OPERATING SYSTEM

BY :ER. NIRMAL THAPA

INTRODUCTION

- Operating System is a system whose job is to manage all these devices and provide user programs with a simpler interface to the hardware.

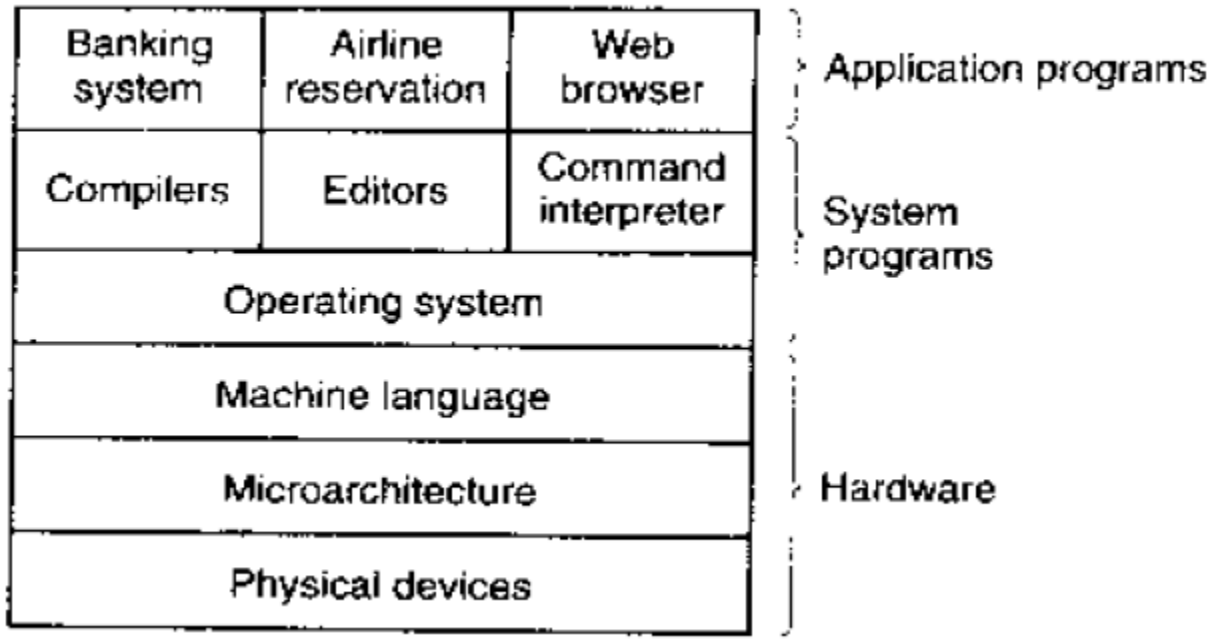
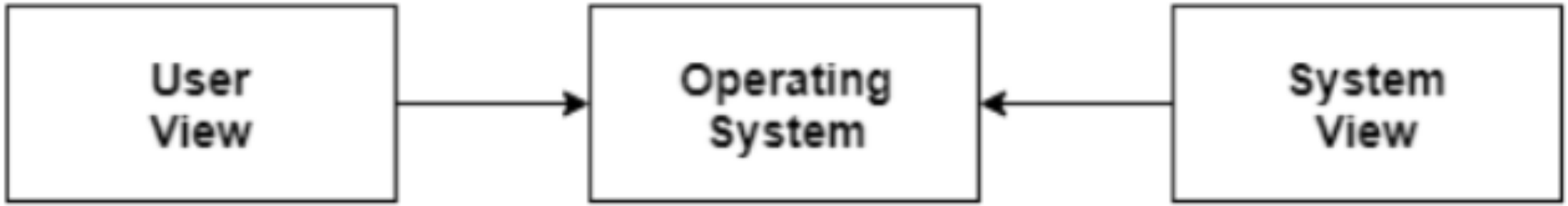


Fig. Different layers of Computer System

TWO VIEWS OF OPERATING SYSTEM



USER VIEW

- The user view depends on the system interface that is used by the users
- Such systems are designed for one user to monopolize its resources, to maximize the work that the user is performing.
- In these cases, the operating system is designed mostly for ease of use, with some attention paid to performance, and none paid to resource utilization.

SYSTEM VIEW

- Operating system can be viewed as a resource allocator
- A computer system consists of resources like - hardware and software - that must be managed efficiently.
- The operating system acts as the manager of the resources, decides between conflicting requests, controls execution of programs etc.

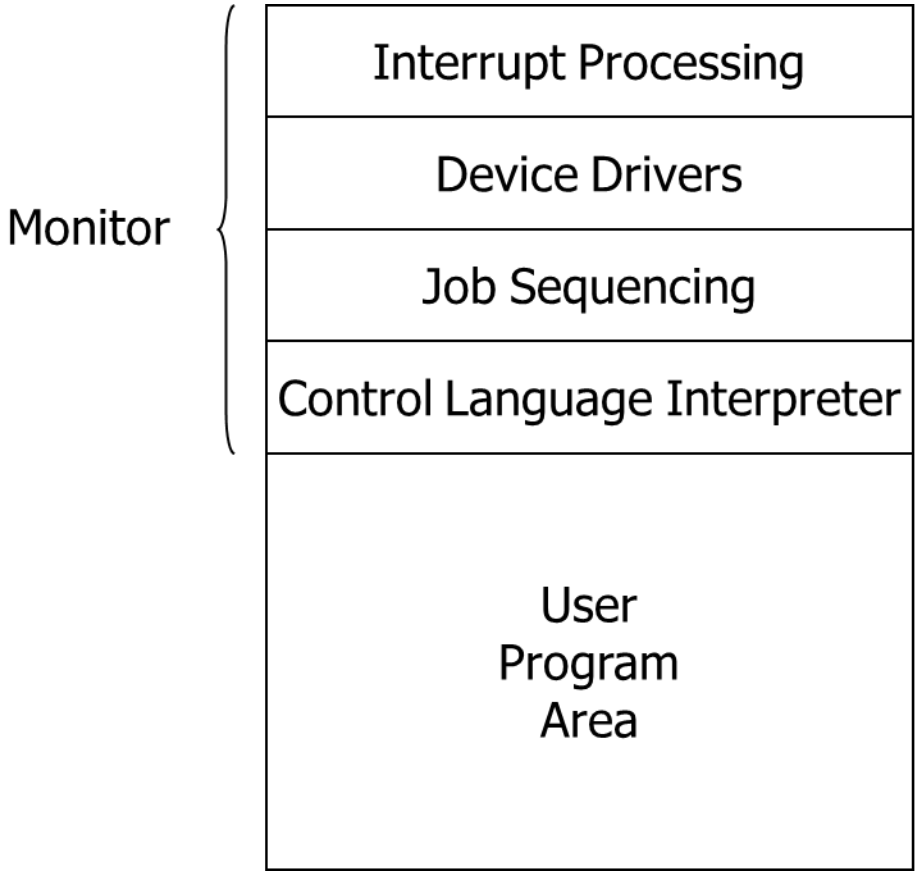
EASE OF EVOLUTION OF AN OS

- A major OS will evolve over time for a number of reasons:
  - Hardware upgrades plus new types of hardware
  - New services demanded by users
  - Fixer – fix holes in OS

EVOLUTION OF OPERATING SYSTEM

- 1940-1950 (OS 1)
  - No operating system
  - Run as an open shop
  - HWs like Vaccum tubes or plugboards were used
  - No protection
  - Generate more heats and consume more electricity
  - User signs up for certain time to use it
  - Work with machine language

SIMPLE BATCH SYSTEMS (OS 2)

