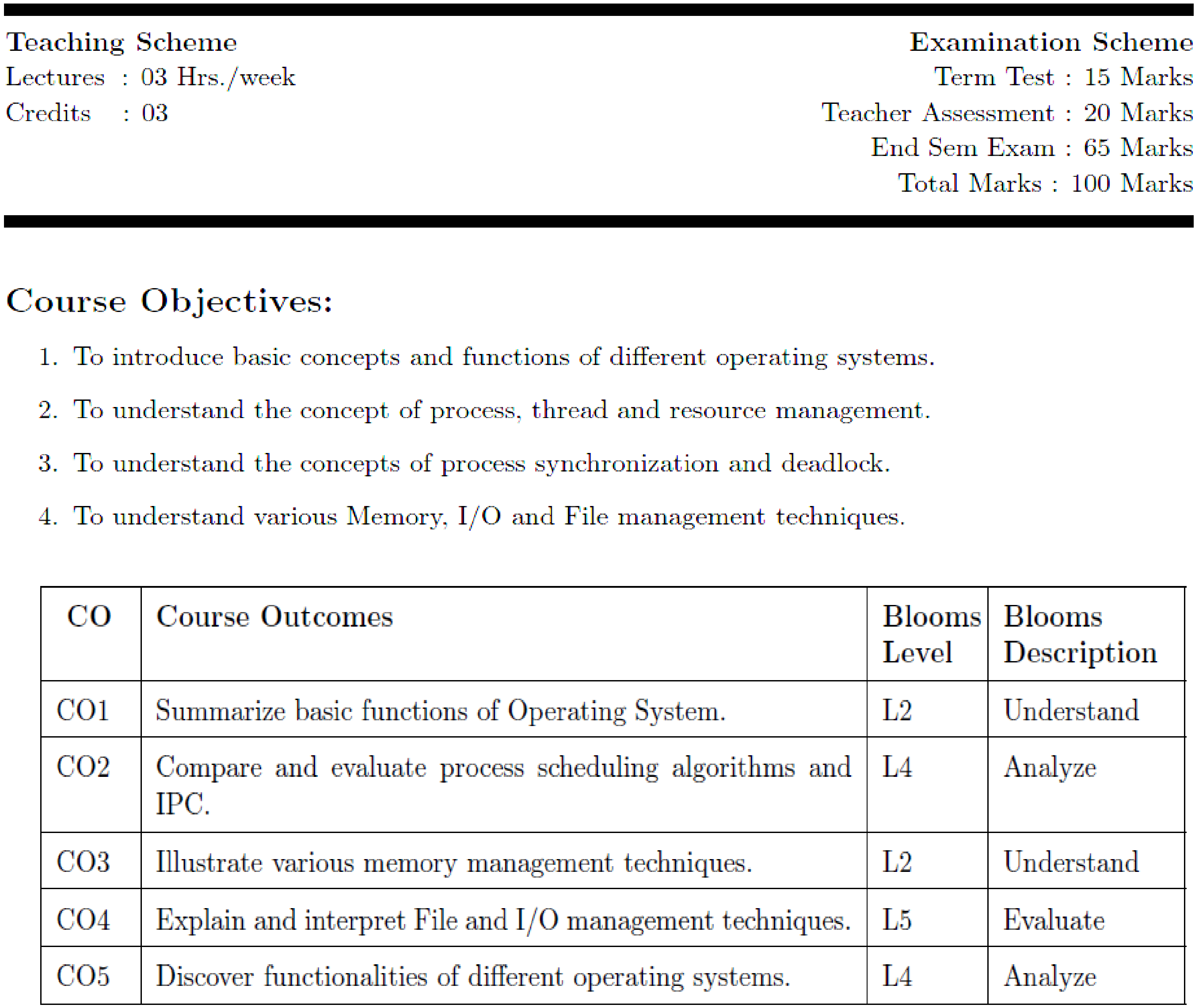
Operating System

Operating System (PCCO4030T)



Teaching Scheme		**Examination Scheme**	
Lectures	03 Hrs./week	Term Test	15 Marks
Credits	03	Teacher Assessment	20 Marks
		End Sem Exam	65 Marks
		Total Marks	100 Marks

Course Objectives:

1. To introduce basic concepts and functions of different operating systems.
2. To understand the concept of process, thread and resource management.
3. To understand the concepts of process synchronization and deadlock.
4. To understand various Memory, I/O and File management techniques.

CO	Course Outcomes	Blooms Level	Blooms Description
CO1	Summarize basic functions of Operating System.	L2	Understand
CO2	Compare and evaluate process scheduling algorithms and IPC.	L4	Analyze
CO3	Illustrate various memory management techniques.	L2	Understand
CO4	Explain and interpret File and I/O management techniques.	L5	Evaluate
CO5	Discover functionalities of different operating systems.	L4	Analyze

Text Books:

1. **William Stallings**, O**perating System: Internals and Design Principles**, 8th Edition, Prentice Hall, 2014.
2. Abraham Silberschatz, Peter Baer **Galvin** and Greg Gagne, **Operating System Concepts**, 9th Edition, John Wiley & Sons, Inc., 2016.
3. Andrew **Tannenbaum**, **Operating System Design and Implementation**, 3rd Edition, Pearson, 2015.

Reference Books:

1. Maurice J. Bach, Design of UNIX Operating System, 2nd Edition, PHI, 2004.
2. Achyut Godbole and Atul Kahate, Operating Systems, 3rd Edition, McGraw Hill Education, 2017.
3. The Linux Kernel Book, Remy Card, Eric Dumas, Frank Mevel, 1st Edition, Wiley Publications, 2013.

Unit-I Introduction to Operating System

Operating System Objectives and Functions, Evolution of Operating System, OS Design Considerations for Multiprocessor Architectures, Operating System Structures, System Calls.

CO 1: Summarize basic functions of Operating System.

Operating System Objectives and Functions

- **Definition:**

  It is software or program that acts as an **interface** between user of computer system and computer hardware.

- **Objectives:**

  An OS is a program that **controls** the execution of application programs and acts as an interface between applications and the computer hardware.

  An OS has **three objectives**:

  - **Convenience:** To make a computer more convenient to use.
  - **Efficiency:** To allow the computer system resources to be used in an efficient manner.
  - **Ability to evolve:** To permit the effective development, testing and introduction of new system functions without interfering with service.

