cheap: give everyone one

CTSS(compatible time sharing system):

- Developed at MIT.
- One of the first timesharing systems.
- Pioneered much of the work in scheduling.
- Motivated MULTICS.

MULTICS:

- Joint development by MIT, Bell Labs
- Building it was more difficult than expected.

Phase 2: UNIX

- Ken Thompson and Dennis Ritchie built a system.
- Originally in assembly language. Rewritten by Ritchie and Thompson in C.
- New idea: portable operating system!
- Universities obtained code for experimentation.
- UNIX becomes a commercial operating system.

Important ideas popularized by UNIX

- OS written in a high-level language.
- OS is portable across hardware platforms.

Phase 3: 1980-1990

- Computers are cheap; people are expensive.
- CP/M first personal computer operating system.
- Approached Bill Gates (Microsoft) to see if they could build one.
- Gates bought 86-DOS, and created MS-DOS.

Phase 3 Technologies

Personal computers

- The Apple II
- The IBM PC
- The Macintosh

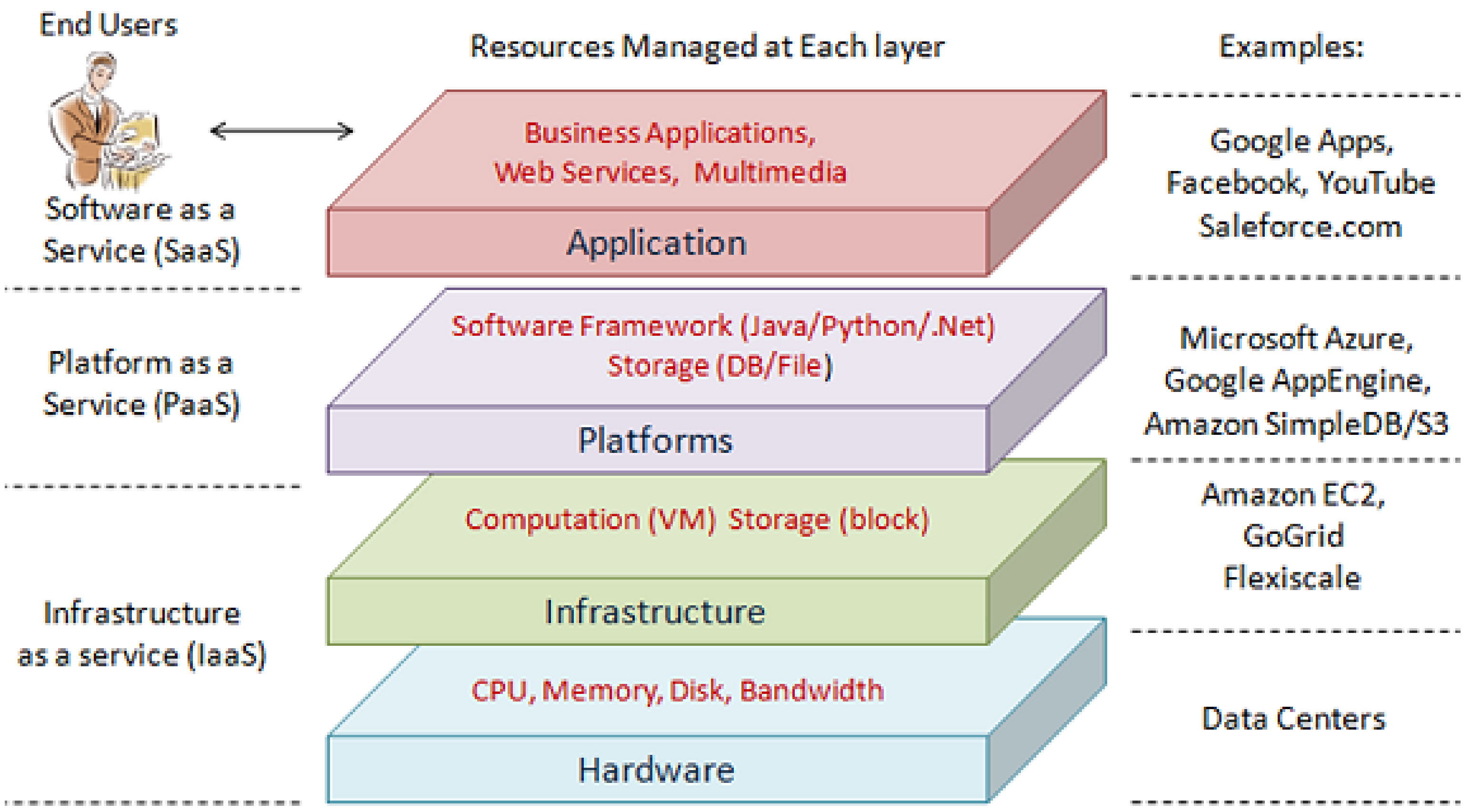
Business applications grow around the industry

- Word processors
- Spreadsheets
- Databases

Phase 4: Networked Systems (1990-200?)

- People want to share *data* not hardware.
- Networked applications.
  - The Web
  - Email
- Protection and multiprogramming less important for personal machines.
- Protection and multiprogramming more important for server machines.
- New network-based architectures:
  - Clusters (domino an application monitoring clusters)
  - Grids(volunteer computing projects, such as BOINC , use CPU scavenging)
  - Distributed operating systems(amoeba, mach, chorus)
  - Network operating systems(unix, windows NT)
  - Cloud (or is this a new generation?, cloud os, osV)

A layered model of cloud computing


End Users
Resources Managed at Each layer
Examples:
Software as a Service (SaaS)
Business Applications, Web Services, Multimedia
Application
Google Apps, Facebook, YouTube Saleforce.com
Platform as a Service (PaaS)
Software Framework (Java/Python/.Net) Storage (DB/File)
Platforms
Microsoft Azure, Google AppEngine, Amazon SimpleDB/S3
Infrastructure as a service (IaaS)
Computation (VM) Storage (block)
Infrastructure
Amazon EC2, GoGrid Flexiscale
CPU, Memory, Disk, Bandwidth
Hardware
Data Centers

Serial Processing:

- Early computers from 1940s to 1950s.
- This mode of operation is turned as serial processing ,reflecting the fact that users access the computer in series.
- The programmer interacted directly with the computer hardware.
- These machine are called bare machine as they don't have OS.
- Every computer system is programmed in its machine language.
- Uses Punch Card, paper tapes.

These system presented two major problems.

1. Scheduling
2. Set up time:

Serial Processing:..

Scheduling:

- A user may sign up for an hour but finishes his job in 45 minutes.
- This would result in wasted computer time, also the user might run into the problem not finish his job in allotted time.

Set up time:

A single program involves:

- ➢ Loading compiler and source program in memory
- ➢ Saving the compiled program (object code)
- ➢ Loading and linking together object program and common function
- Each of these steps involves the mounting or dismounting tapes, setting up punch cards.
- If an error occur, user had to go the beginning of the set up sequence.
- Thus, a considerable amount of time is spent in setting up the program to run.

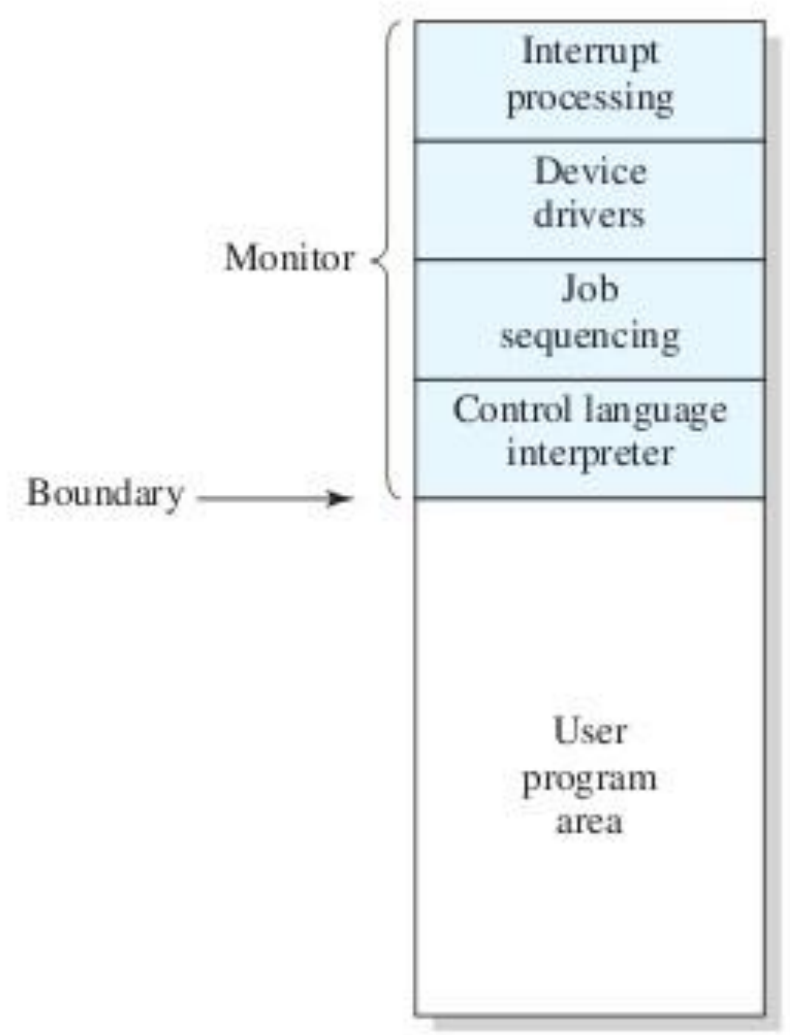
Simple Batch Processing:

- Early computers were very expensive, and therefore it was important to maximize processor utilization.
- The wasted time due to scheduling and setup time in Serial Processing was unacceptable.
- To improve utilization, the concept of a batch operating system was developed.
- Batch is defined as a group of jobs with similar needs.
- Computer executes each batch sequentially, processing all jobs of a batch considering them as a single process called batch processing.
- The central idea behind the simple batch-processing scheme is the use of a piece of software known as the **monitor**.