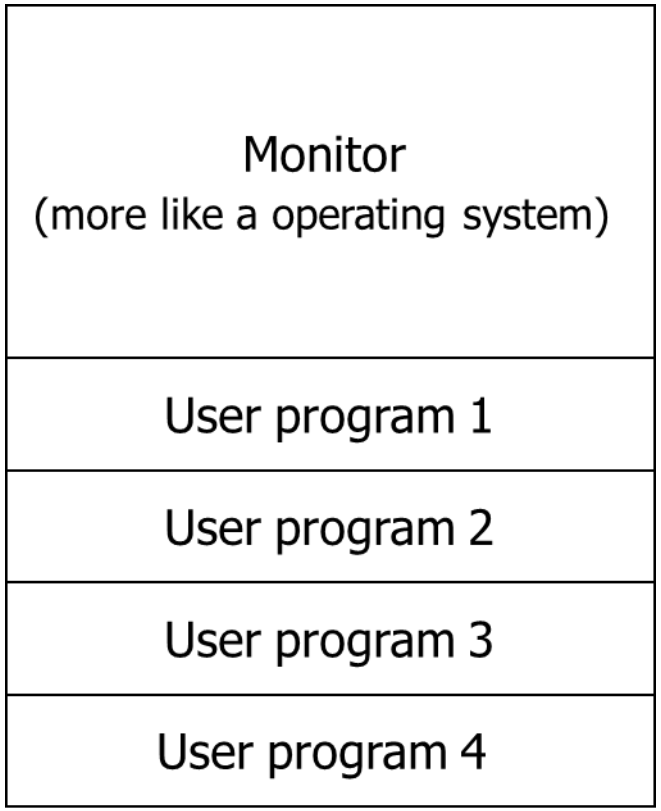


- mid 1950s - mid 1960s
  - User no longer has direct access to the machine
  - processor had to remain Idle.
  - Submit the job to an operator who batches the jobs together sequentially and places the entire batch on the input device
  - Often jobs of a similar nature can be bundled together to further increase economy
  - Laid the foundation of multitasking

SIMPLE BATCH SYSTEMS (CONT.)

- Job Control Language (JCL)
  - Special type of programming language
  - Provide instruction to the monitor
- Need additional hardware features to support the batch OS:
  - Memory Protection – protect the OS from being wiped out
  - Timer – prevent the job run infinitely
  - Privileged Instruction
    - certain instruction can only be executed by OS, and not by user.
    - I/O could only be performed in monitor (supervisor) mode,
    - CPU runs in supervisor mode or user mode
  - Interrupts
    - early models did not have this capability.
    - Later models have.
    - Make more efficient use of resources

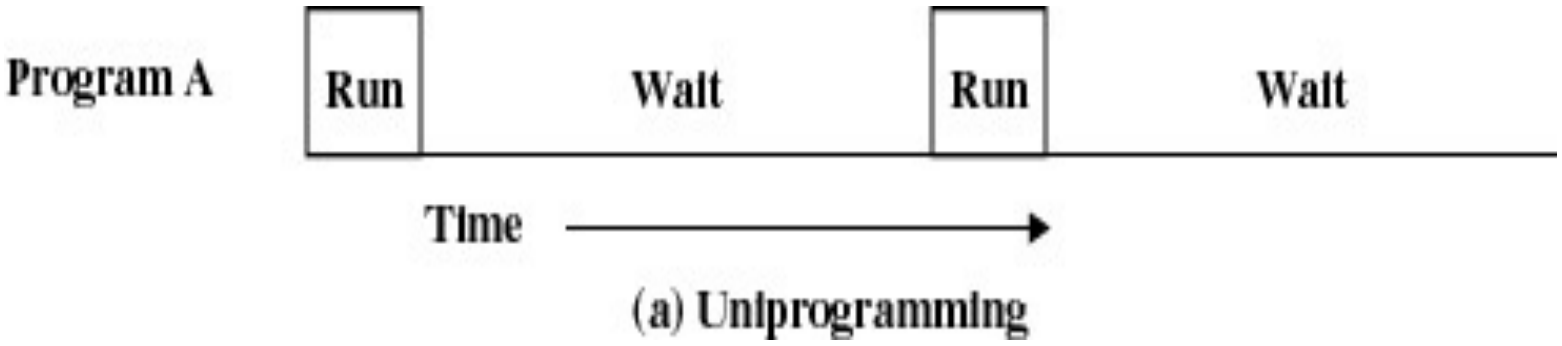
MULTI-PROGRAMMED BATCH SYSTEMS (OS 3)



- 1960s - present
  - Several users are in memory at the same time
  - Match I/O intensive job with CPU intensive job
  - Important to have Interrupt-Driven I/O or DMA to support multiprogrammed batch system.

UNIPROGRAMMING

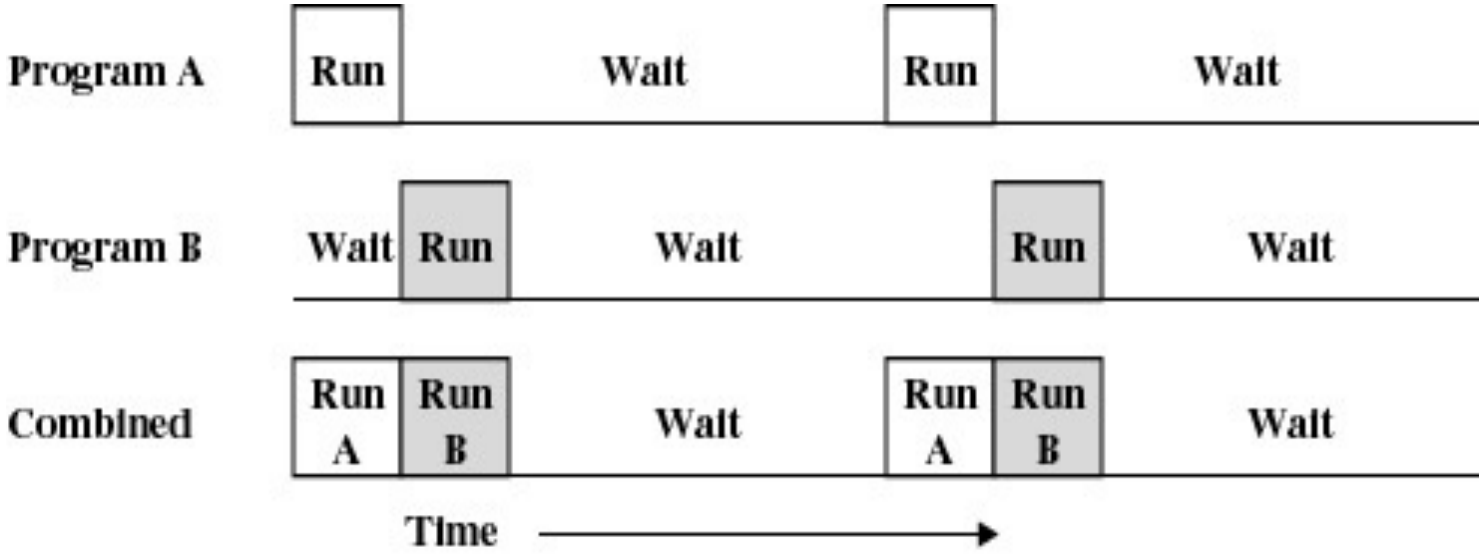
- Processor must wait for I/O instruction to complete before preceding