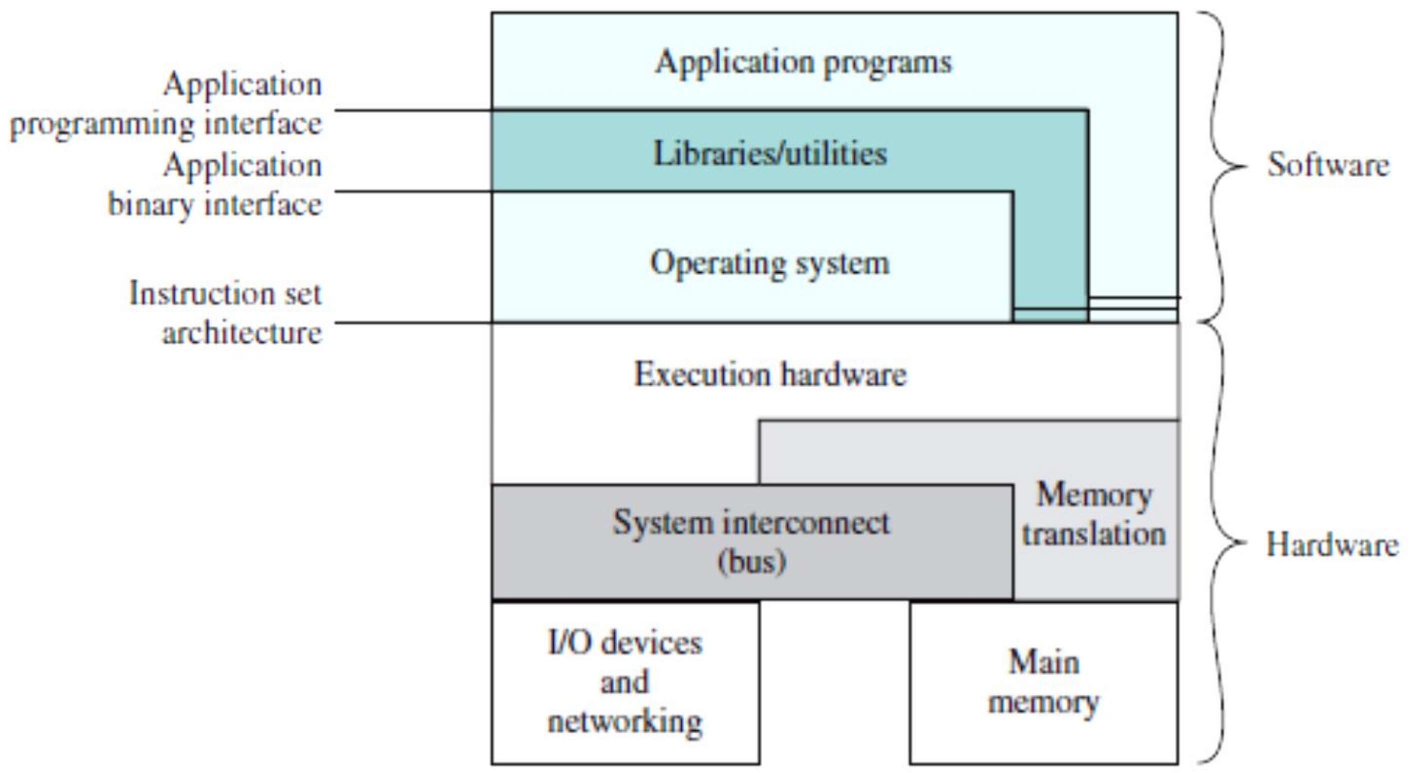
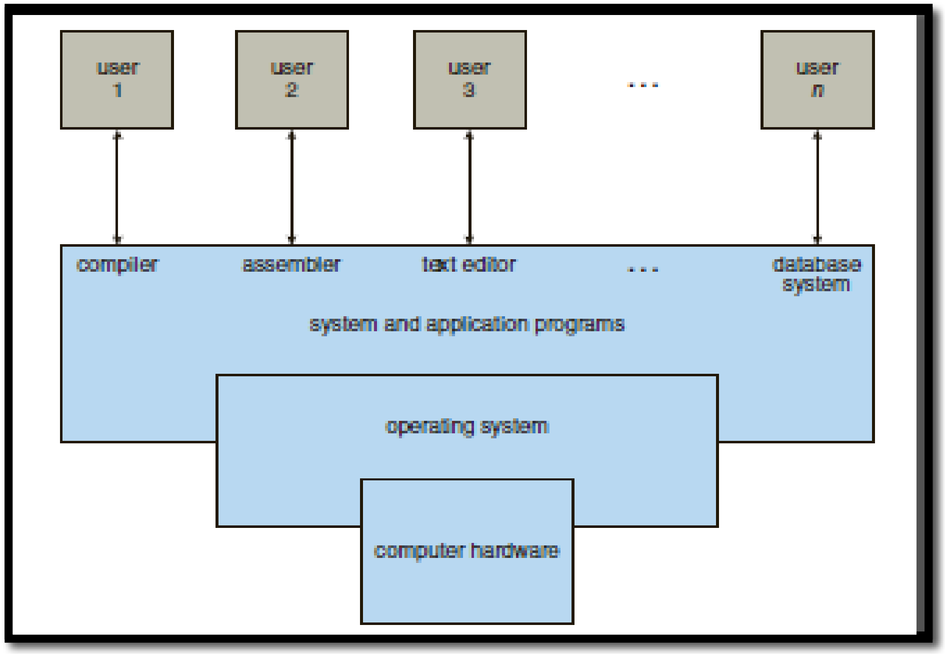

Application programming interface
Application binary interface
Instruction set architecture
Application programs
Libraries/utilities
Operating system
Execution hardware
Memory translation
System interconnect (bus)
I/O devices and networking
Main memory
Software
Hardware

Figure: Computer Hardware and Software Structure



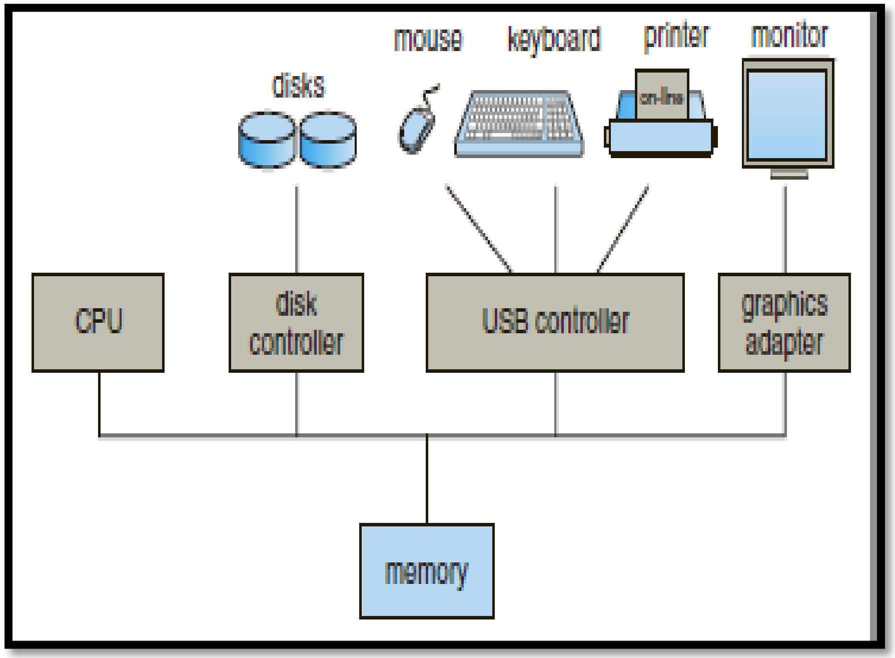


Fig: Abstract View of System Components

Figure: A Modern Computer System

- **OS typically provides services in the following areas:**

1. **Program development:** The OS provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs.
2. **Program execution:** A number of steps needed to be performed to execute a program like:
   - Loading instructions and data into main memory
   - Initialize I/O devices and files
   - Prepare other resources

   The OS handles these scheduling duties for the user.
3. **Access to I/O devices:**

   Each I/O device requires its own set of instructions or control signals for operation.

   The OS provides a uniform interface (by hiding the details) to such devices using simple reads and writes.

4. Controlled access to files:

OS **provides file access** according to the nature of the I/O device (disk drive, tape drive) & the structure of the data contained in the files on the storage medium.

For a system with multiple users, the OS may provide protection mechanisms to control access to the files.

5. System access:

For shared or public systems, the **OS controls access** to the system as a whole and to specific system resources.

The access function must :

- provide protection of resources and data from unauthorized users
- resolve conflicts for resource contention/dispute

6. Error detection and response:

A variety of errors can occur while a computer system is running.

Internal and external **Hardware errors** – eg: memory error or a device failure or malfunction

Software errors – eg: division by zero, attempt to access forbidden memory location

OS must provide a response that clears the error condition with the least impact on running applications.

7. Accounting:

OS collects **usage statistics** for various resources and monitors the performance parameters such as response time.

This information is useful for future enhancements and in tuning the system to improve performance.

(**keeps track of which user use how much & what kind of resource**)

- **The Operating System as Resource Manager**

- A **computer system is a set of resources** for the movement, storage, and processing of data and for the control of these functions.
- The OS is **responsible for** managing these resources.
- By managing the computer's resources, the **OS has control** of the computer's basic functions.
- The OS functions in the same way as ordinary computer software i.e. it is **a program or set of programs** executed by the processor.
- The OS frequently **surrenders control** and must **depend on the processor** to allow it to regain control.
- The OS **directs the processor** in the use of the other system resources and in the timing of its execution of other programs.

Evolution of Operating System

❖ **Serial Processing** (late 1940s to mid-1950s)

- OS can perform its work serially.
- The programmer interacts directly with the computer hardware.
- Typical serial execution sequence.
  - Editor program is loaded.
  - Language translator is loaded with source code.
  - For Syntax error- whole process restarts.
  - Loading & execution of corrected code.
  - Debugging of runtime error (bug).
- These early systems presented two main problems:
  - **Scheduling:** An user could sign up for a block of time in multiples of a half hour or so.
  - **Setup time:** Each of these **steps** for single program/job execution involve mounting or dismounting tapes or setting up card decks.

❖ **Simple Batch Systems** (mid-1950s)

- The central idea behind the simple batch-processing scheme is the use of a piece of software known as the monitor.
- 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**,
  - Further **similar jobs are placed in a batch** together sequentially and the entire batch is placed **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.
  - With each job, instructions are included in **job control language (JCL)** which is a special type of programming language used to provide instructions to the monitor.