Simple Batch Processing:

- With this type of OS, the user no longer has direct access to the processor.
- Instead, the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device, for use by the monitor.
- Each program is constructed to branch back to the monitor when it completes processing, at which point the monitor automatically begins loading the next program.

Simple Batch Processing:



Simple Batch Processing:

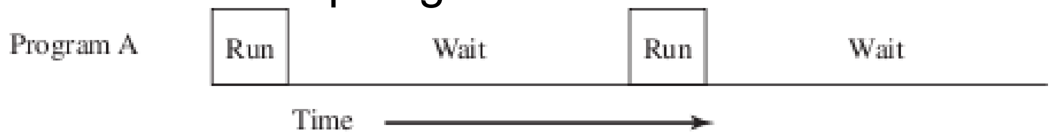
- With a batch operating system, processor time alternates between execution of user programs and execution of the monitor.

There have been two sacrifices:

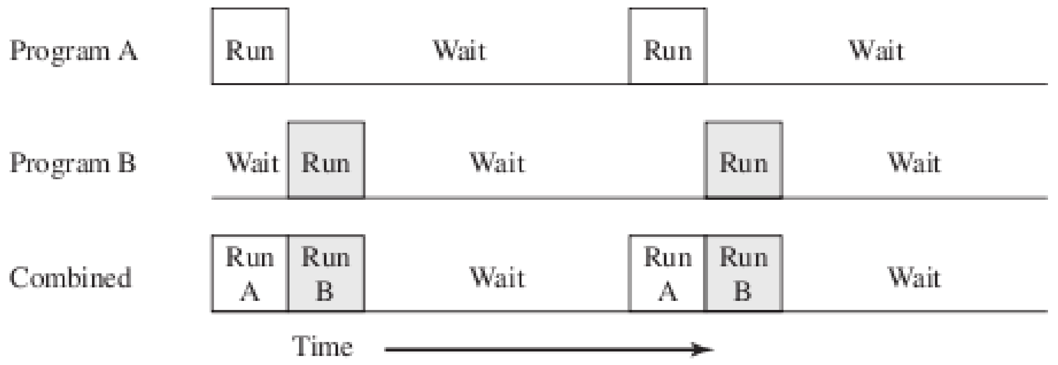
- Some main memory is now given over to the monitor and
- Some processor time is consumed by the monitor.
- Both of these are forms of overhead.

Multiprogrammed Batch System:

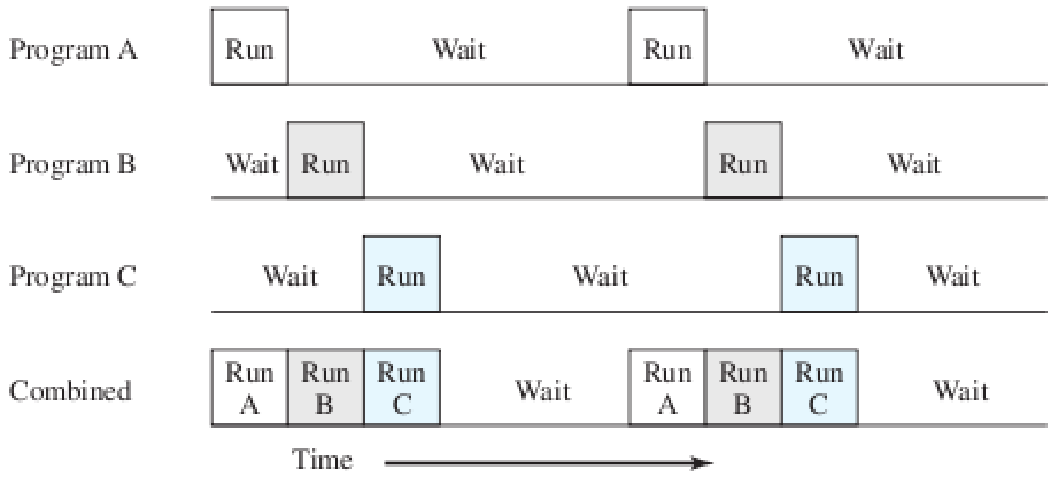
- A single program cannot keep either CPU or I/O devices busy at all times.
- Multiprogramming increases CPU utilization by organizing jobs in such a manner that CPU has always one job to execute.
- If computer is required to run several programs at the same time, the processor could be kept busy for the most of the time by switching its attention from one program to the next.
- Additionally I/O transfer could overlap the processor activity i.e, while one program is awaiting for an I/O transfer, another program can use the processor.
- So CPU never sits idle or if comes in idle state then after a very small time it is again busy. This is illustrated in fig below.
- **Multiprogramming** is first used in OS/360.



(a) Uniprogramming



(b) Multiprogramming with two programs



(c) Multiprogramming with three programs

Multitasking or Time Sharing System:

- Multiprogramming didn't provide the **user interaction** with the computer system.
- Time sharing or Multitasking is a logical extension of Multiprogramming that provides user interaction.
- There are more than one user interacting the system at the same time
- The switching of CPU between two users is so fast that it gives the impression to user that he/she is only working on the system but actually it is shared among different users.
- CPU bound is divided into different time slots depending upon the number of users using the system.
- just as multiprogramming allows the processor to handle multiple **batch jobs** at a time, multiprogramming can also be used to handle multiple **interactive jobs**.
- In this latter case, the technique is referred to as time sharing, because processor time is shared among multiple users.

Multitasking or Time Sharing System:

- A multitasking system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time shared computer.
- Each user has at least one separate program in memory.
- Multitasking are more complex than multiprogramming and must provide a mechanism for jobs synchronization and communication and it may ensure that system does not go in deadlock.
- Most of the system today available uses the concept of multitasking and Multiprogramming.

Real-time Systems

- Real-time systems are used when there are rigid time requirements on the operation of a processor or the flow of data.
- 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.
- Primary objective of Real Time Operating System is to provide quick response time and thus to meet deadline.
- User convenience and resource utilization are secondary concern to these systems.
- E.g., Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-applicance controllers.

There are two types of real-time systems:

Hard real-time systems

- If any deadline is missed then system will fail to work or does not work properly.
- This system guarantees that critical task is completed on time
- Eg: military(firing of missiles), industrial process control, robotics etc.

Soft real-time systems

- Soft real time systems are less restrictive.
- Critical real-time task gets priority over other tasks and retains the priority until it completes.
- If certain deadlines are missed then system continues its working with no failure but its performance degrade.
- E.g., Multimedia, virtual reality etc.

Personal-Computer Systems (PCs)

- A computer system is dedicated to a single user is called personal computer.
- Micro computers are considerably smaller and less expensive than mainframe computers.
- The goals of the operating system have changed with time; instead of maximizing CPU and peripheral utilization, the systems developed for maximizing user convenience and responsiveness.

  For e.g., MS-DOS, Microsoft Windows and Apple Macintosh.

- Hardware costs for microcomputers are sufficiently low. Decreased cost of computer hardware (such as processors and other devices) .

MAINFRAME OPERATING SYSTEMS

- These computers distinguish themselves from personal computers in terms of their I/O capacity.
- A mainframe are equipped with several disks and thousands of gigabytes of data.
- The operating systems for mainframes are heavily oriented toward processing many jobs at once.
- They typically offer three kinds of services: batch, transaction processing, and timesharing.

MAINFRAME OPERATING SYSTEMS

- A **batch system** is one that processes routine jobs without any interactive user, for example, sales reporting for a chain of stores is typically done in batch mode.
- **Transaction processing systems** handle large numbers of small requests, for example, check processing at a bank or airline reservations. Each unit of work is small, but the system must handle hundreds or thousands per second.
- **Timesharing systems** allow multiple remote users to run jobs on the computer at once, such as querying a big database.

