RSET

RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY

ITT303
OPERATING SYSTEM CONCEPTS

Module I

Course Out line

- No of Hours: 4(3+1)
- (1) Introduction to OS,
- (2) Process Management,
- (3) Process Synchronization,
- (4) Memory Management
- (5) Storage Management

Text Books

1. Andrew S. Tanenbaum, “Modern Operating Systems”, Prentice Hall
2. 2. J. L. Peterson and A. Silberschatz , Operating System Concepts, Addison Wesley.

Course Outcome

- Explain the concepts and functionality of operating systems.
- Describe the concepts of process management and process synchronization and apply them to solve problems.
- Illustrate deadlock and deadlock – prevention and avoidance techniques.
- Illustrate the memory management techniques.
- Explain the file system and its implementation
- Use the disk scheduling algorithms to solve problems.

Syllabus

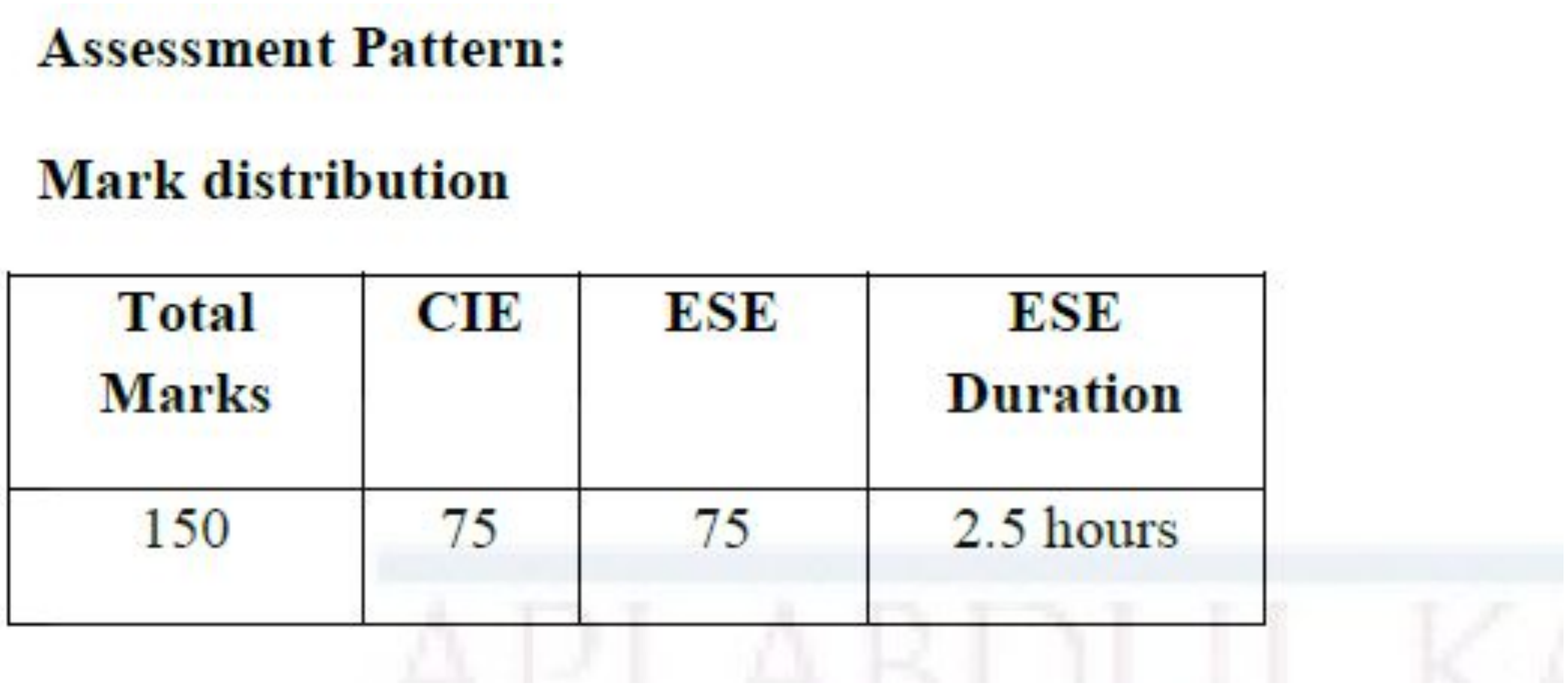
1. Introduction: Operating Systems-different types, System kernel, Shell,
2. Processes- . Process Scheduling methods, Inter process Communication,
3. Memory management : fixed &variable partitions - paging & segmentation - virtual memory concepts - demand paging - page replacement
4. Device management : disk scheduling algorithms - sector queuing -device drivers.
5. Dead locks - conditions for deadlock - prevention - avoidance - detection – recovery from dead lock -bankers' algorithm. - resource trajectories –starvation,
6. File system concepts – Access methods – Directory structure – Directory implementation – Linear list, Hash table

- OPERATING SYSTEM AND NETWORK PROGRAMMING LAB

Course Code : ITL 331

Course Outcome

- Analyse CPU Scheduling Algorithms like FCFS, Round Robin, SJF and Priority.
- Implement inter process communication and process synchronization problems.
- Implement memory management schemes - first fit, best fit and worst fit.
- Implement client server communication using sockets.
- Implement MAC protocols.
- Familiarization of network simulation tool.