(a) Uniprogramming

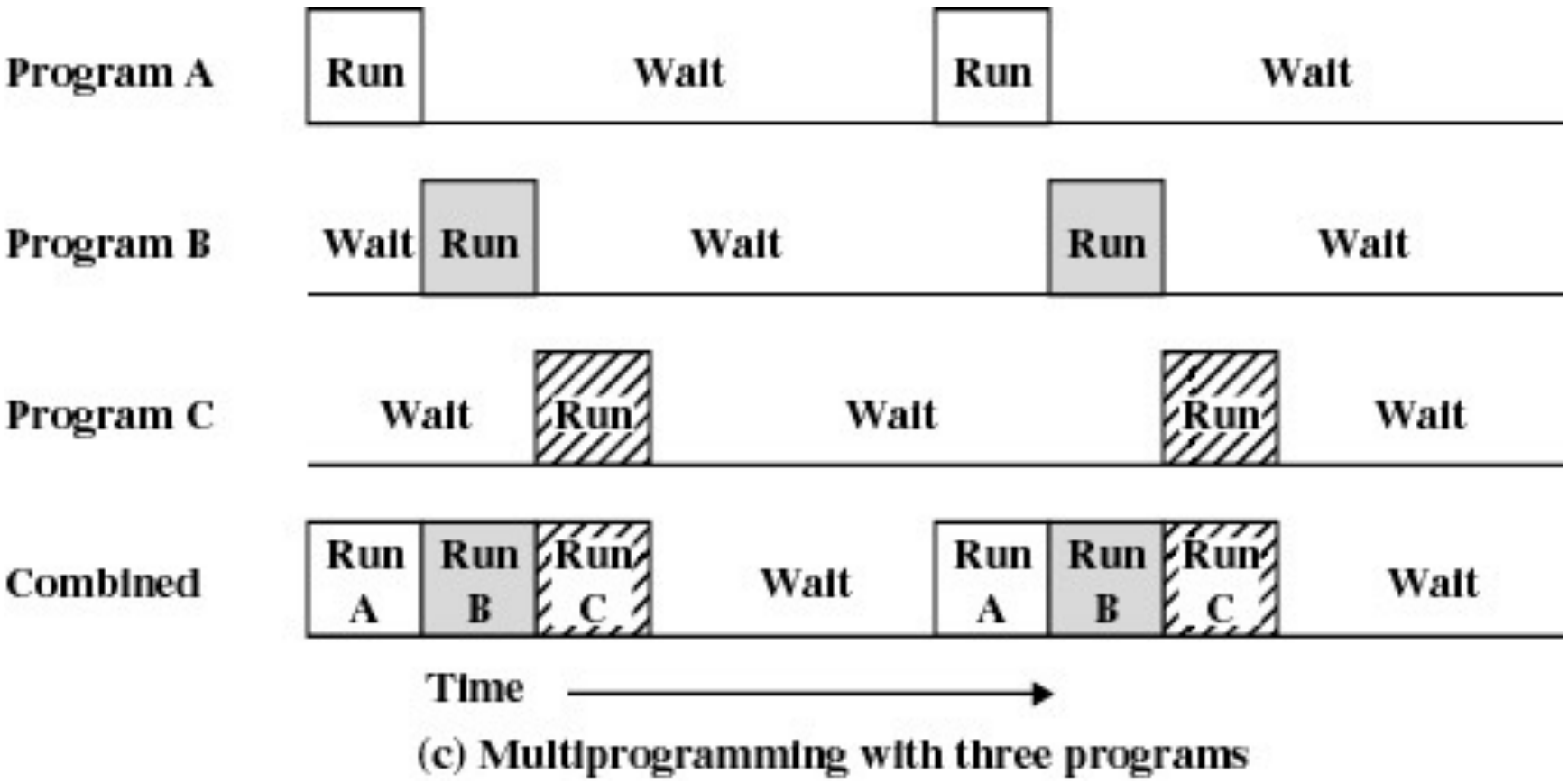
MULTIPROGRAMMING

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

MULTIPROGRAMMING



(c) Multiprogramming with three programs

TIME-SHARING SYSTEMS (OS 4)

- 1970s - present
  - To support interactive computing
  - In a time-sharing system, multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation.
  - Context Switching
    - A job (now often called process) can get "switched in" or "switched out".
    - OS should give the illusion for the process as if it exists in the CPU continuously

BATCH MULTIPROGRAMMING VERSUS TIME SHARING

	Batch Multiprogramming	**Time Sharing**
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

EVOLUTION OF OS 5

- Real-time computer
  - Execute programs that are guaranteed to have an upper bound on tasks that they carry out.
  - e.g. guided missile systems, medical monitoring equipment
- Multiprocessor – have more than one CPU
  - Shared memory multiprocessors
    - Access the same memory - memory access must be synchronized
  - Distributed memory multiprocessors
    - Each CPU has its own associated memory – communication between processors is often slow and complicated
- Networked/Distributed Systems – consist of multiple computers
  - Networked systems: users are aware of the different computers that make up the system
  - Distributed systems: multiple computers are transparent to the user.

TYPES OF OPERATING SYSTEM

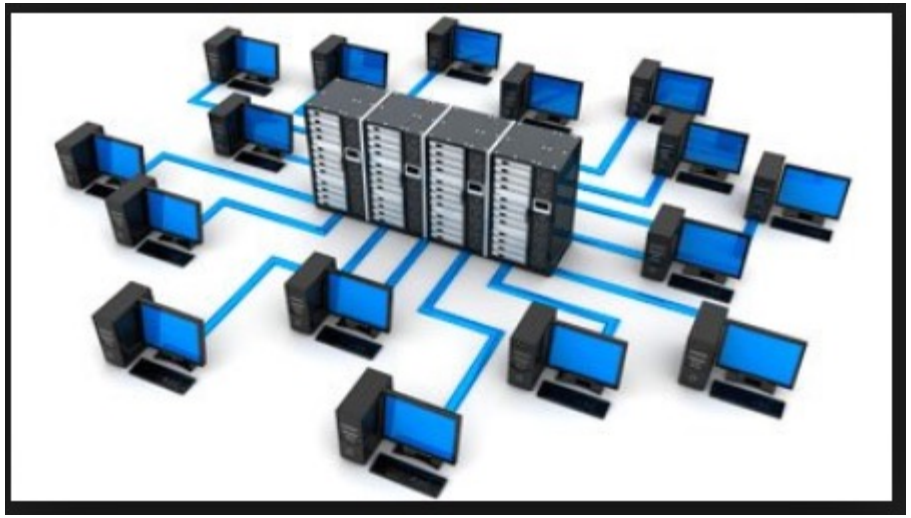
- Mainframe
- Server
- Multiprocessor
- PC
- Real-Time
- Embedded
- Smart card

MAINFRAME OS

- A mainframe computer is a device capable of processing large amounts of information and supporting a great number of users.
- Such computers were generally used to support government and business projects. The interface in such computers is known as Mainframe Operating System.

SERVER OS

- An OS mainly developed for multi-user computer programs, programs important for business computing and networked application is referred to as server operating system.
- Server OS is a special type of Operating system designed to work on server computers.
- They are created or coded based on the functions of the server.



SERVER

- It manages and monitors client's computers and operating systems.
- Act as a central interface for many computers and different operations
- Ability to access server both GUI and command-level interface.
- Higher security on resources and data and large storage management.
- Advanced backup capability and other innovations.
- Applications of modern machine learning and Artificial Intelligence.

SERVER OS TYPES

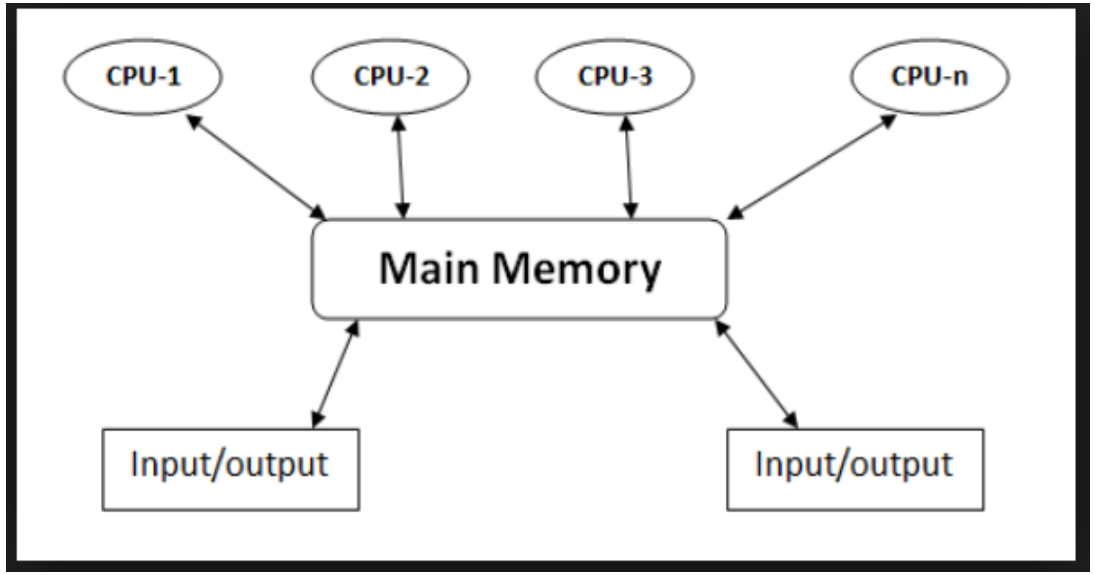
- There are different types of server operating systems based on the kind and functions of the server.
- **UNIX Operating System**
- It was originally a time-sharing operating system for personal and small computers but eventually, it became one of the best server operating systems for client-server business.
- It is focused on the multi-user environment and currently, more than 90 % of the service sites use the UNIX Operating system.
- **LINUX Operating System**
- Linux is open-source or free software that can be used as an alternative for UNIX.
- It has all the features available with the UNIX with high customization options.