Handheld Computer Operating Systems

- Continuing on down to smaller and smaller systems, we come to tablets, smartphones and other handheld computers.
- **PDA** (**Personal Digital Assistant**), is a small computer that can be held in your hand during operation.
- Smartphones and tablets are the best-known examples (OHA, Google's Android and Apple's iOS.)
- Most of these devices use multicore CPUs, GPS, cameras and other sensors, copious(abundant) amounts of memory, and sophisticated operating systems.

Handheld Computer Operating Systems

- Symbian - Used in cell phones, mainly ones made by Nokia
- BlackBerry OS - For BlackBerry phones
- iOS - Subset of Mac OS X, used in Apple's mobile devices
- Palm OS
- Windows Mobile

Embedded Operating Systems

- Embedded systems run on the computers that control devices which are not generally thought of as computers and which do not accept user-installed software.
- Typical examples are home appliance like microwave ovens, TV sets, automobiles like cars.
- You cannot download new applications to your microwave oven—all the software is in ROM.
- Systems such as Embedded Linux, QNX and VxWorks are popular in this domain.

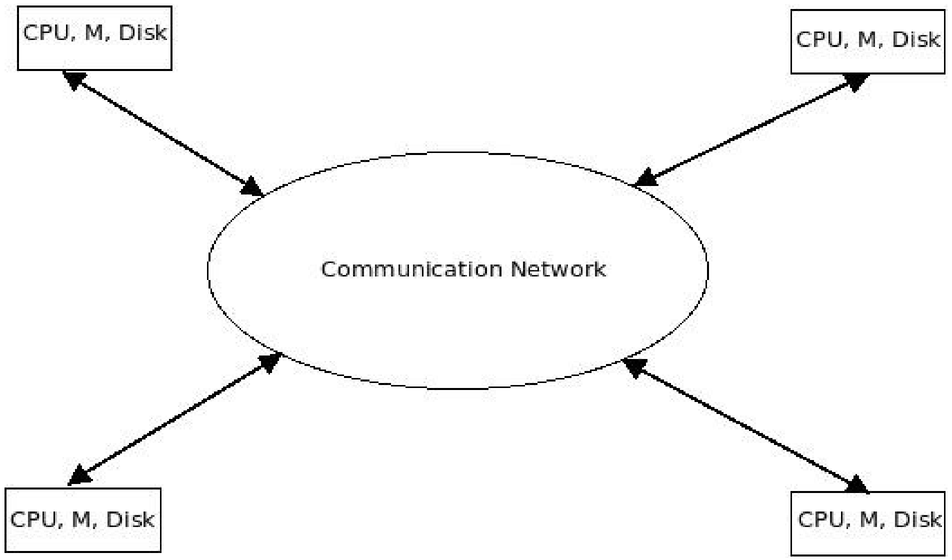
Distributed Operating System:

- A recent trend in computer system is to distribute computation among several processors.
- It contrasts to the tightly coupled system where the processors 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 computer network.
- Distributed operating system are the operating system for a distributed system(a network of autonomous computers connected by a communication network through a message passing mechanisms).

Distributed Operating System:..

- A distributed operating system controls and manages the hardware and software resources of a distributed system.
- When a program is executed on a distributed system, user is not aware of where the program is executed or the location of the resources accessed.
- Example of Distributed OS:

Amoeba, Chorus, Mach



Operating System Structure:

- Operating systems are broadly classified into following categories, based on the their structuring mechanism as follows:

a. Monolithic System

b. Layered System

c. Virtual Machine

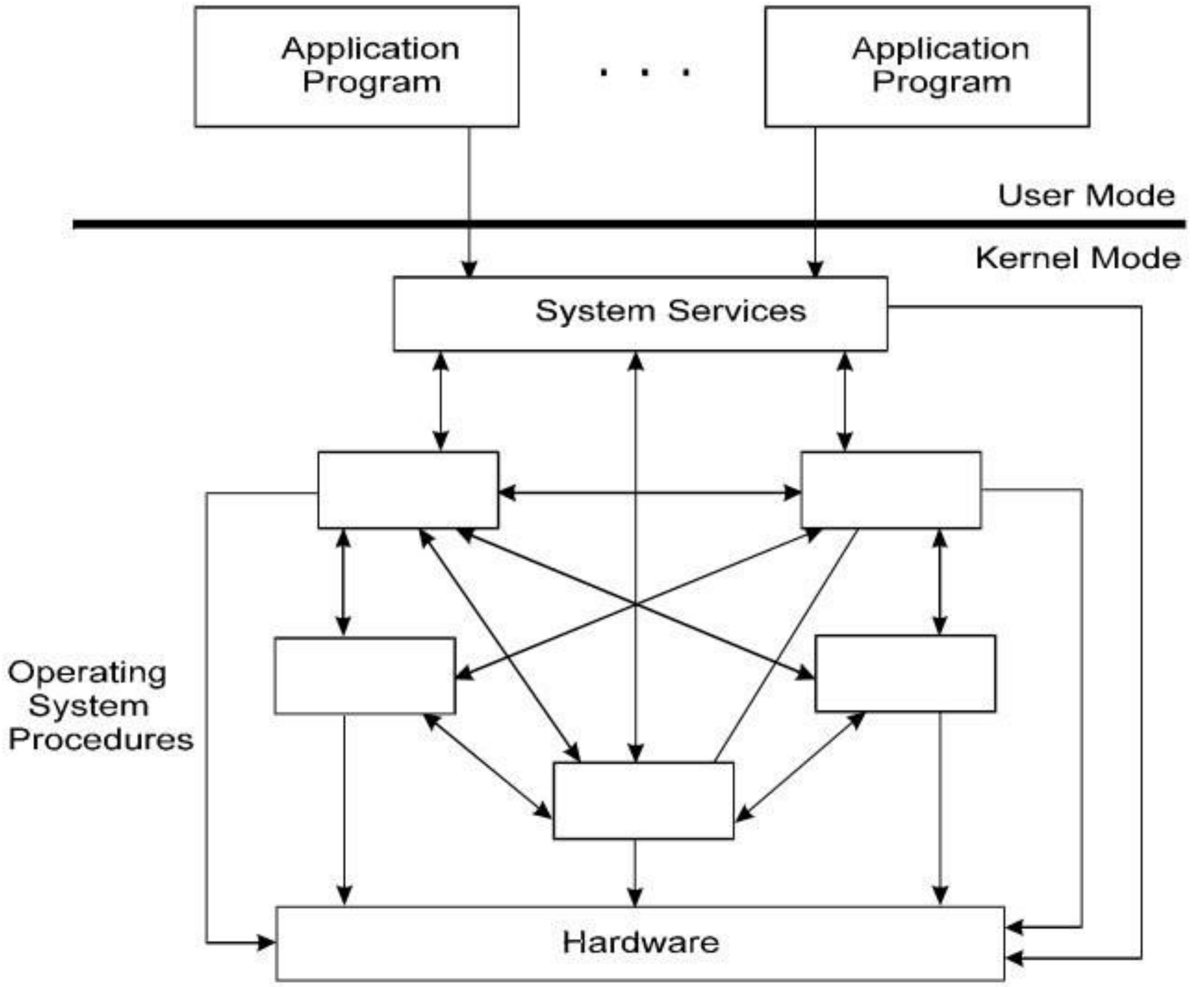
d. Exokernels

e. Client-Server Model(microkernel)

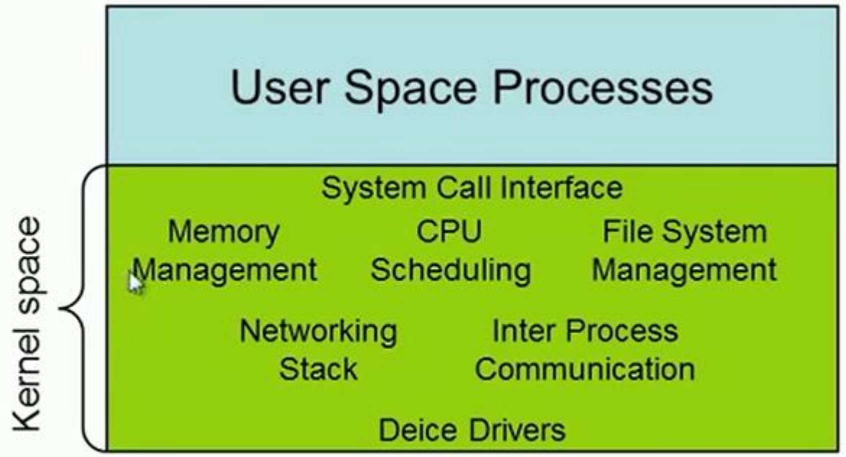
1.Monolithic System

- The components of monolithic operating system are organized haphazardly and any module can call any other module without any reservation.
- The operating system code runs in a privileged processor mode (referred to as kernel mode), with access to system data and to the hardware;
- Applications run in a non-privileged processor mode (called the user mode), with a limited set of interfaces available and with limited access to system data.
- The operating system is written as a collection of procedures, each of which can call any of the other ones whenever it needs to.
- This approach might well be subtitled "The Big Mess." The structure is that there is no structure.
- The monolithic operating system structure with separate user and kernel processor mode is shown in Figure. **Example Systems: CP/M and MS-DOS**

1.Monolithic System..