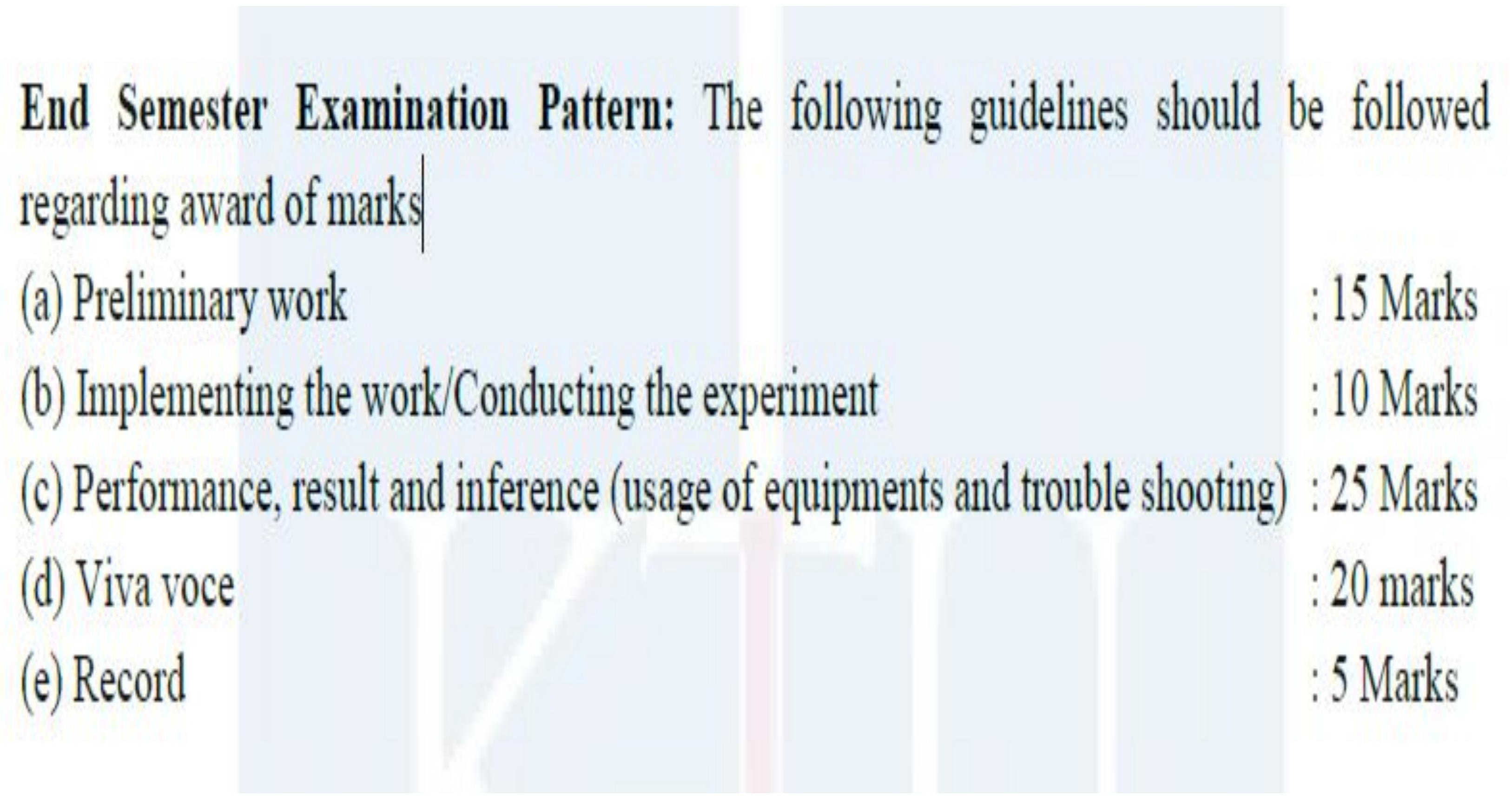


Assessment Pattern:

Mark distribution

Total Marks	CIE	ESE	ESE Duration
150	75	75	2.5 hours

- **Continuous Internal Evaluation Pattern:**
- Attendance : 15 marks
- Continuous Assessment : 30 marks
- Internal Test (Immediately before the second series test) : 30 marks



End Semester Examination Pattern: The following guidelines should be followed regarding award of marks

(a) Preliminary work	: 15 Marks
(b) Implementing the work/Conducting the experiment	: 10 Marks
(c) Performance, result and inference (usage of equipments and trouble shooting)	: 25 Marks
(d) Viva voce	: 20 marks
(e) Record	: 5 Marks

Module-I Contents

- Operating Systems: Introduction, Functions of OS, Types of OS (Batch, Multi programmed, Time-sharing and Real time systems) –System calls – System Programs –– System structure (Simple structure, Layered approach, Microkernel system structure, Modules)– Kernel, Shell.