SERVER OS TYPES

- **Netware Operating System**
- It is a multi-user server-based network operating system.
- It requires a dedicated server in the network; it is mostly used in the early computer networks.
- **Windows Operating System**
- Windows has released different series of operating systems.
- They update them regularly. It is the best choice if you are looking for a managed or hassle-free server operating system.
- There is no large room for customization.

MULTIPROCESSOR OS

- Within a single computer system, the usage of multiple central processing units (CPUs) is known as multiprocessor operating system.
- Such systems are used to process large volumes of data at a very high speed.



PERSONAL COMPUTER

- An operating system serving the general purpose (word processing, spreadsheets, web browser, digital media and others) of the end user is known as a personal computer operating system.

REAL TIME. OS

- The operating systems for serving real time application requests is known as real time operating systems.
- The real time operating system should be capable of responding and serving requests as and when they come in.
- Most RTOSes are designed to run on microcontrollers.
- As a result, they can often forgo complex user interfaces (e.g. no command line or GUI) in order to accomplish a few tasks at a time without needing to handle user input.
- Additionally, applications written for microcontrollers may have strict timing deadlines that need to be met,
  - such as firing a spark plug every so many milliseconds,
  - controlling a medical device to keep someone alive,
  - and fire thrusters on a satellite at a precise time to stay in orbit.

REAL TIME OS

Examples of Real-Time Operating Systems

- Below are some examples of the real-time operating system, such as:
  - The operating system of the microwave oven.
  - The operating system of the Washing machine.
  - The operating system of the aeroplane.
  - The operating system of digital cameras and many more.
- The quick response of the process is a must in real-time operating systems.
- There is no chance of any delay in completing any process because a little delay can cause several dangerous issues.

EMBEDDED. OS

- The compact and efficient resource utilizing operating systems specifically developed for embedded systems to perform specific
- specialized (OS) designed to perform a specific task for a device that is not a computer.
- The main job of an embedded OS is to run the code that allows the device to do its job
- makes the device's hardware accessible to software that is running on top of the OS
- An embedded system is a computer that supports a machine.
- It performs one task in the bigger machine.
- Examples include

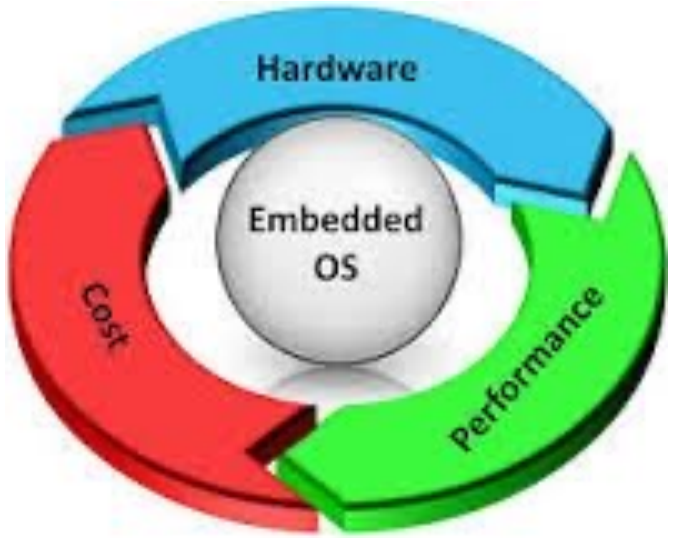
1.computer systems in cars, 2.traffic lights,

3. digital televisions, 4.ATMs,

5. airplane controls, 6. point of sale (POS) terminals,

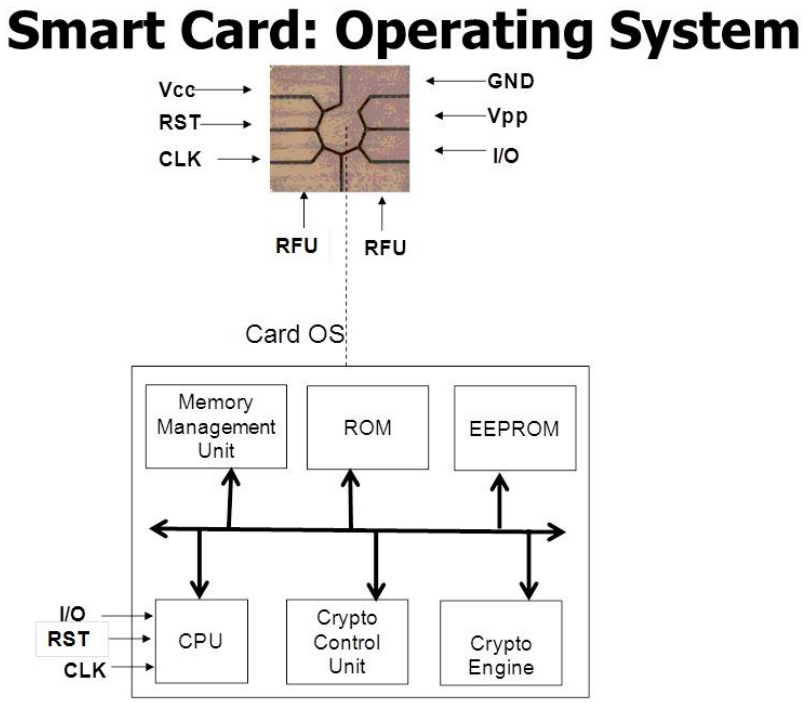
7. digital cameras, 8.GPS navigation systems,

9. elevators and Smart meters



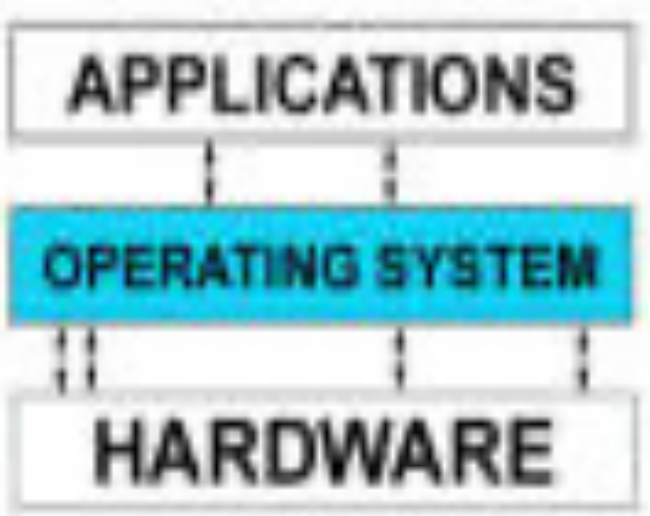
SMART CARD OS

- A smart card has an operating system which is specific to the hardware and provides the basic functionalities (storage, encryption and authentication).