OS Structure : Monolithic Structure



- Linux, MS-DOS, xv6
- All components of OS in kernel space
- Cons : Large size, difficult to maintain, likely to have more bugs, difficult to verify
- Pros : direct communication between modules in the kernel by procedure calls

2. Layered Operating System

- The layered approach consists of breaking the operating system into the number of layers(level), each built on the top of lower layers.
- The bottom layer is the hardware layer; the highest layer is the user interface.
- The main advantages of the layered approach is modularity. The layers are selected such that each uses functions (operations) and services of only lower-level layers.
- In this approach, the Nth layer can access services provided by the

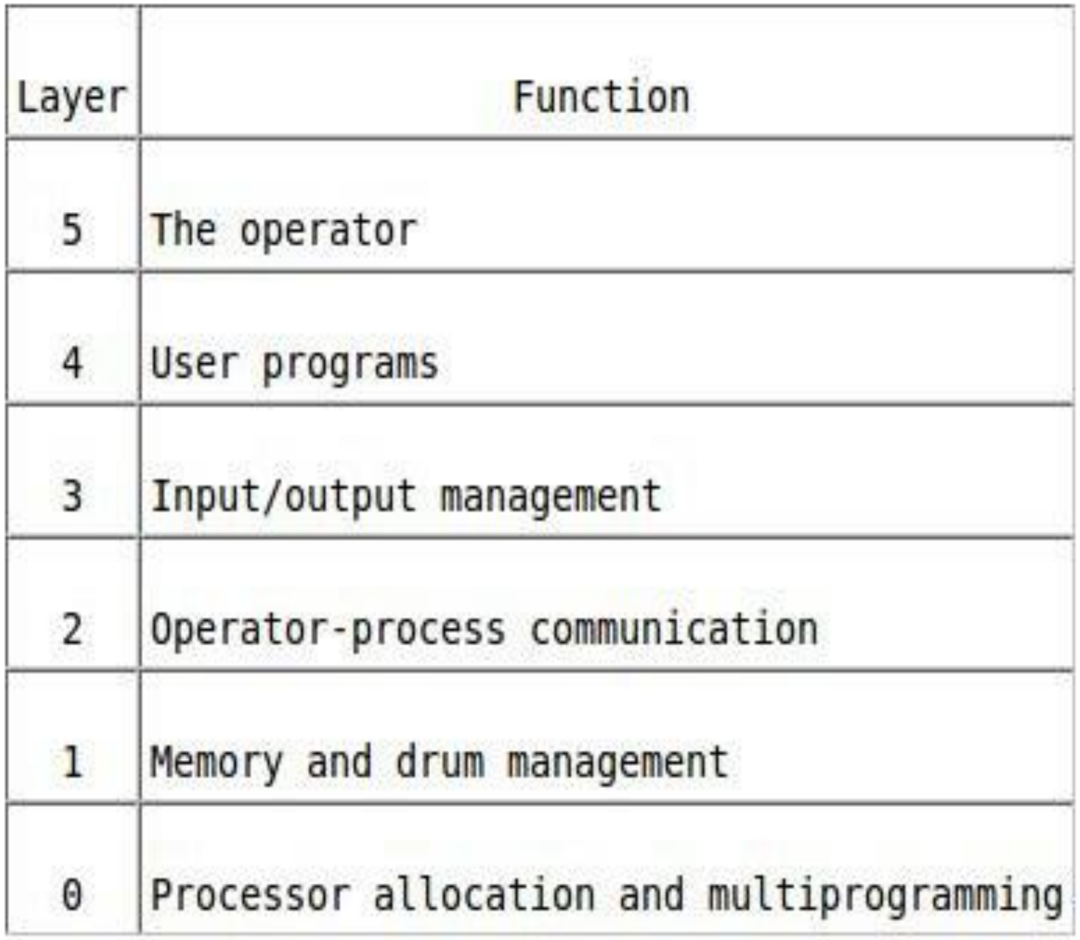
  (N-1)th layer and provide services to the (N+1)th layer.
- This structure also allows the operating system to be debugged starting at the lowest layer, adding one layer at a time until the whole system works correctly.
- Layering also makes it easier to enhance the operating system.

2. Layered Operating System..

- If an error is found during the debugged of particular layer, the layer must be on that layer, because the layer below it already debugged.
- Because of this design, the system is simplified when operating system is broken up into layer.
- Os/2 operating system is example of layered architecture of operating system another example is earlier version of Windows NT.
- The main disadvantage of this architecture is that it requires an appropriate definition of the various layers & a careful planning of the proper placement of the layer.
- The layer approach design was first used in the THE operating system at the **Technische Hogeschool Eindhoven.**
- The THE system was defined in the six layers , as shown in the fig below.

2. Layered Operating System..

- Example Systems:
- VAX/VMS,
- Multics,
- UNIX



Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

3. Virtual Machines:

- Virtual machine is an illusion of a real machine.
- It is created by a real machine operating system, which make a single real machine appears to be several real machine.
- The architecture of virtual machine is shown below.
- The best example of virtual machine architecture is IBM 370 computer.
- In this system each user can choose a different operating system.
- Actually, virtual machine can run several operating systems at once, each of them on its virtual machine.
- Its multiprogramming shares the resource of a single machine in different manner

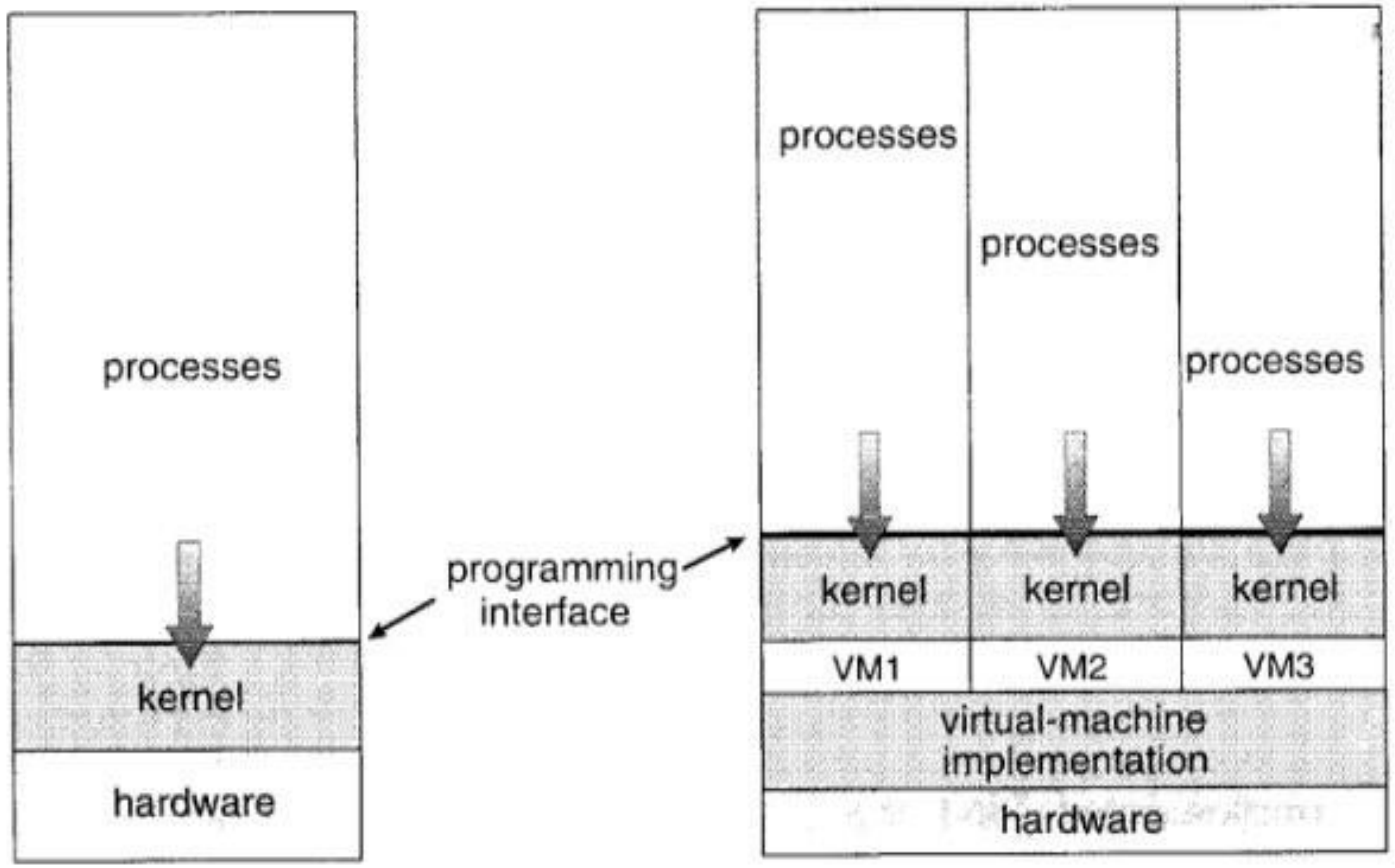


Fig: a). Non Virtual Machine b). Virtual Machine.