Continued…

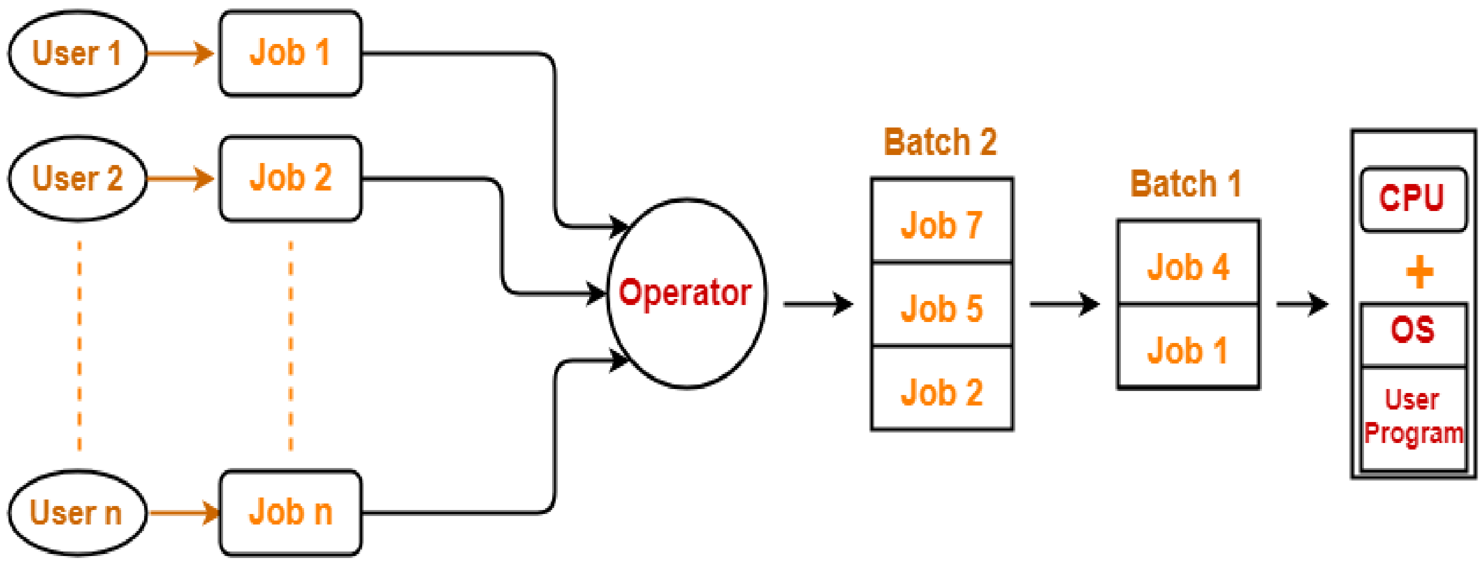


Figure: Batch O.S.

❖ Multiprogrammed Batch Systems

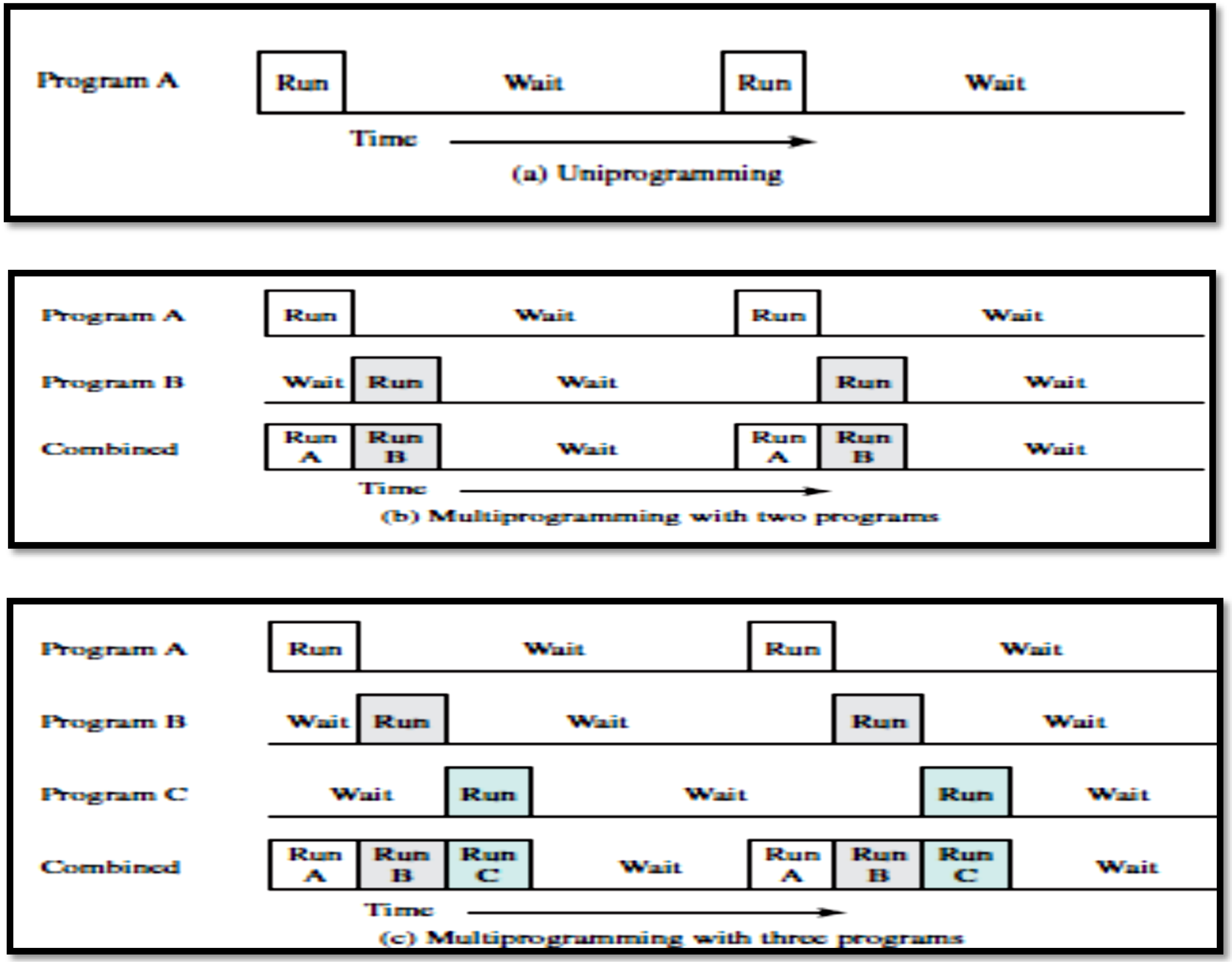


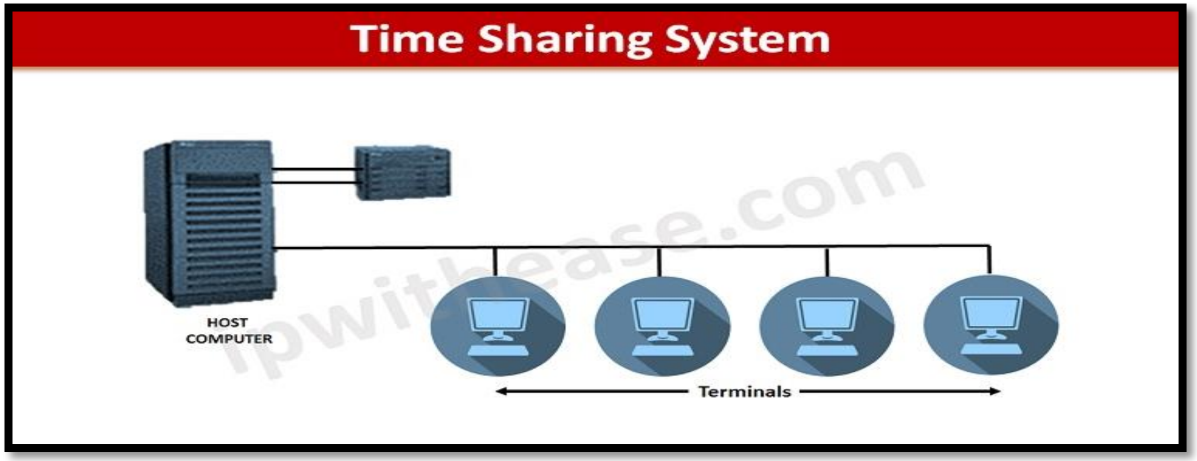
Figure: Multiprogramming Example

- With the **use of multiprogramming**, batch processing can be quite efficient.
- When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O.
- Furthermore, **memory is expanded** to hold three, four or more programs and switch among all of them.
- The approach is known as **multiprogramming / multitasking** .
- It is the **central theme** of modern operating systems.
- Thus the multiprogramming OS :
    - Supports multiple programs.
    - Increase resource utilization.
    - Most programs oscillates between CPU & I/O phase.

❖ Time-Sharing Systems

- For many jobs, it is desirable to provide a mode in which the **user interacts directly with the computer.**
- Some jobs **need an interactive mode essentially**.
- 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.
- At regular time intervals (say time slicing), the **current user would be preempted** and another user loaded in.
- To preserve the old user program status for later resumption, the old user programs and data were **written out to disk** before the new user programs and data were read in.
- Subsequently, the old user program code and data were restored **in main memory** when that program was next given a turn.