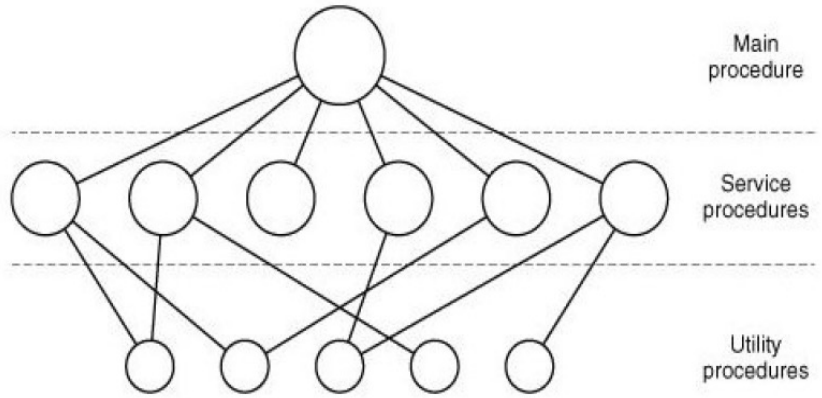
OPERATING SYSTEM STRUCTURES

- Monolithic Structure
- Layered Structure
- Virtual Machine Structure
- Exo-kernel Structure
- Micro-kernel Structure / Client-Server Structure

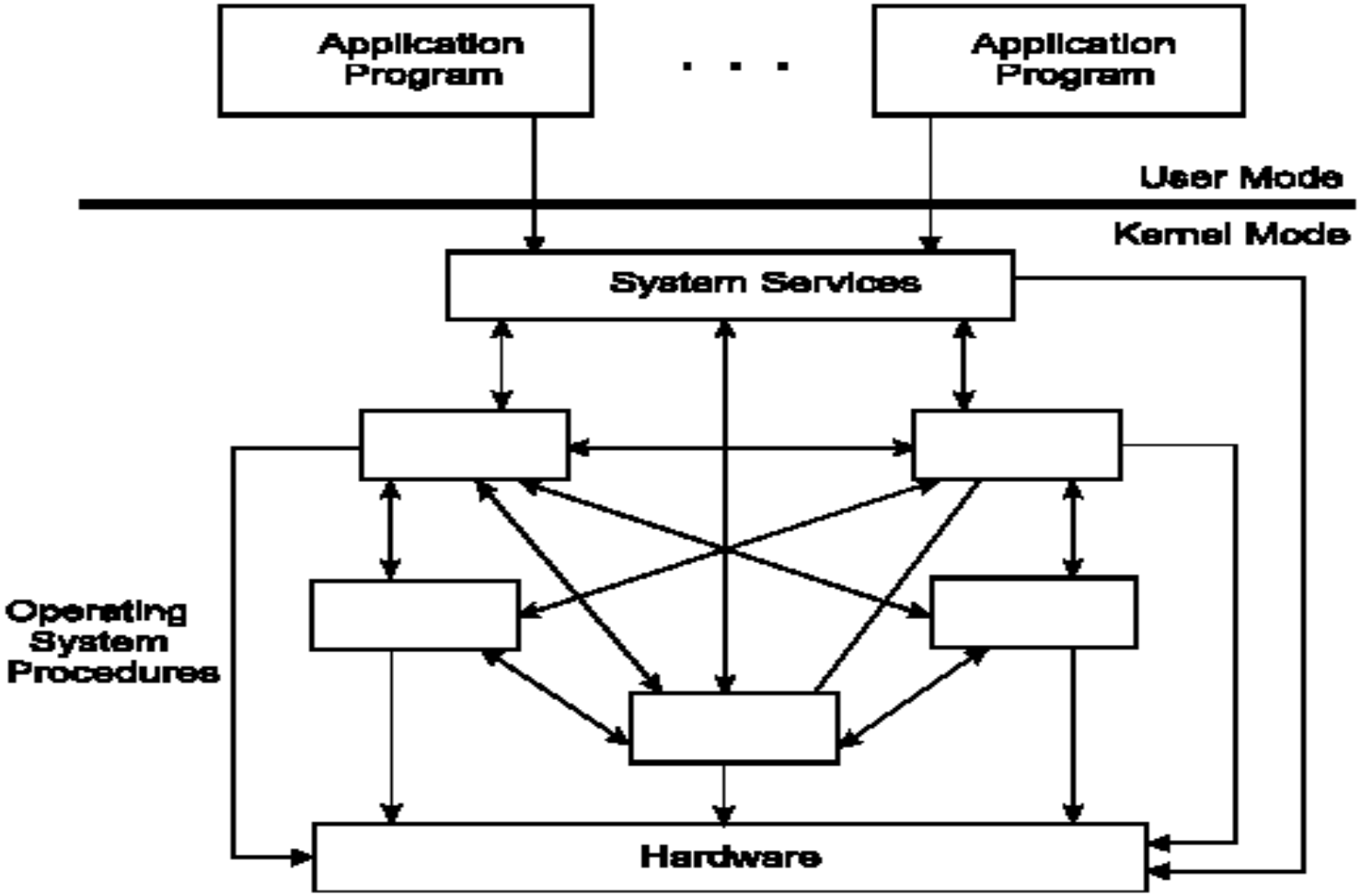


MONOLITHIC SYSTEM

- The components of monolithic OS are organized haphazardly and any module can call any other module without any reservation.
- Similar to the other operating systems, applications in monolithic OS are separated from the operating system itself.
- That is, 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 monolithic operating system structure with separate user and kernel processor mode
- There is no structure. The OS is written as a collection of procedures, each of which can call any of the other ones whenever it needs to.
- Examples: **CP/M and MS-DOS**



MONOLITHIC SYSTEM



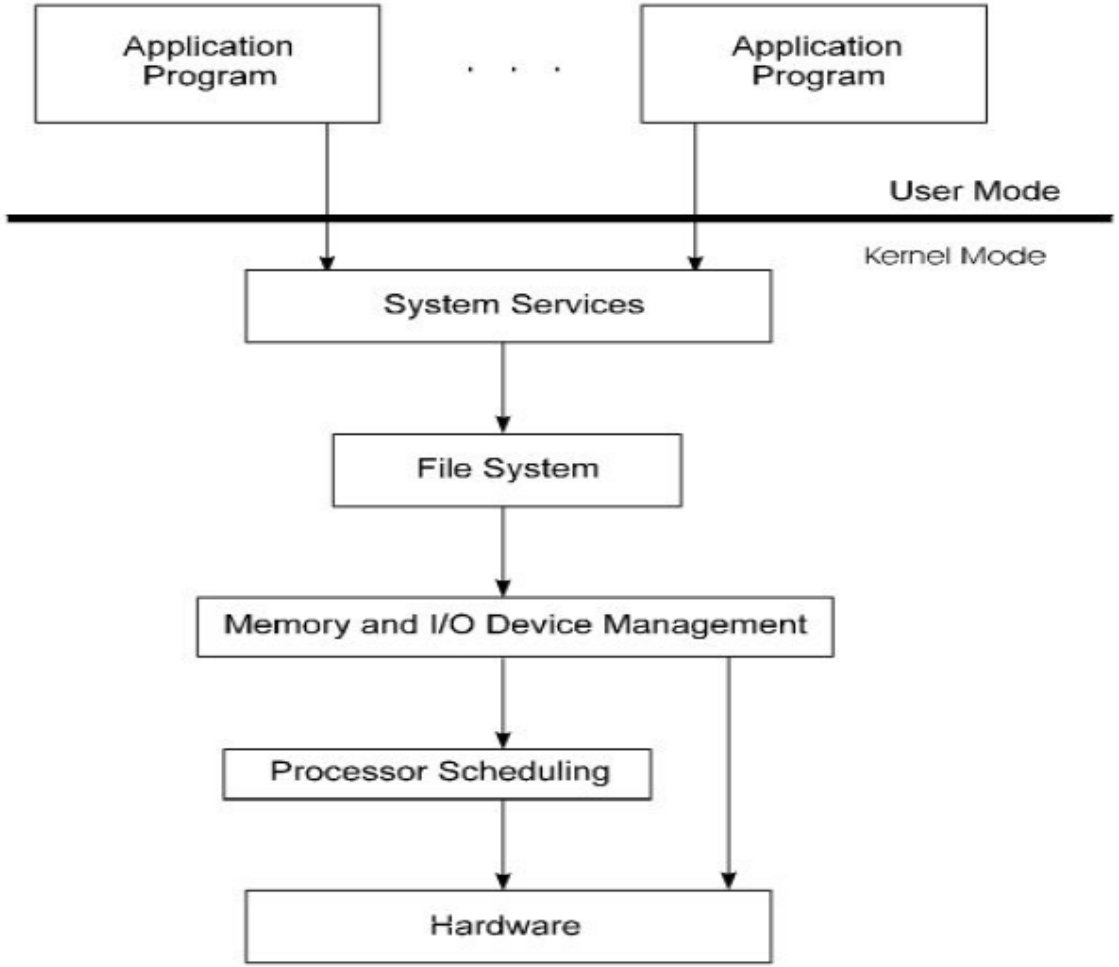
LAYERED OPERATING SYSTEM

- 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 (layer 0) 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.
- This approach simplifies debugging and system verifications.
- That is 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.