Although the virtual machine concept is useful, it is difficult to implement. Much effort is required to provide an exact duplicate of the underlying machine.

Example. Java

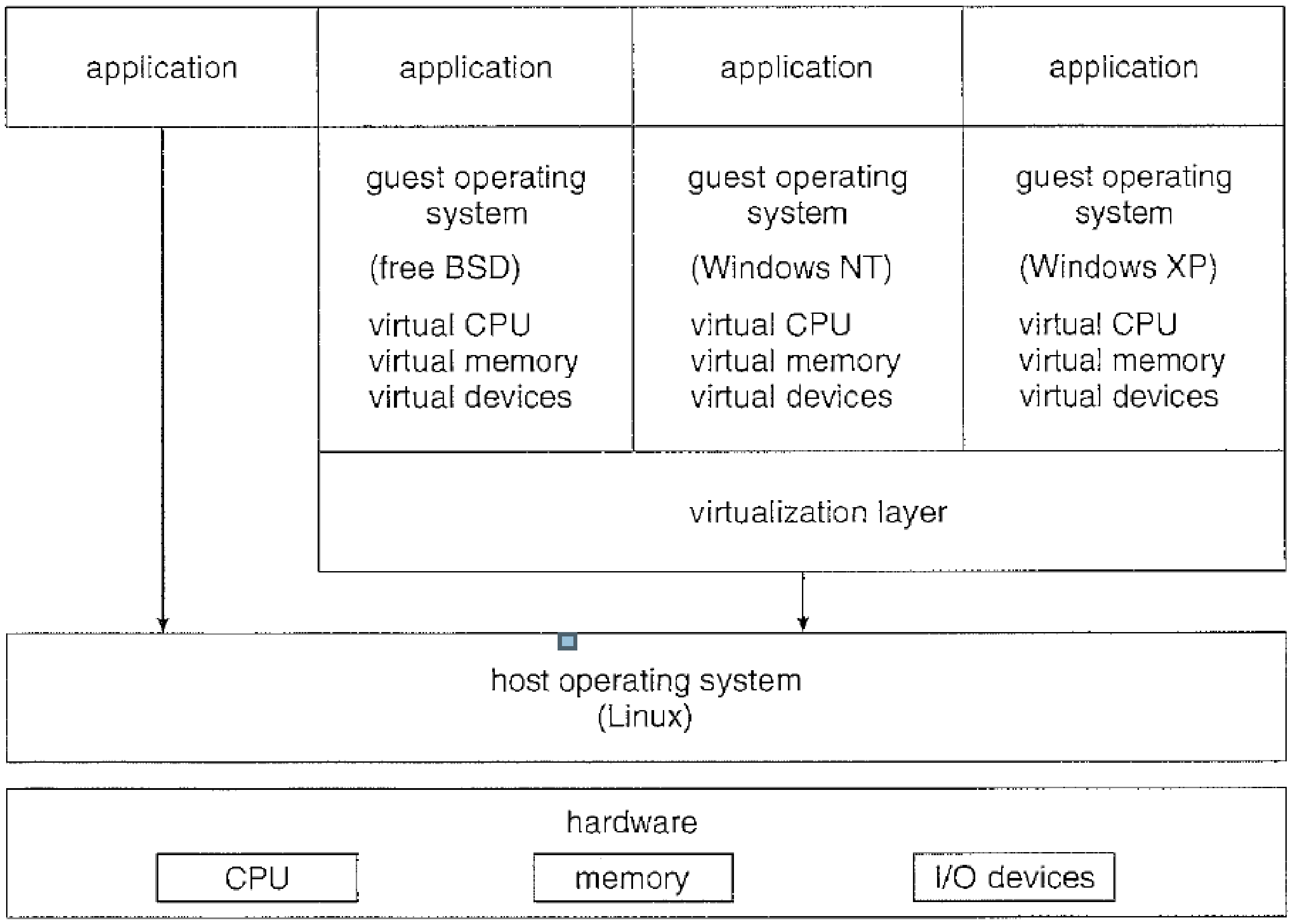
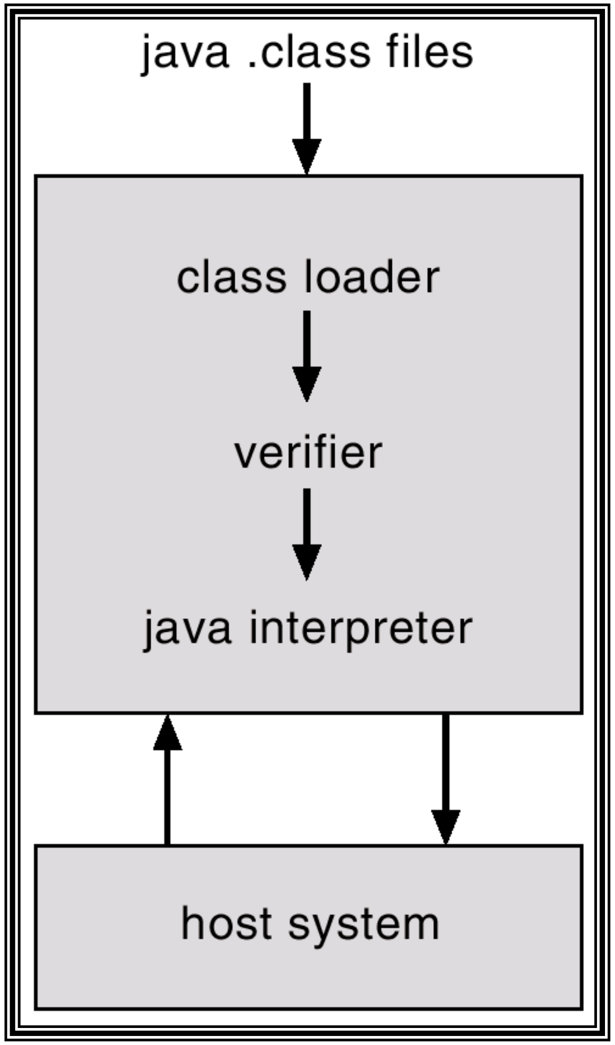


Figure 2.19 VMware architecture.

⇒ **JVM (Java Virtual Machine):** JVM consists of

- class loader
- class verifier
- runtime interpreter



3. Virtual Machines:…

⇒ **Advantages:**

● The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.

● A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.

⇒ **Disadvantage:**

● The virtual machine concept is difficult to implement due to the effort required to provide an exact duplicate to the underlying machine

4.Client Server Structure (microkernel)

- Two classes of processes - Server and Clients
- Communication between client and server is via message passing
- Client and server can run on different computers connected by LAN/WAN
- Servers run as user mode. Hence, no system down even if the server crashed
- Well adopted in distributed system
- E.g. Windows NT