Continued…



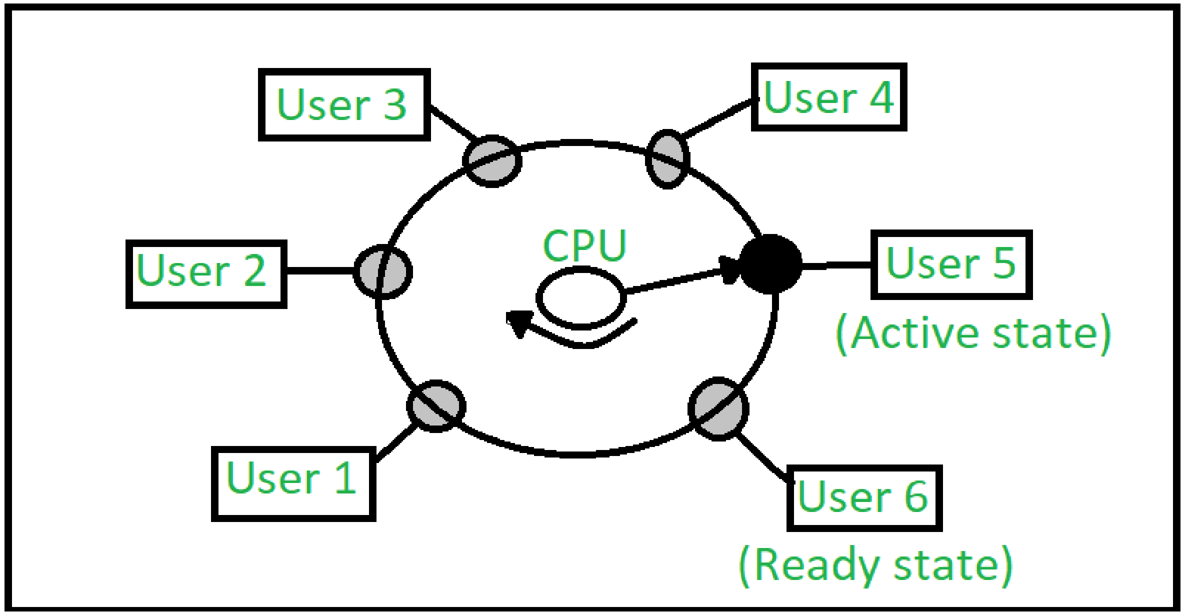


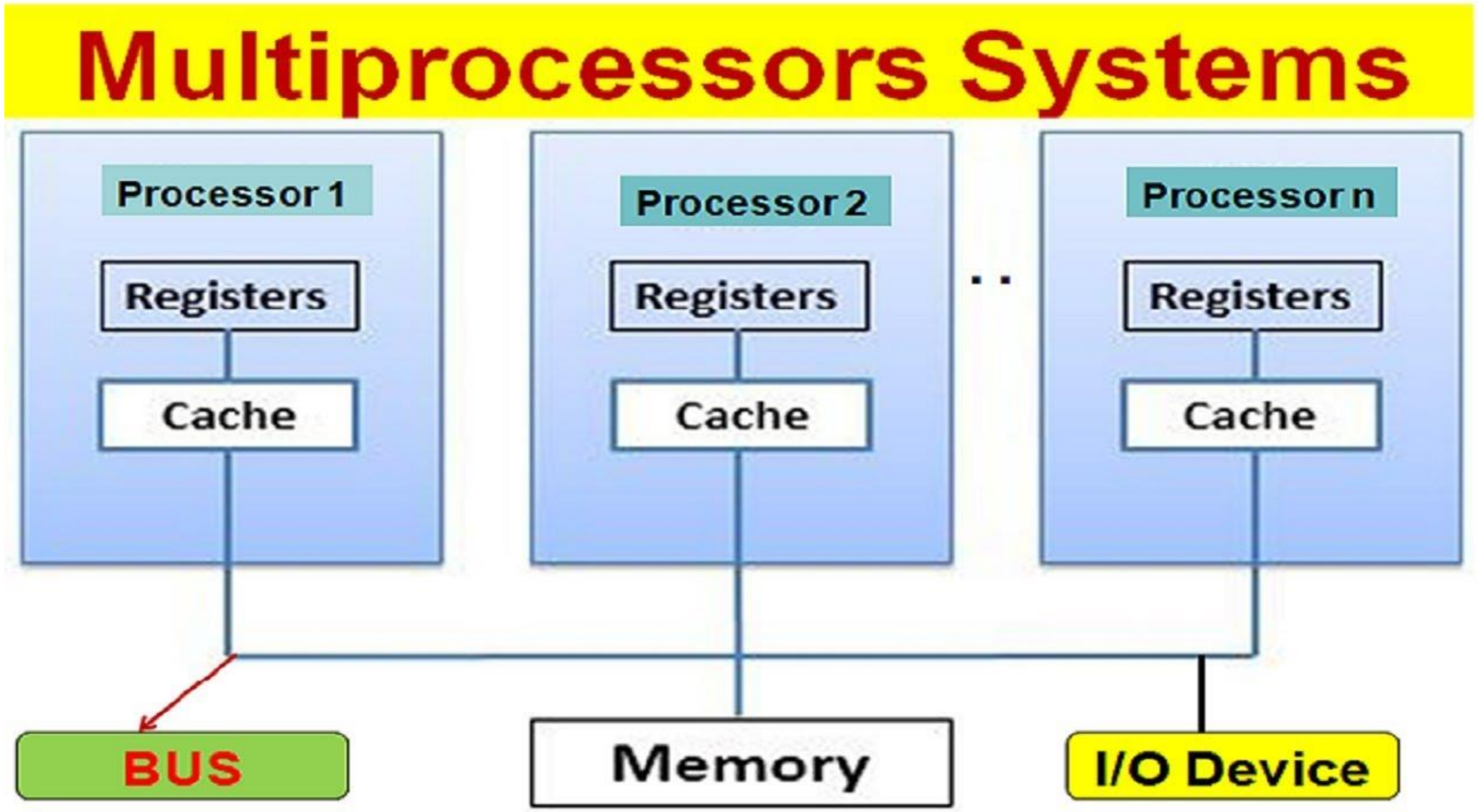
Figure: Time Sharing System

OS Design Considerations for Multiprocessor Architectures

Symmetric Multiprocessor (SMP) OS Considerations:

- In a SMP system, the **kernel can execute on any processor** and typically each processor does **self-scheduling** from the pool of available processes or threads.
- The kernel can be constructed as multiple processes or multiple threads, allowing portions of the **kernel to execute in parallel**.
- A SMP operating system **manages processor and other computer resources** so that the user may view the system in the same fashion as a multiprogramming uniprocessor system.
- A multiprocessor OS must **provide all the functionality** of a multiprogramming system **plus additional features** to accommodate multiple processors.

Continued…


Multiprocessors Systems
Processor 1
Registers
Cache
Processor 2
Registers
Cache
. .
Processor n
Registers
Cache
BUS
Memory
I/O Device

Continued…

The **key design issues** include the following:

- **Simultaneous concurrent processes or threads**: Allows several processors to execute the same kernel code simultaneously.
- **Scheduling:**

  Any processor may perform scheduling.
- **Synchronization:**

  With multiple active processes, care must be taken to provide **effective synchronization**.
- **Memory management:**

  Memory management on a multiprocessor **must deal with** all of the issues found on uniprocessor computers
- **Reliability and fault tolerance**

  The OS should **provide recovery** in case of hardware and software failure.

- **Types of O.S.**
- **Types with respect to four aspects:**

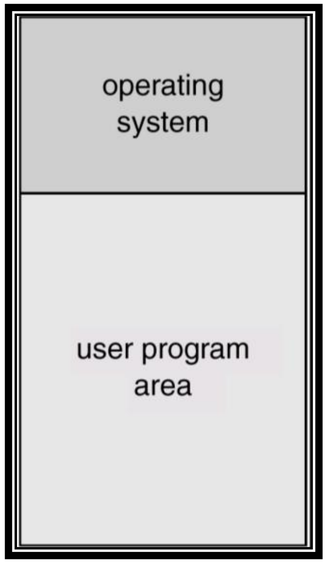
  a) Process scheduling
  b) Memory management
  c) I/O Management
  d) File Management

- **Batch O.S.**

✓ Similar jobs are placed in a batch.

✓ Turnaround time for a job is more.

- Requires program, data & some commands to be submitted together in form of job.
- No human intervention.
- Memory divided in two areas:



- No critical device management.
- Little protection & no concurrency control required for file.

- **Multiprogramming O.S.**

- ✓ Support multiple programs.
- ✓ Increase resource utilization.

- ▪ Various scheduling algorithms are used.
- ▪ Memory management & protection is required.
- ▪ Device scheduling (eg: disk scheduling)
- ▪ Various rights to file for file management.

- **Time Sharing O.S.**

- ✓ Logical extension of multiprogramming.
- ✓ Switching between the jobs is done frequently.
- ▪ Round robin scheduling algorithm.
- ▪ Memory management provides isolation & protection.
- ▪ I/O management handles multiple users & devices.
- ▪ Different file management techniques can be used.

- **Real Time O.S.**(**<u>RTOS</u>**)

- Supports real time applications.
- Characterized by supporting immediate response.
- Applications:
  - Nuclear reactors / scientific experiments
  - Weather forecasting system
  - Controlling traffic signal
  - Home appliances
  - Defense application like RADAR
- **The primary functions of the RTOS are to:**

1. **Manage** the processor and other system resources to meet the requirements of an application.
2. **Synchronize** with and **respond** to the system events.
3. **Move** the data efficiently among processes and to perform coordination among these processes.

Continued…

- **Types:**

1. Hard RTOS –

- Processing under all conditions.
- The deadline should not be crossed.

  e.g. embedded Linux

  Airbag control in cars

  Anti-lock brake

  Engine control system

2. Soft RTOS –

- A process might not be executed in given deadline (but doesn't harm the system/application).
- Can't guarantee to meet requirements.