LAYERED OPERATING SYSTEM

- 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; one entire layer can be replaced without affecting other parts of the system.
- Example Systems: VAX/VMS, Multics, UNIX

LAYERED OPERATING SYSTEM



VIRTUAL MACHINES

- Virtual machine approach provides an interface that is identical to the underlying bare hardware.
- Each process is provided with a (virtual) copy of the underlying computer as shown in the fig.
- The resources of the physical computer are shared to create the virtual machine.
- CPU scheduling can be used to share the CPU and to create the appearance that users have their own processors.
- 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

VIRTUAL MACHINES

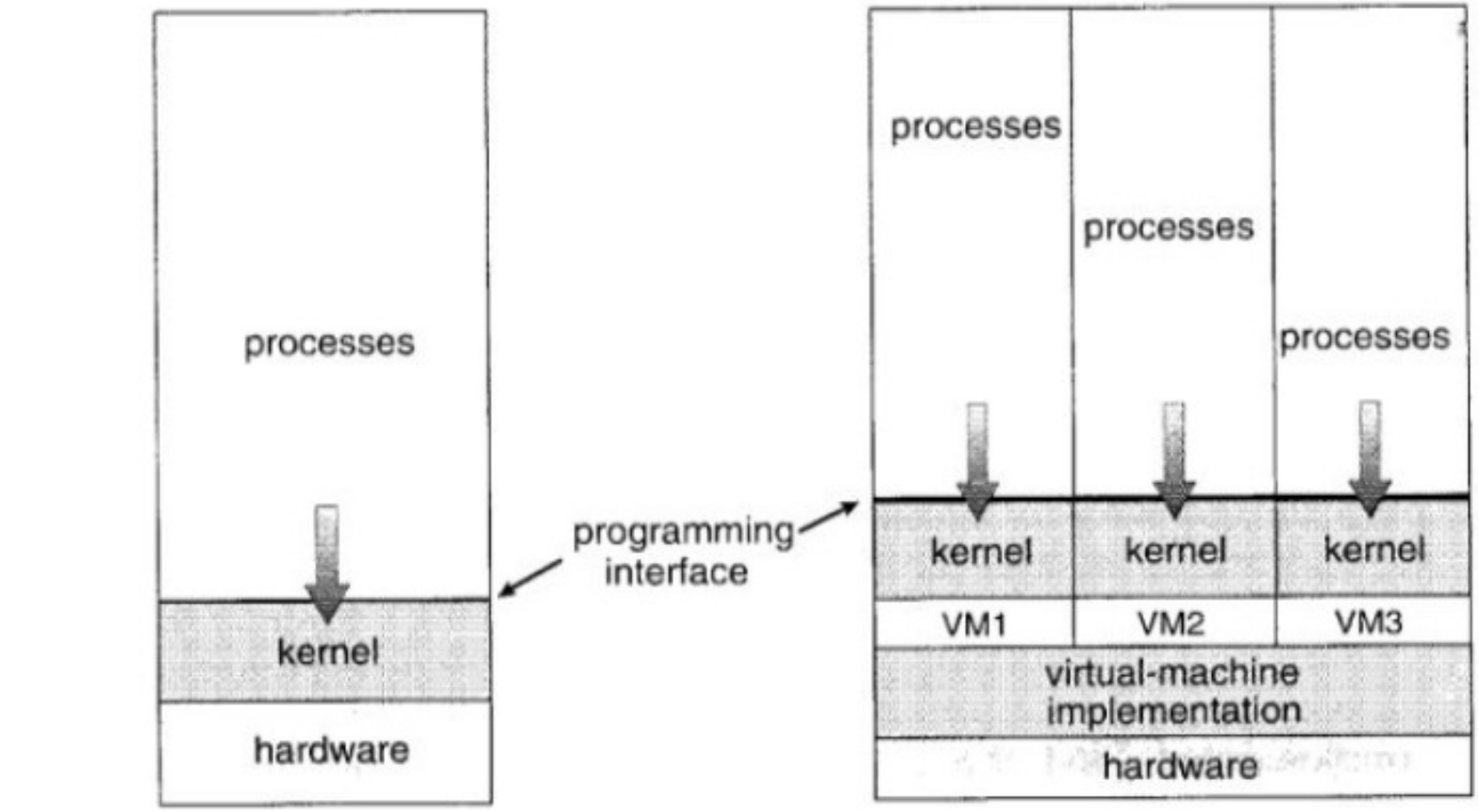


Fig: a). Non Virtual Machine

b). Virtual Machine.

CLIENT-SERVER OR MICROKERNEL

- The advent of 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 into several processes, each of which implements a single set of services - for example, I/O servers, memory server, process server, threads interface system.
- Each server runs in user mode, provides services to the requested client.

CLIENT-SERVER OR MICROKERNEL

- 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.

CLIENT-SERVER OR MICROKERNEL

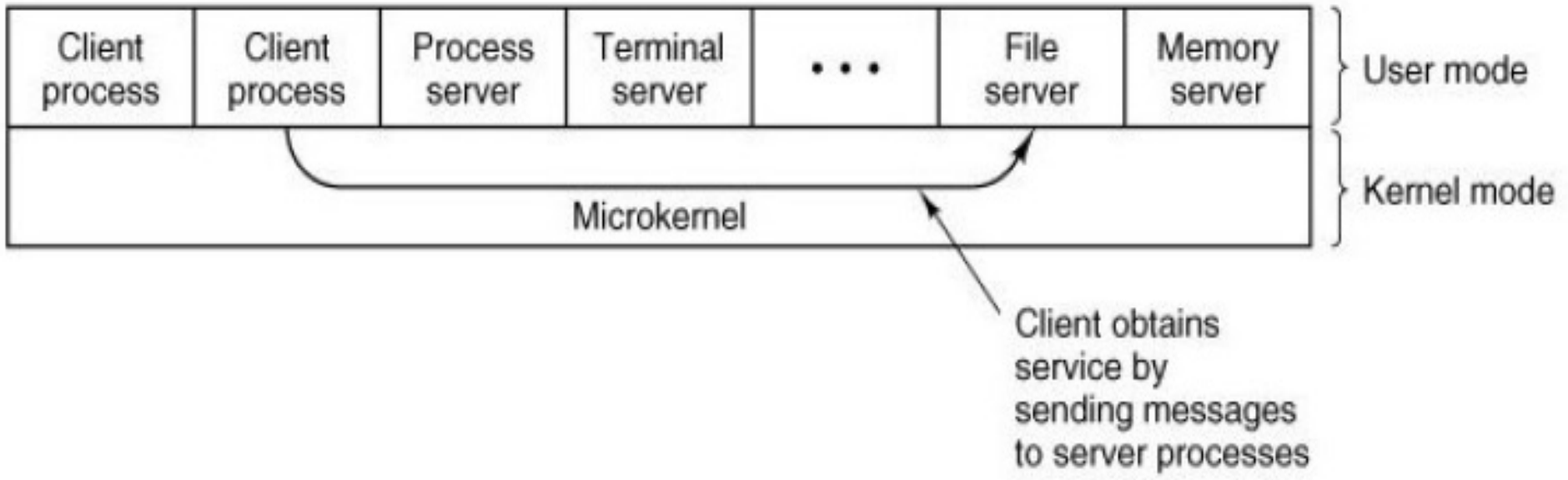


Fig: The client-server model.