4.Client Server Structure (microkernel)…

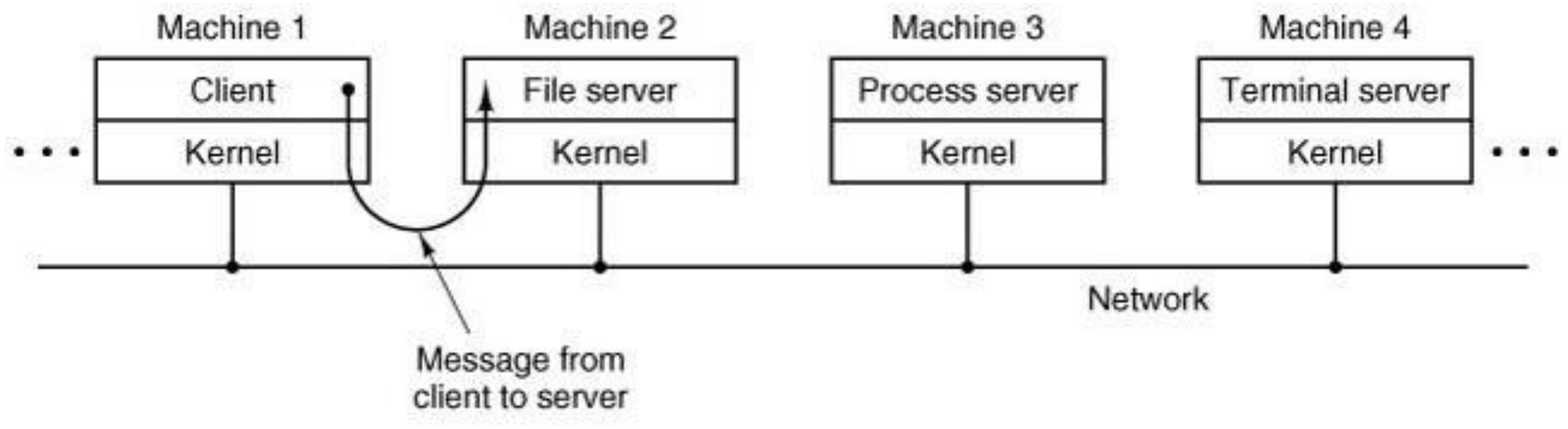
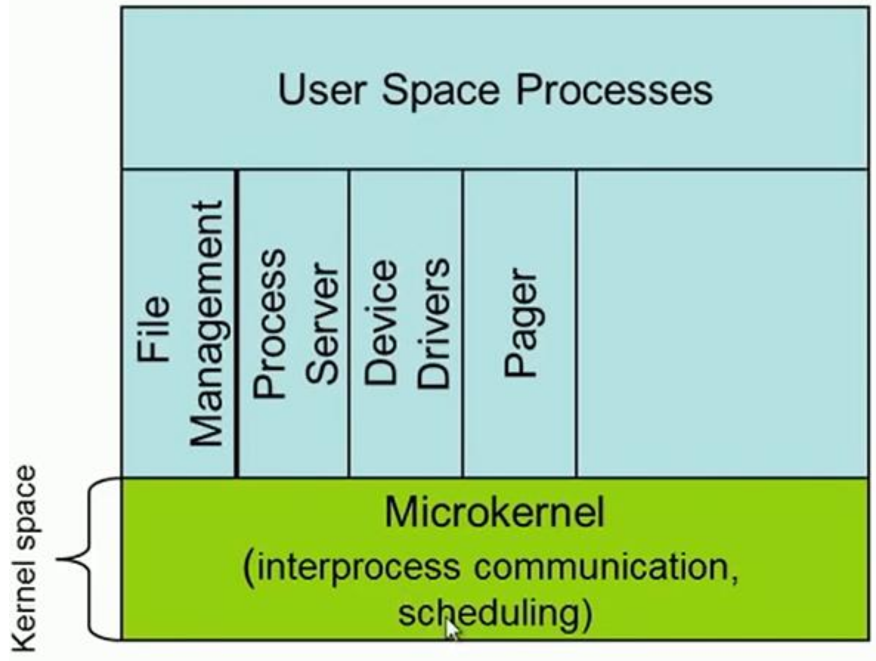


Fig: The client-server model (in a distributed system.)

5. Microkernel

- The new concepts in operating system design, microkernel, is aimed at migrating traditional services of an operating system out of the monolithic kernel into the user-level process.
- The idea is to divide the operating system task into several processes, each of which implements a single set of services
- - for example, I/O servers, memory server, process server, threads interface system.

OS Structure : Microkernel



- Highly modular.
  - Every component has its own space.
  - Interactions between components strictly through well defined interfaces (no backdoors)
- Kernel has basic inter process communication and scheduling
  - Everything else in user space.

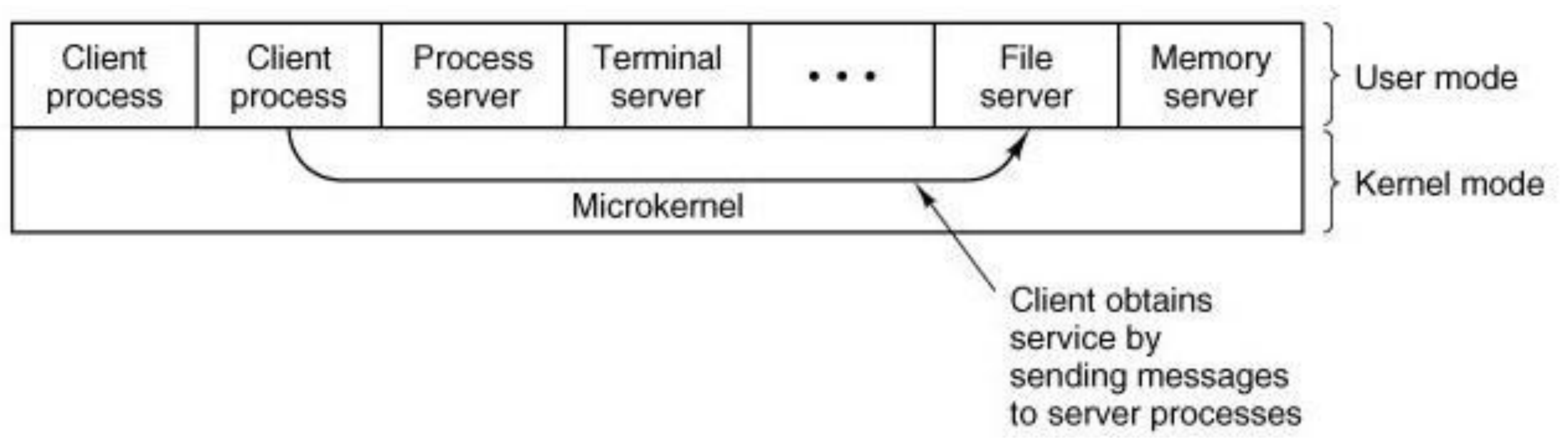


Fig: The client-server model(microkernel in a non distributed system).

5. Microkernel..

- Each server runs in user mode, provides services to the requested client.
- The client, which can be either another operating system component or application program, requests a service by sending a message to the server.
- An OS kernel (or microkernel) running in kernel mode delivers the message to the appropriate server;
- The server performs the operation; and microkernel delivers the results to the client in another message, as illustrated in Figure of client server structure earlier.

6. Exo-kernel architecture

- It is a further extension of the micro-kernel approach where the kernel is devoid of functionality.
- This means the request for file access by one process would be passed by the kernel to the library that is directly responsible for managing file system.
- **Comparison as:**

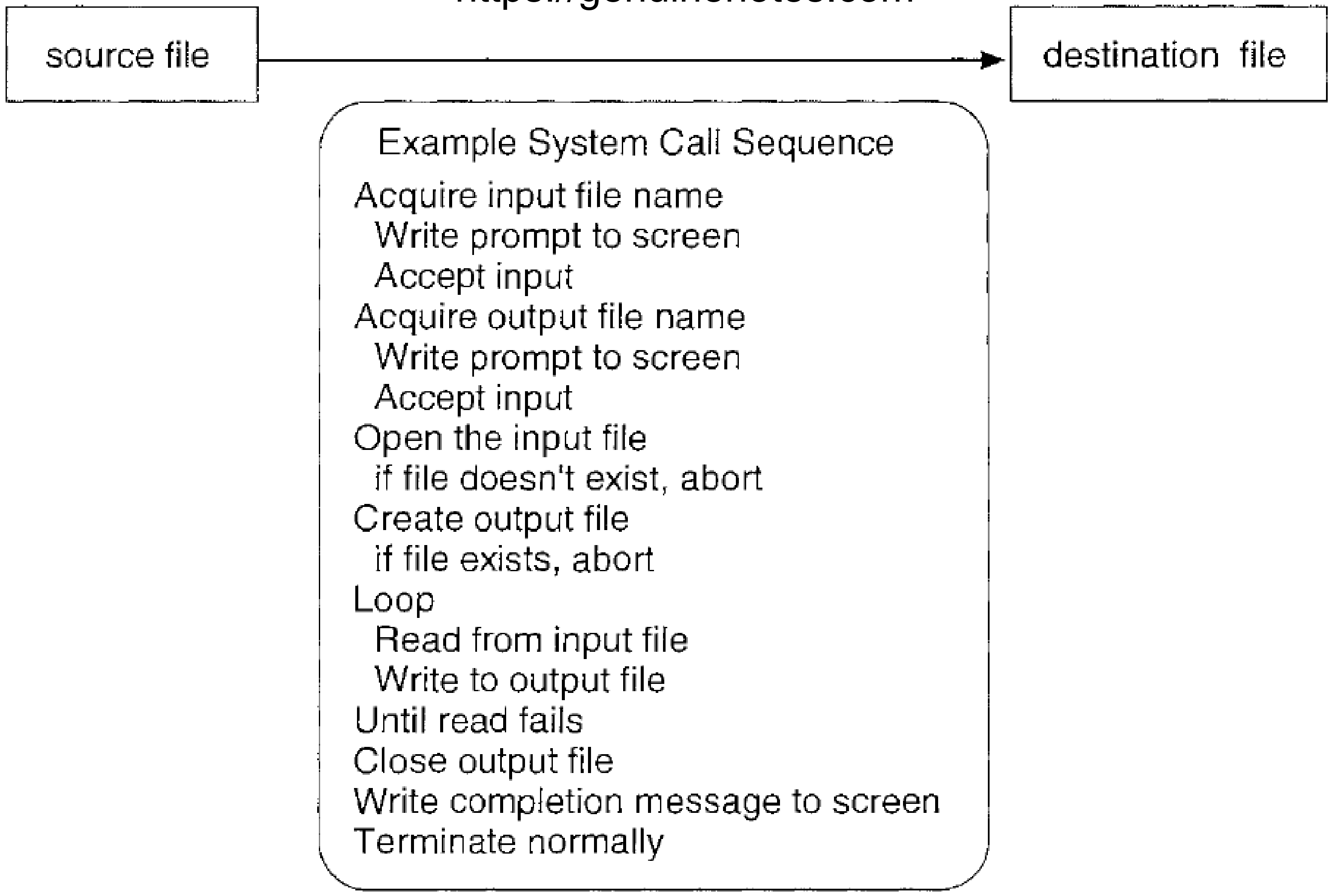
-In monolithic everything is implemented in the kernel space

-In microkernel only the lower level operating system facilities are implemented in the kernel

-In Exo-kernel nothing is implemented in kernel space.