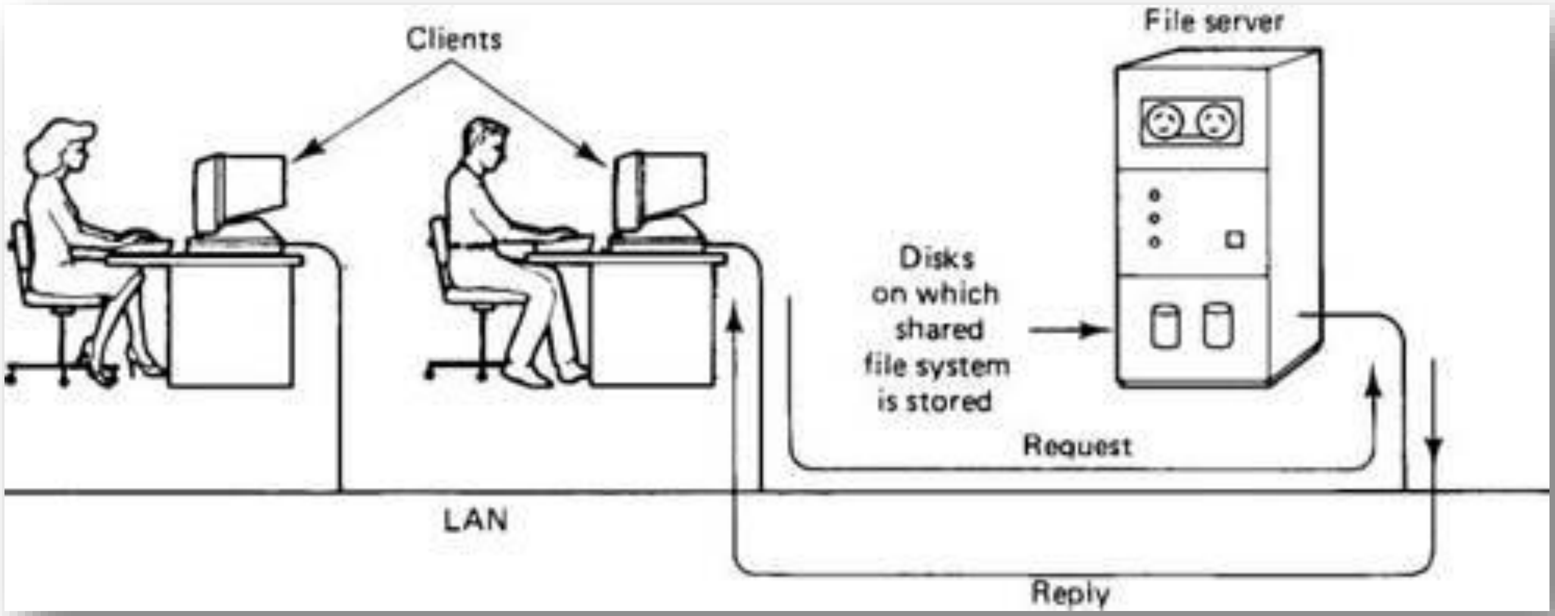
  e.g. RT Linux, µC Linux

  digital camera,

  mobile phones

- **Distributed OS**

✓ In a distributed operating system, users access remote resources in the same way they access local resources.



- **Data Migration**
  - Data/file migration:
    either file or only needed portions of file is transferred to destination

- **Computational Migration**
  - Computation/result migration

- **Process Migration**
  - Logical extension of computation migration.
  - Entire process or its part may be executed at different site.

Operating System Structures

- **Structures of O.S.**

- OS can be classified based on their structuring mechanism.

- The structure of OS depends on the way in which the components are interconnected and melded into a kernel.

Continued…

O.S. with Monolithic Structure:

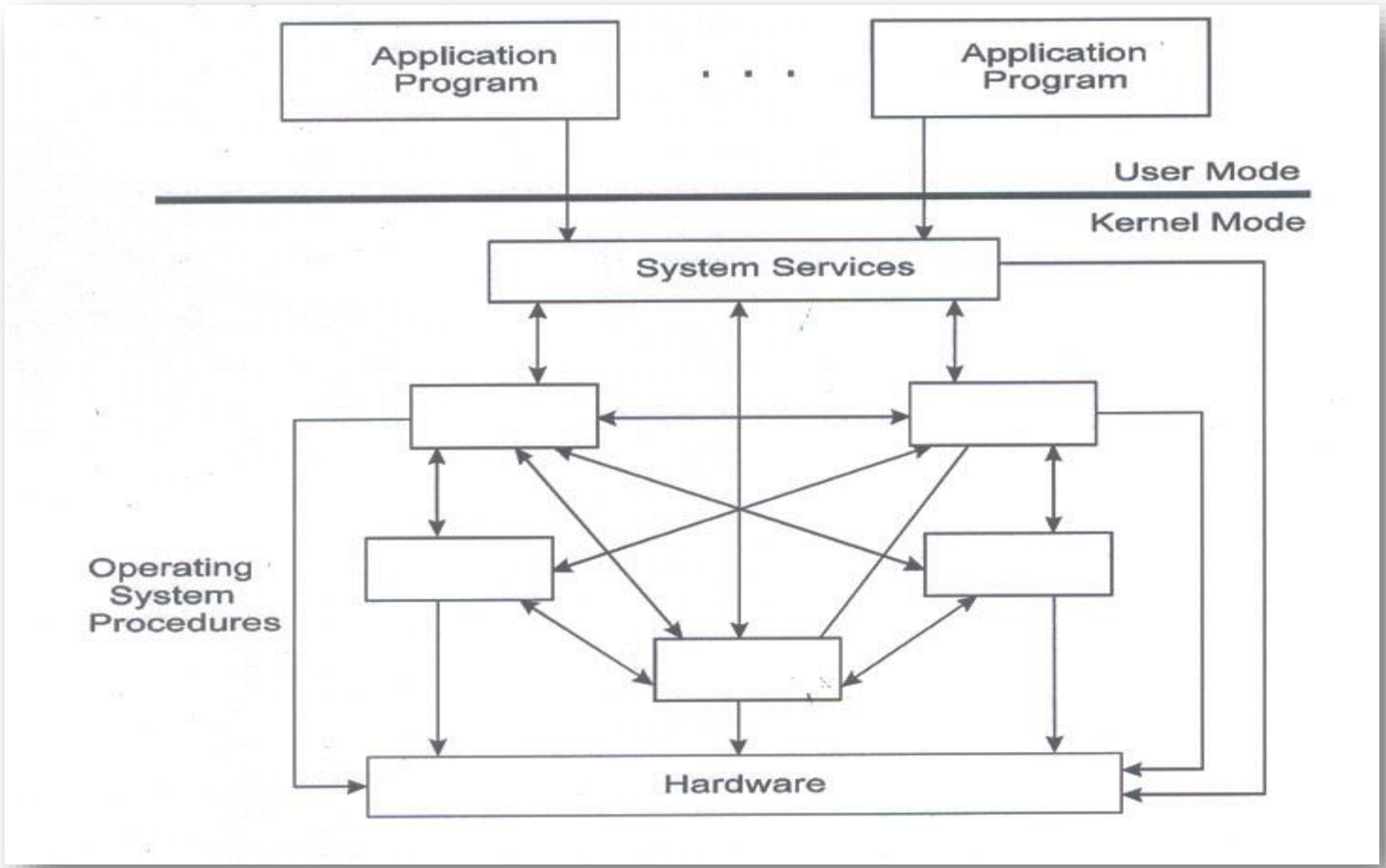


Figure: Monolithic Structure of Operating System

- Applications are separated from OS
- OS runs in kernel mode & applications run in user mode
- OS is a piece of block composed with different modules
- Difficult to debug, modify, and upgrade
- Entire OS resides in main memory
- Due to multiple connections:
  - s/w becomes disordered
  - Impact of one module on another
  - Distributes OS in one or more ways in multiple processors
- Example – MS DOS, Unix

Layered Approach of O.S.