CLIENT-SERVER OR MICROKERNEL

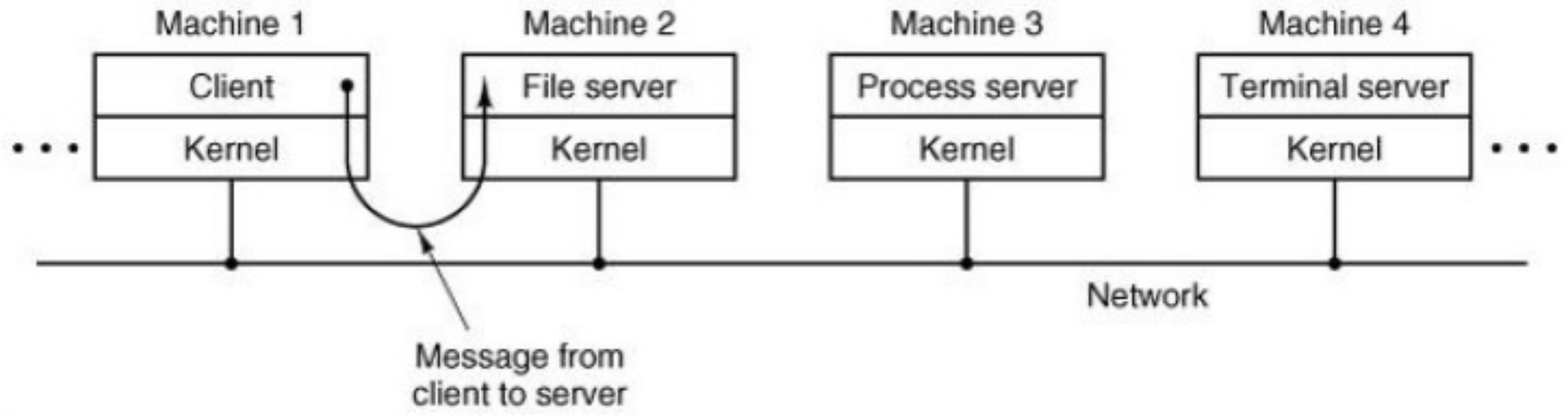


Fig: The client-server model in a distributed system.

SYSTEM CALLS

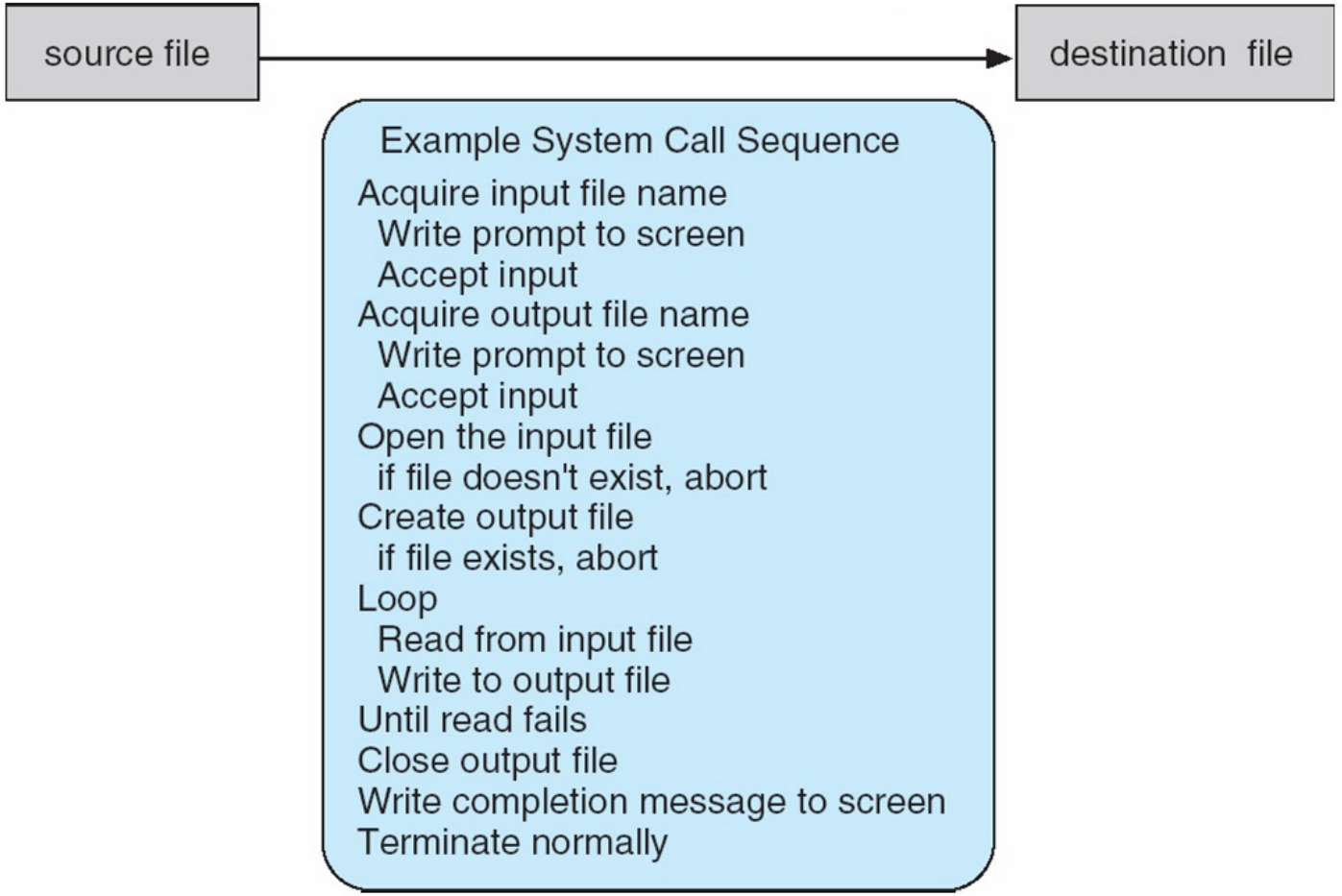
- The method used by a process to request action by the operating system:

1. After system call parameter preparations, it uses the trap instruction to transfer control to the requested service routine in the OS.
2. The system verifies that the parameters are correct and legal, and executes the request.
3. Returns control to the instruction following the system call.

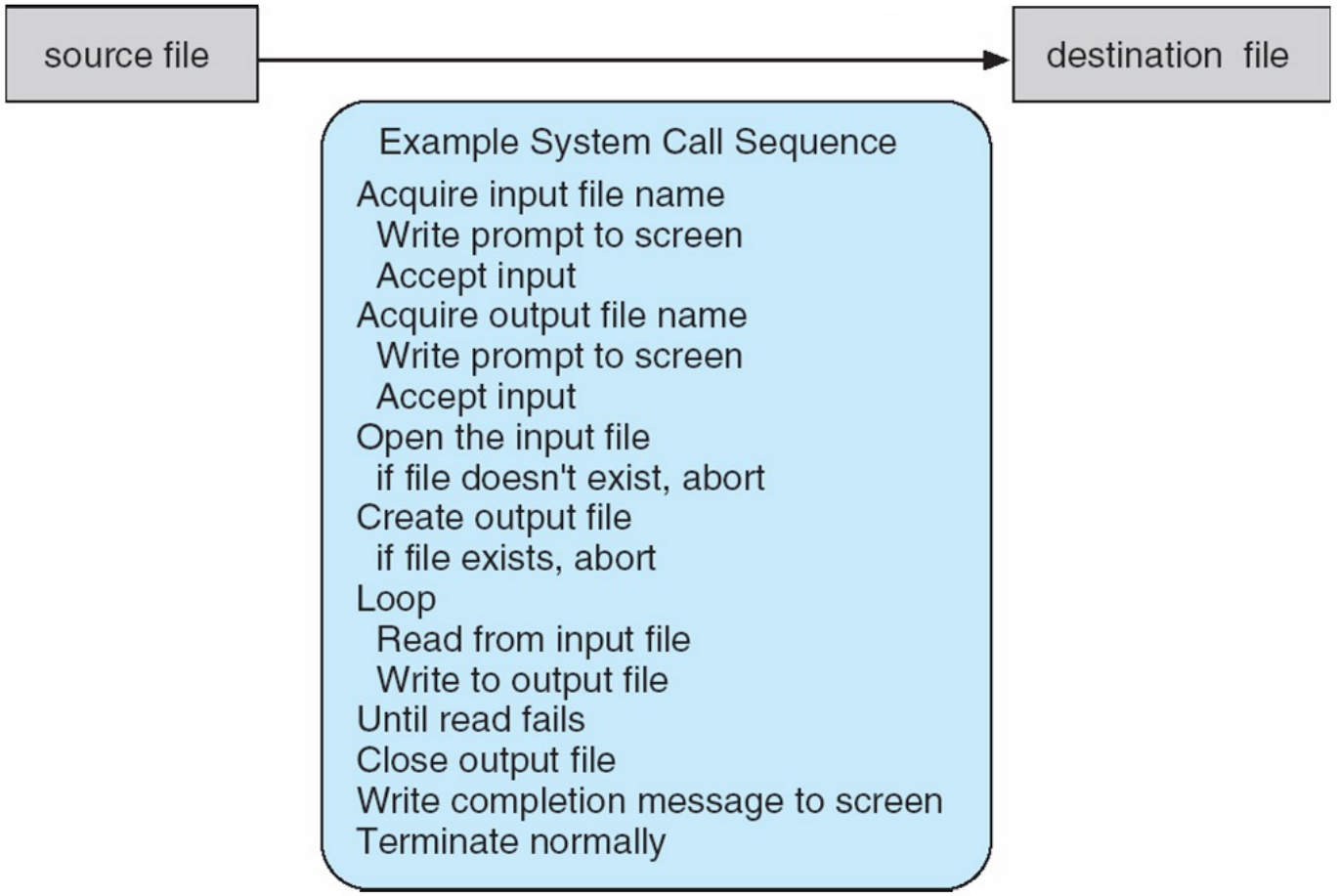
- For example, Linux has over 300 different calls.