System Call:

- In computing, a **system call** is the mechanism used by an application program requests a service from an operating system's kernel.
- This may include hardware related services (e.g. accessing the hard disk), creating and executing new processes, and communicating with integral kernel services (like scheduling).
- System calls provide the interface between a process and the operating system.
- On Unix, Unix-like and other POSIX-compatible (Portable operating system interface) operating systems, popular system calls are open, read, write, close, wait, fork, exit, and kill.
- Many of today's operating systems have hundreds of system calls.
- For example, Linux has over 300 different calls.



Example System Call Sequence

```
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```

Figure 2.4 Example of how system calls are used.

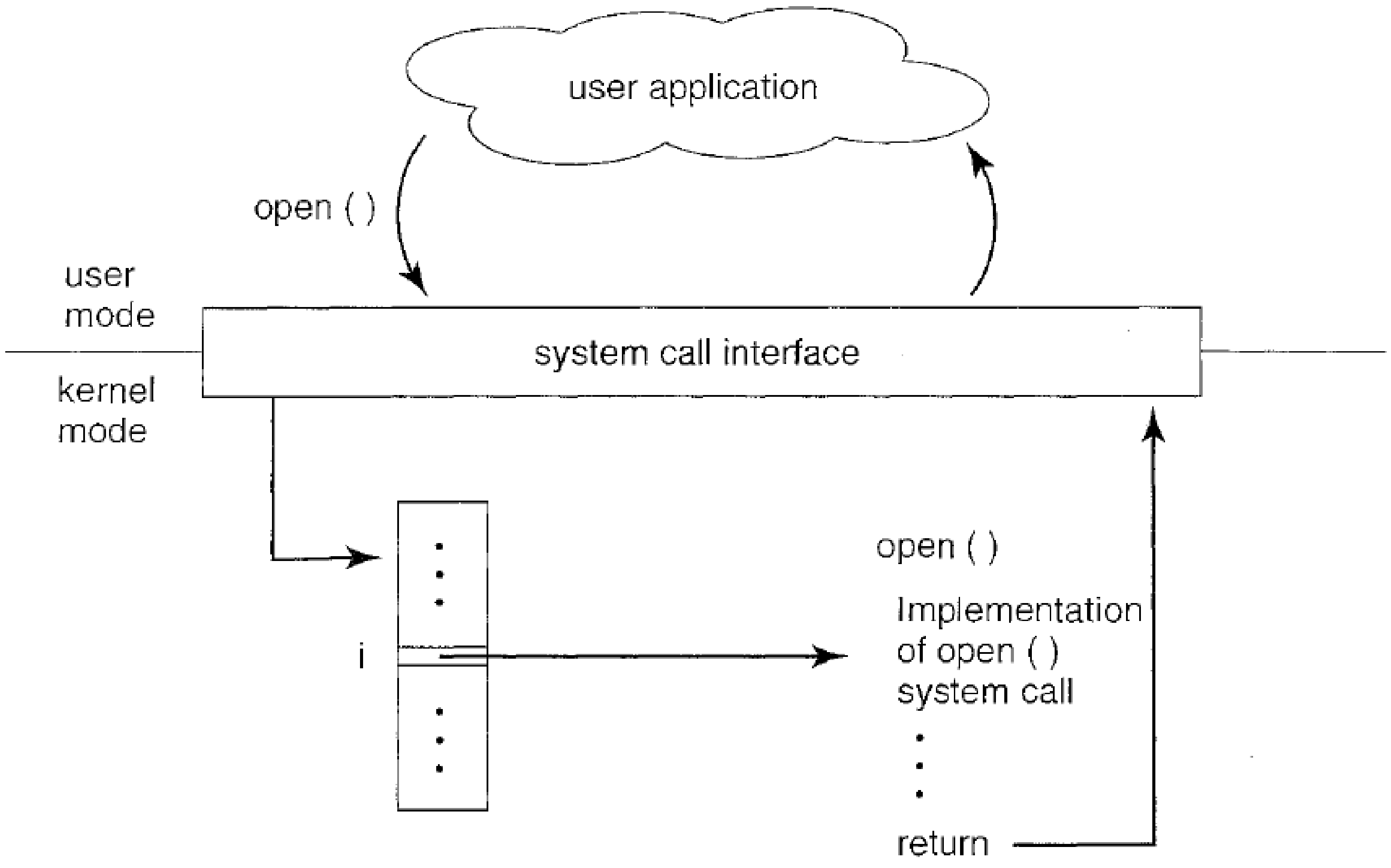
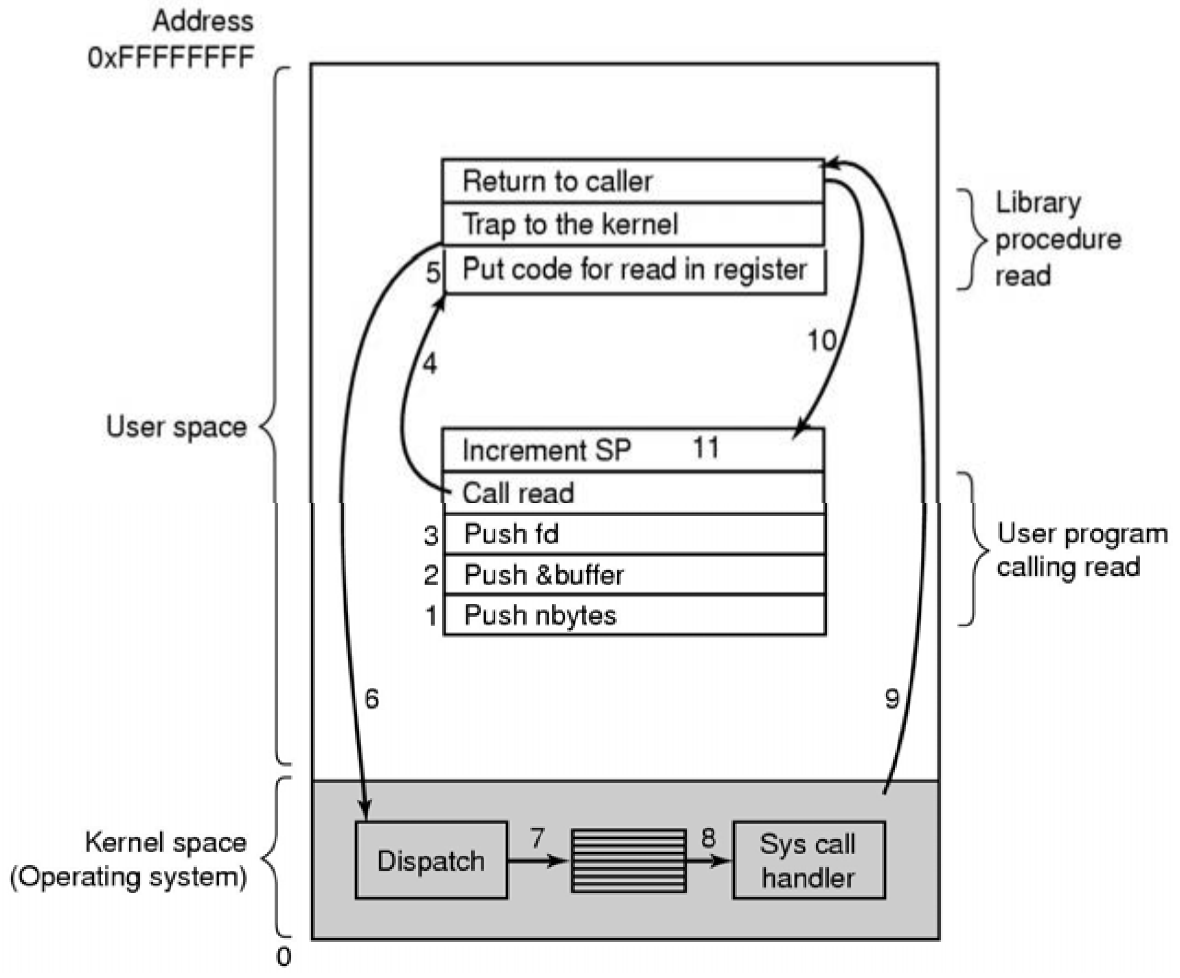


Figure 2.6 The handling of a user application invoking the `open()` system call.

Steps in Making a System Call (example)

There are 11 steps in making the system call ***read (fd, buffer, nbytes)***

- Push parameter into the stack (1-3)
- Calls library procedure (4)
- Pass parameters in registers (5)
- Switch from user mode to kernel mode and start to execute (6)
- Examine the system call number and then dispatch to the correct system call handler via a table of pointer (7)
- Run System call Handlers (8)
- Once the system call handler completed its work, control return to the library procedure (9)
- This procedure then return to the user program in the usual way (10)
- Increment SP to clean up the stack before call to finish the job. (11)



UNIX has a system call read with three parameters:
count=read(fd,buffer,n bytes)
One to specify the file, one to tell where the data are to be put and one to tell how many bytes to be read

Different types of system call

1. Process Control:

Many system calls exist for the purpose of controlling processes. These system calls include:

- **end, abort –** A running program needs to be able to halt its execution either normally (*end*) or abnormally (*abort*).
- **load, execute –** A process or job executing one program may want to *load* and *execute* another program.
- **create process, terminate process –** A new job or process is created when needed (*create process*) and terminated if it is incorrect or no longer needed (*terminate process*).
- **get process attributes, set process attributes –** To control the execution of a job or process, it requires the ability to determine (*get process attributes*) and reset (*set process attributes*) the attributes of job or process, including the job's priority, its maximum allowable execution time, and so on.

1. Process Control:...

- **wait for time –** Having created new jobs or processes, we may need to wait for a certain amount of time to finish their execution (*wait time*).
- **wait event, signal event –** Having created new jobs or processes, we may want to wait for a specific event to occur (*wait event*). The jobs or processes should then signal when that event has occurred (*signal event*).
- **allocate memory, free memory –** A program needs memory before it executes (*allocate memory*) and that memory should be freed after its termination (*free memory*).

2. File Management:

Several system calls dealing with files are:

- **create file, delete file –** We may want to create (*create*) and delete (*delete*) files.
- **open file, close file –** Once the file is created, we need to open (*open*) it and to use it. Finally, we need to close (*close*) the file, indicating that we are no longer using it.
- **read file, write file** – We may also read (*read*), write (*write*) the file after opening it.
- **get file attributes, set file attributes –** We may also need to determine (*get file attribute*) and reset (*set file attribute*) file attributes if necessary. File attributes include the file name, file type, and so on.

If we have directory structure for organizing files in the file system, the same set of operations are needed for directories.

3. Device Management:

System calls dealing with devices include:

- **request device, release device –** A running program must first request the device (*request*) to ensure exclusive use of it. After finished with the device, it must be released (*release*). These functions are similar to the *open* and *close* system calls for files.