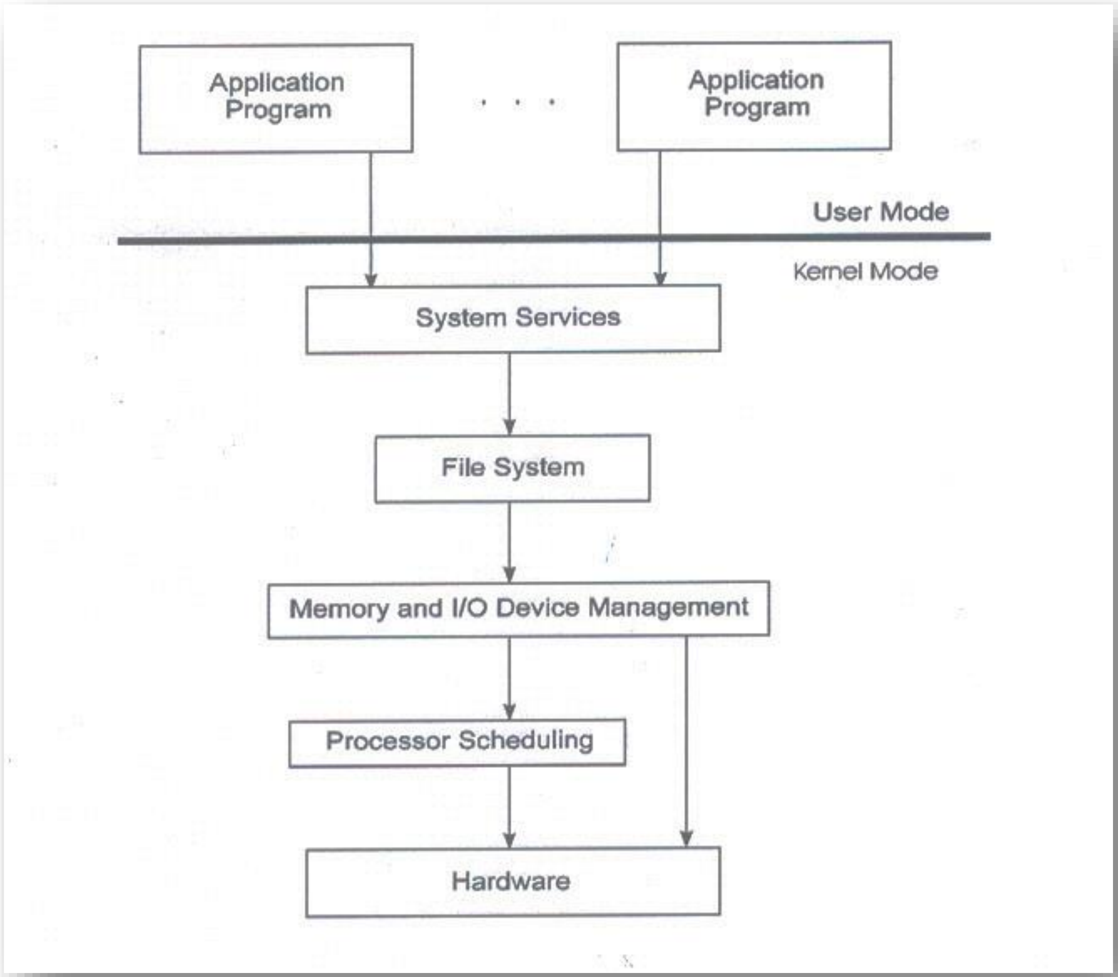
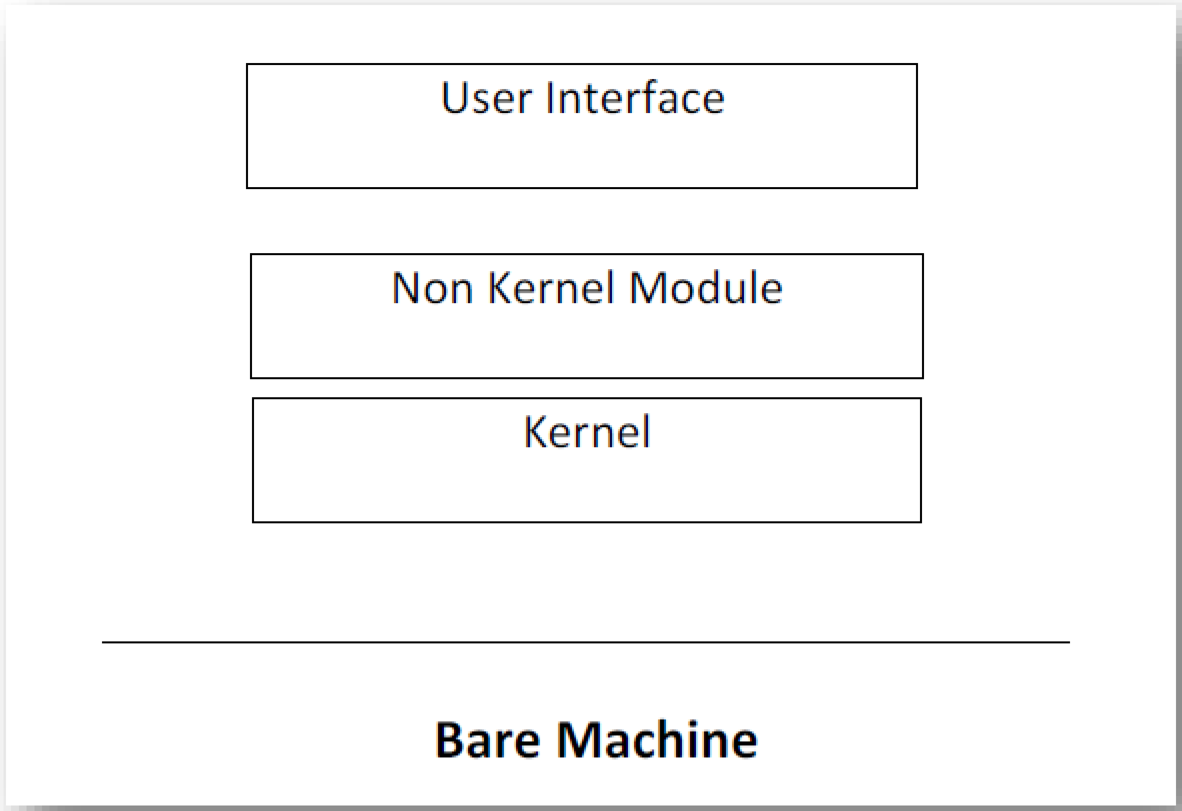


Figure: Layered Operating System

- OS is broken into no of layers/levels.
- Bottom layer is h/w & highest is user interface.
- A layer is encapsulation of data & operations to manipulate data.
- Information hiding between layers increases the security and protection.
- **A layer can use operations & services of lower level layer only.**
- Simplifies debugging.
- Careful definition of each layer is essential.
- Less efficient than other types.
- e.g. Multics, Unix.

- **Kernel Based O S**



Continued…

- OS is made up of several separated/independent modules and only kernel resides in main memory.
- The operation of OS is interrupt driven.
- OS gets control:
  - When an interrupt signal occurs. e.g.- timer, I/O interrupts etc.
  - When a non-kernel program makes a system call to invoke kernel function or service.
- Kernel based OS structure is convenient in coding non-kernel programs.
- It is portable.
- Kernel is monolithic to ensure efficiency.
- e.g. Linux, BSD, Mac OS

Continued…

- Disadvantages:
  - May suffer stratification (formation of layers).
  - Offers poor extensibility.
- Typical mechanisms in a kernel are as follows:

Sr. No	Function	Mechanism example
1	Interrupt processing	i) Saves CPU state
2	Scheduling	i)Dispatch a program ii)Manipulate scheduling list
3	Memory management	i)Set memory protection information ii)Swapping in & out iii)Virtual memory mechanism such as page loading, handling page fault
4	I/O	i) I/O initialization ii) I/O completion iii) Error recovery
5	Communication	i) Inter process communication mechanism ii) Networking mechanism