EXAMPLE OF SYSTEM CALLS

- System call sequence to copy the contents of one file to another file



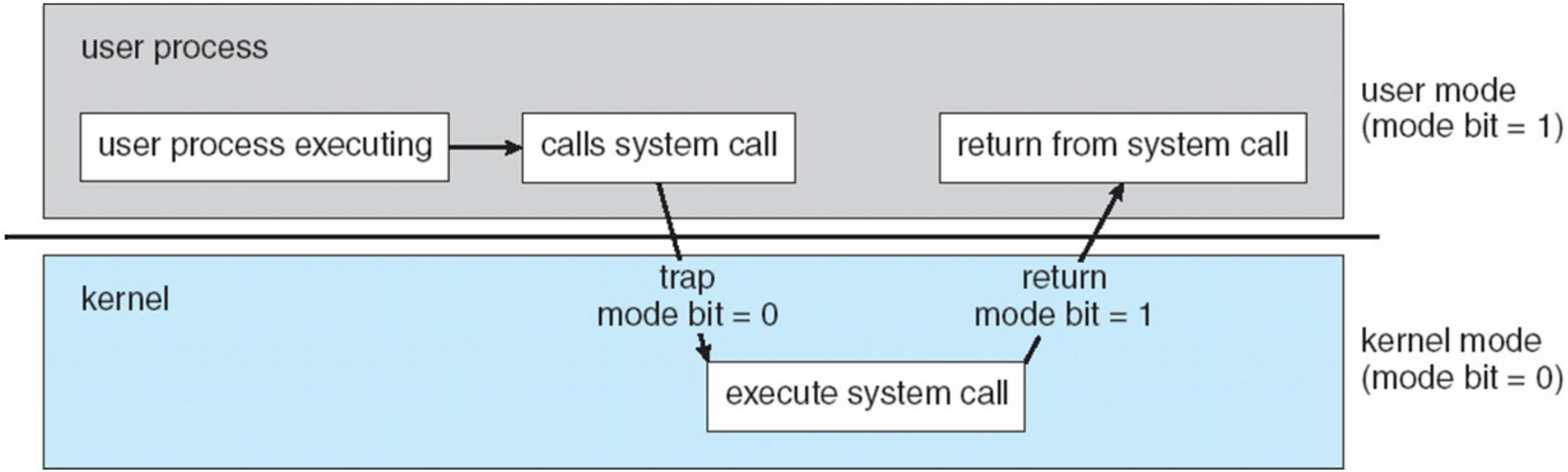
EXAMPLE OF STANDARD API



HANDLING OF SYSTEM CALLS

- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result call
  - Most details of OS interface hidden from programmer by API
    - Managed by run-time support library (set of functions built into libraries included with compiler)

HANDLING OF SYSTEM CALLS



SYSTEM CALLS FOR PROCESS

- create process, terminate process
- end, abort
- load, execute
- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory
- Dump memory if error
- Debugger for determining bugs, single step execution
- Locks for managing access to shared data between processes

SYSTEM CALLS FOR FILE

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

SYSTEM CALLS FOR DEVICE MANAGEMENT

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

SYSTEM PROGRAMS

- System programs provide a convenient environment for program development and execution. They can be divided into:
  - File manipulation
  - Status information sometimes stored in a File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Background services
  - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls

SYSTEM PROGRAMS (CONTD..)

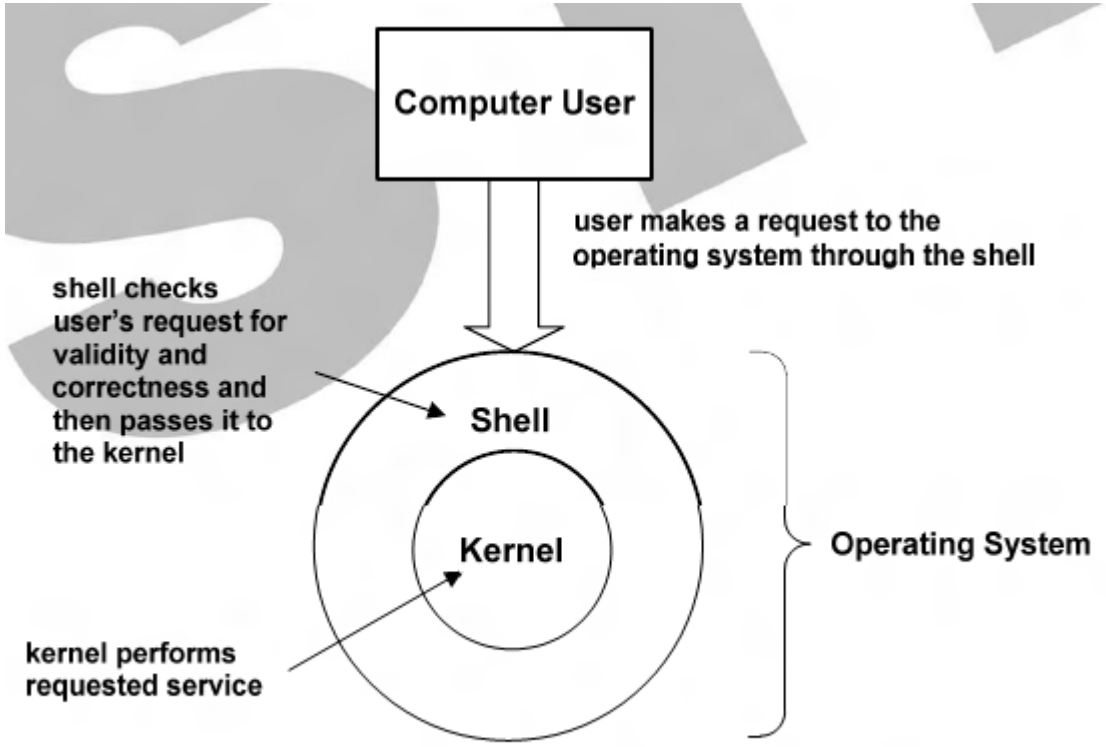
- **File modification**
  - Text editors to create and modify files
  - Special commands to search contents of files or perform transformations of the text
- **Programming-language support** - Compilers, assemblers, debuggers and interpreters sometimes provided
- **Program loading and execution**- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- **Communications** - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - Allow users to send messages to one another' s screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

SYSTEM PROGRAMS (CONTD..)

- **Background Services**
  - **Launch at boot time**
    - **Some for system startup, then terminate**
    - **Some from system boot to shutdown**
  - **Provide facilities like disk checking, process scheduling, error logging, printing**
  - **Run in user context not kernel context**
  - **Known as services, subsystems, daemons**
- **Application programs**
  - **Don't pertain to system**
  - **Run by users**
  - **Not typically considered part of OS**
  - **Launched by command line, mouse click, finger poke**

THE SHELL

- A shell is a program that provides the traditional text only user interface for Linux and other Unix operating system.
- Its primary function is to read commands typed into a console or terminal window and then execute it.
- The term shell derives its name from 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: , Bash shell, Korn Shell, C shell



The Kernel and the Shell

OPEN-SOURCE OPERATING SYSTEMS

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), VirtualBox (open source and free on many platforms
- Use to run guest operating systems for exploration

Thank You