- **read, write –** Once the device has been requested and allocated, a process can read, write, the device, just as with ordinary files.
- **get device attributes, set device attributes –** A process may also need to determine (*get device attribute*) and reset (*set device attribute*) device attributes if necessary. File attributes include the device name, device type, and so on.

4. Information Maintenance:

These system calls exist for the purpose of transferring information between the user program and the operating system.

- **get time or date, set time or date –** Most systems have system calls to determine and reset system time or date.
- **get system data, set system data –** These system calls determine and reset 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.
- **get process, file, or device attributes –** These system calls determine attributes about all its processes, files and devices used in the processes.
- **set process, file, or device attributes –** These system calls reset information about all its processes, files and devices used in the processes.

5. Communications:

There are two common models of communication: **message passing model** and **shared memory model**.

In the message-passing model, processes communicate by exchanging information through an interprocess-communication facility provided by the OS. The system calls associated with this model are:

- **get host id** – Translates the host name into an equivalent identifier.
- **get process id** – Translates the process name into an equivalent identifier.
- **open** and **close** or specific **open connection** and **close connection**

– The above identifiers are passed to the general purpose **open** and **close** system calls provided by the file system or to specific **open connection** and **close connection** system calls.

5. Communications: ..

- **accept connection** – The recipient process usually must give its permission for communication to take place with an **accept connection** call.
- **wait for connection –** The processes that will be receiving connections execute a wait for connection call and are awakened when the connection is made.
- **read message** and **write message –** The sending and receiving processes exchange messages by **read message** and **write message** system calls.

In the shared-memory model, processes communicate by using a common memory. The system calls associated with this model are:

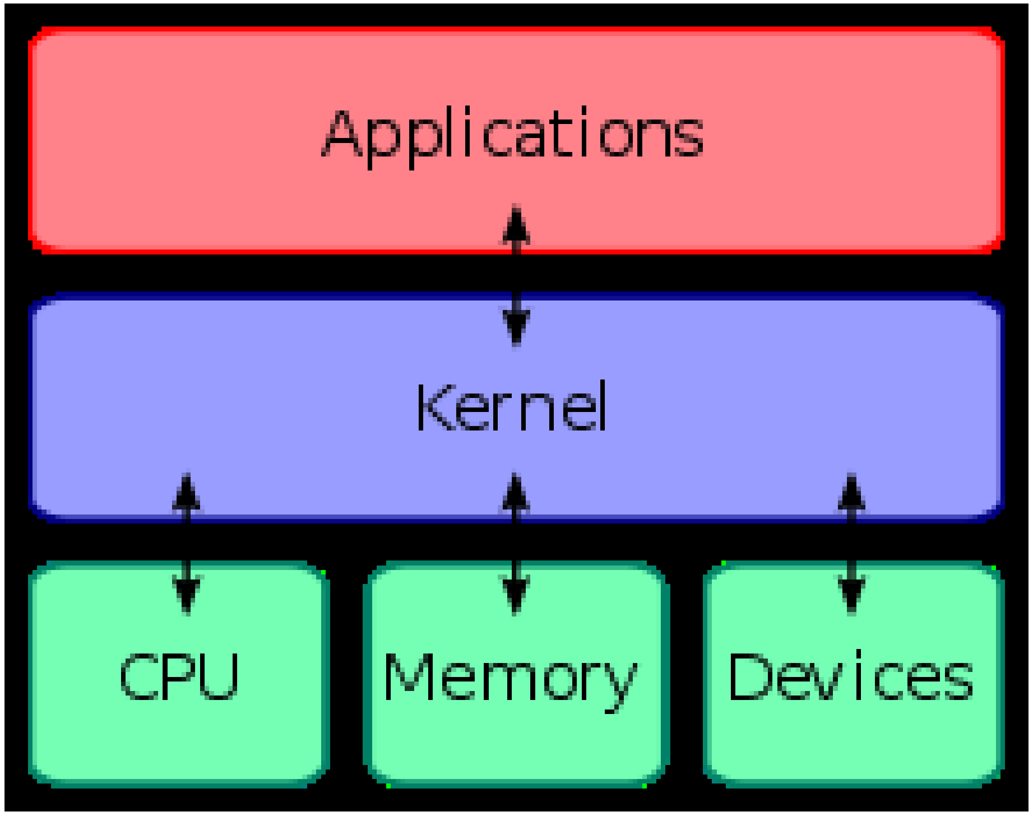
- **map memory** – A process use this system call to gain access to regions on memory owned by other processes.
- **read** and **write –** They may then exchange information by reading and writing data in these shared areas.

Shell:

- A shell is a program that provides the traditional text only user interface for Linux and other Unix operating system.
- In computing, a shell is a piece of software that provides an interface for users and provides access to the services of a kernel.
- Its primary function is to read commands typed into a console or terminal window and then execute it.
- The term shell derives its name form the fact that it is an outer layer of the OS.
- A shell is an interface between the user and the internal part of the operating system.
- A user is in shell(i.e interacting with the shell) as soon as the user has logged into the system.
- A shell is the most fundamental way that user can interact with the system and the shell hides the detail of the underlying system from the user.
- **Example:** Bourne Shell, Bash shell, Korn Shell, C shell

Kernel:

- In computing, the **kernel** is the main component of most computer operating systems;
- It is a bridge between applications and the actual data processing done at the hardware level.
- The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).
- It typically makes these facilities available to application processes through inter-process communication mechanisms and system calls.
- Monolithic kernels execute all the operating system code in the same address space,
- Microkernels run most of the operating system services in user space as servers, aiming to improve maintainability and modularity of the operating system.


Applications
Kernel
CPU
Memory
Devices

Finished Unit 1