- **Microkernel Based O S**

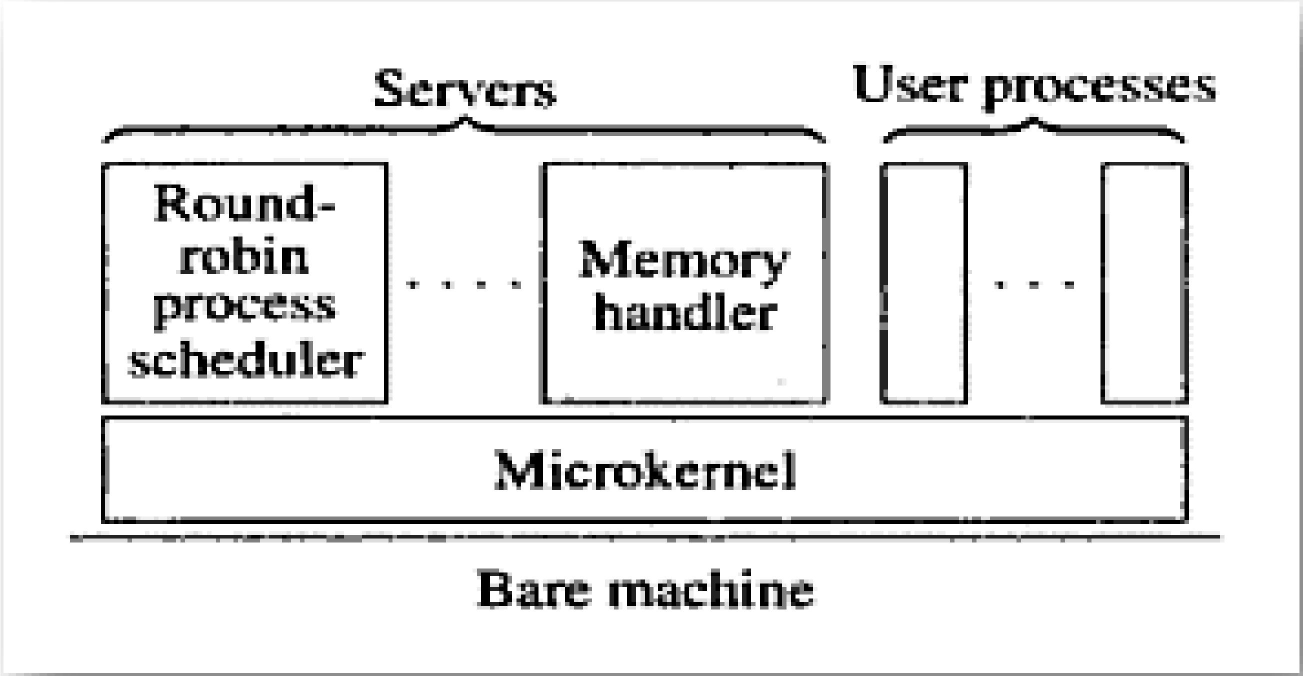


Figure: Structure of Microkernel Based O S

Continued…

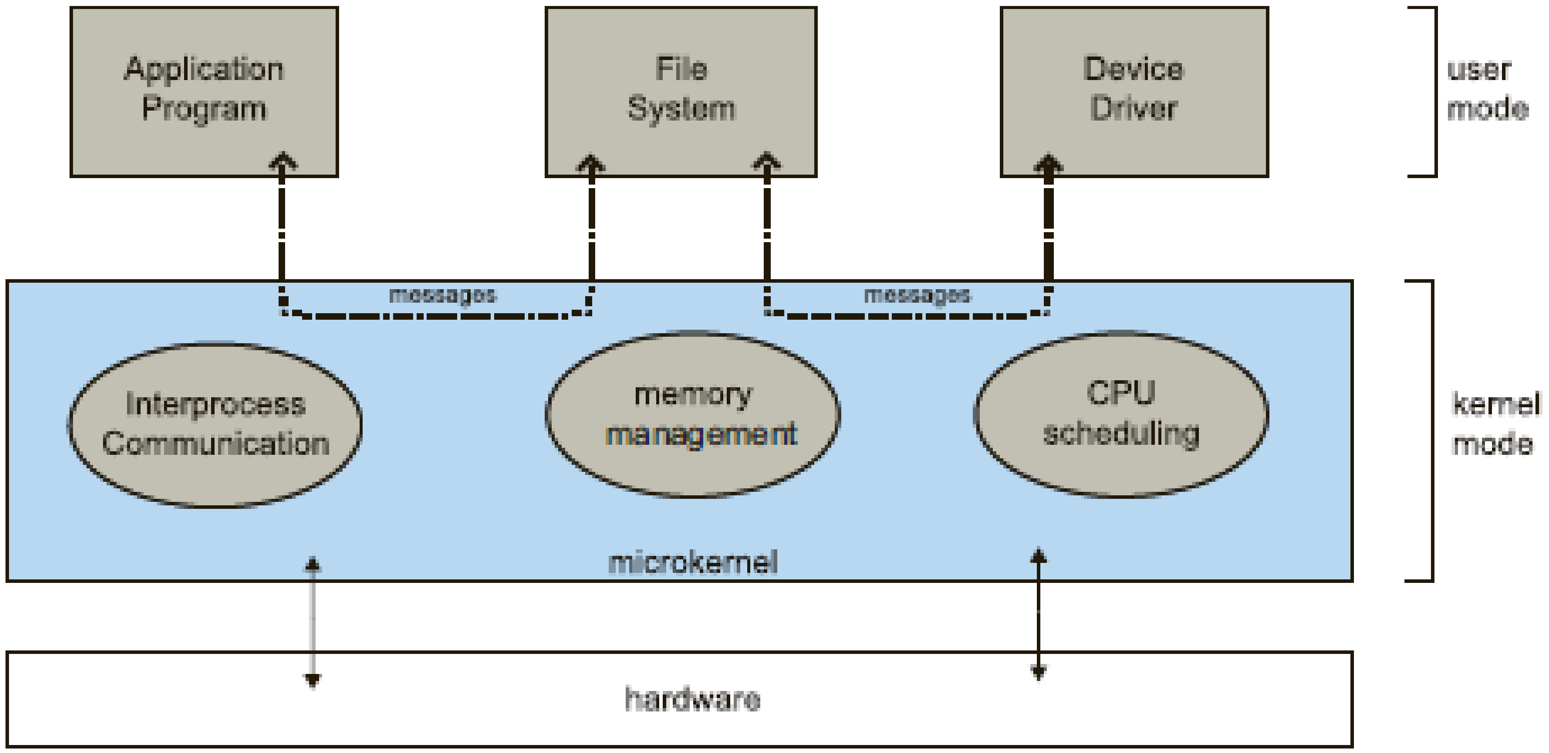


Figure: Architecture of typical Microkernel

Continued…

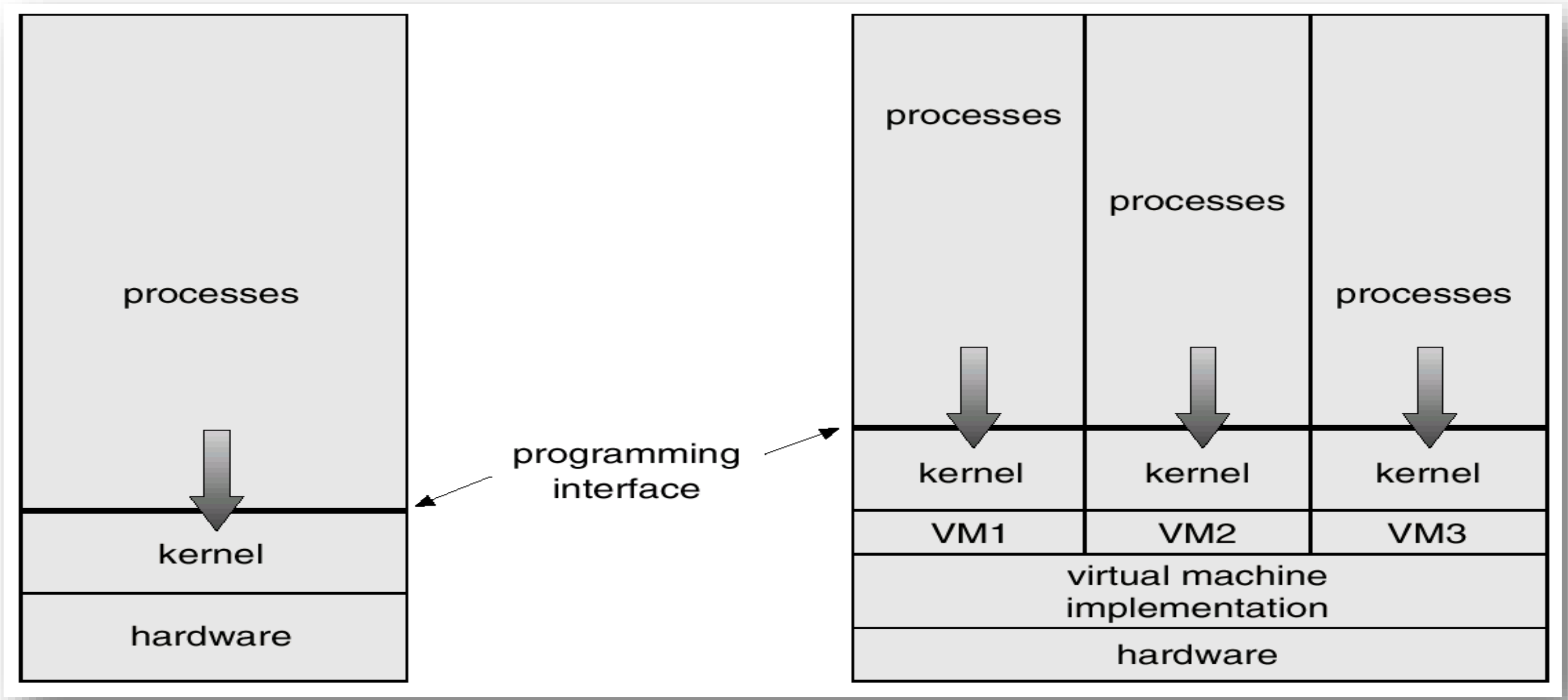
- Developed to overcome problems of kernels concerning portability, extensibility & reliability.
- Microkernel is essential core of OS code.
- Contains subset of mechanisms & supports small no of system calls which are heavily tested.
- Less essential part of OS code are outside microkernel.
- New functionality & features can be added to user space & hence don’t require modification to kernel.
- Can be considered as Client – Server structure
- e.g. C-DAC PARAS, Windows NT

Virtual Machines

- The **layered approach is taken to its logical conclusion** in the concept of a *virtual machine*.
- The idea is to abstract H/W of a single computer (CPU, memory, disk drive etc) into **several different execution environments.**
- This creates the illusion that each separate execution environment is running its own private computer.
- It **treats hardware and the operating system kernel as they are all hardware**.
- e.g. Java Virtual Machine, VM-Ware
- The resources of the physical computer are shared to create the virtual machines:
  - CPU scheduling can create the appearance that users have their own processor.

Continued…

System Models

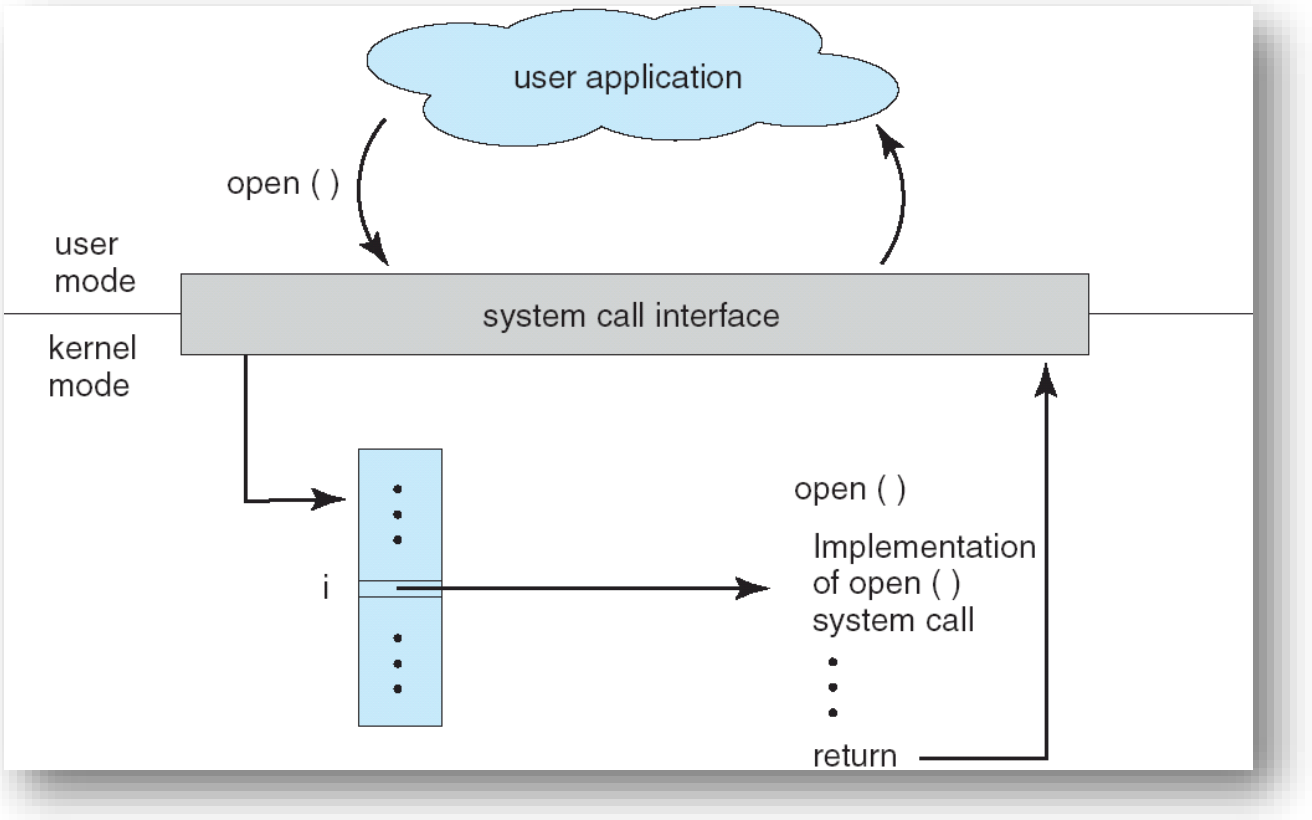


Figure(a): Non virtual Machine

Figure: Virtual Machine

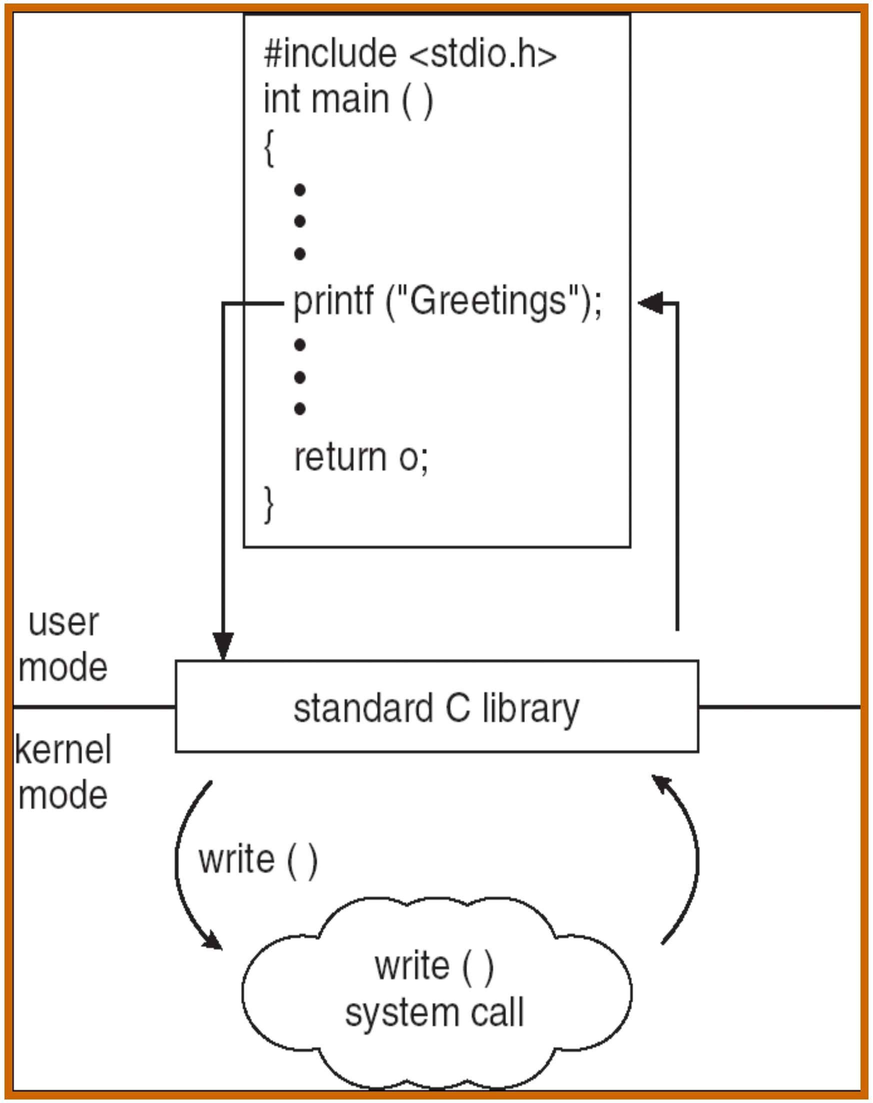
- **System calls**

- Systems calls are the **programming interface** to the services provided by the OS.
- It provides interface between process and OS.



Continued…

- C program invoking printf( ) library call, which calls the write( ) system call



```
#include <stdio.h>
int main ( )
{
  •
  •
  •
  printf ("Greetings");
  •
  •
  •
  return o;
}
```

Continued…

Types of System Call

1) Process Management

-end, abort

-load, execute

2) File Management

- Create File, Delete File
- open , close
- get file attribute, set file Attribute

Continued…

3) **Device Management**
   - -Request Device, Release Device
   - -Read, Write, Reposition
   - -get device attribute, set device attribute

4) **Information maintenance**
   - -get time/date, set time/date
   - -get system data, set system data
   - - get process, file or device attribute

5) **Communication**
   - -create, delete communication connection
   - -send, receive message
   - -transfer status information
   - -Attach, Detach remote device

Continued…

- **OS Services for process management/System calls for process management:**
  - CREATE(Pid, attr)
  - DELETE(Pid)
  - ABORT(Pid)
  - SUSPND(Pid)
  - DELAY(Pid, time)
  - GETATTRIBUTE(Pid, attr)
  - CHANGEPRIORITY(Pid, New priority)
  - DISPATCH A PROCESS(Pid)
  - TIMEUP A PROCESS(Pid)
  - WAKEUP A PROCESS(Pid)

References

Text Books:

1. **William Stallings**, O**perating System: Internals and Design Principles**, 8th Edition, Prentice Hall, 2014.

2. Abraham Silberschatz, Peter Baer **Galvin** and Greg Gagne, **Operating System Concepts**, 9th Edition, John Wiley & Sons, Inc., 2016.

3. Andrew **Tannenbaum**, **Operating System Design and Implementation**, 3rd Edition, Pearson, 2015.

Reference Books:

1. Maurice J. Bach, Design of UNIX Operating System, 2nd Edition, PHI, 2004.

2. Achyut Godbole and Atul Kahate, Operating Systems, 3rd Edition, McGraw Hill Education, 2017.

3. The Linux Kernel Book, Remy Card, Eric Dumas, Frank Mevel, 1st Edition, Wiley Publications, 2013.