Introduction

What Is An Operating System

A modern computer consists of:

- One or more processors
- Main memory
- Disks
- Printers
- Various input/output devices

Managing all these components requires a layer of software – the **operating system**

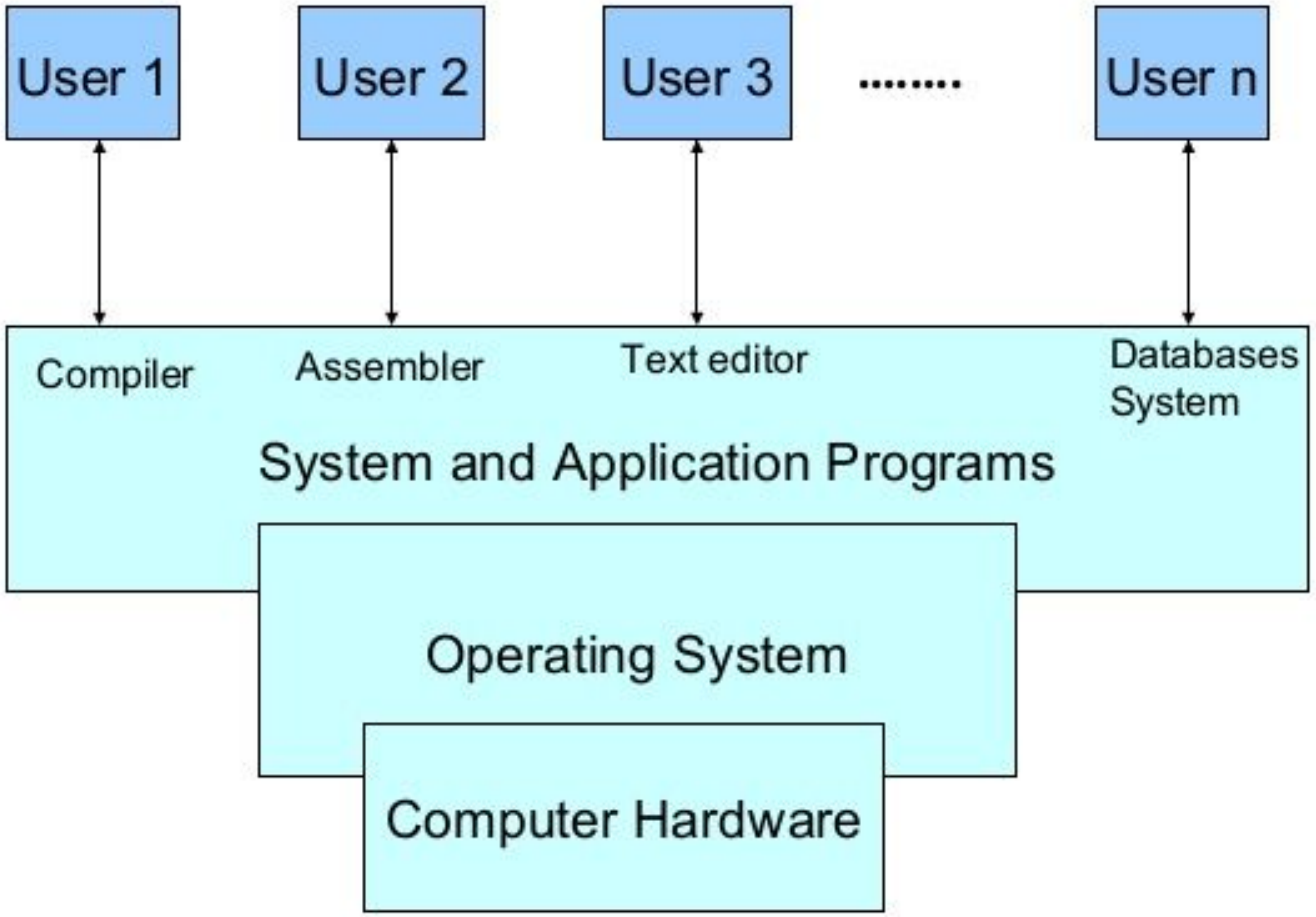


Figure: Abstract view of the components of a computer system

Computer System Structure

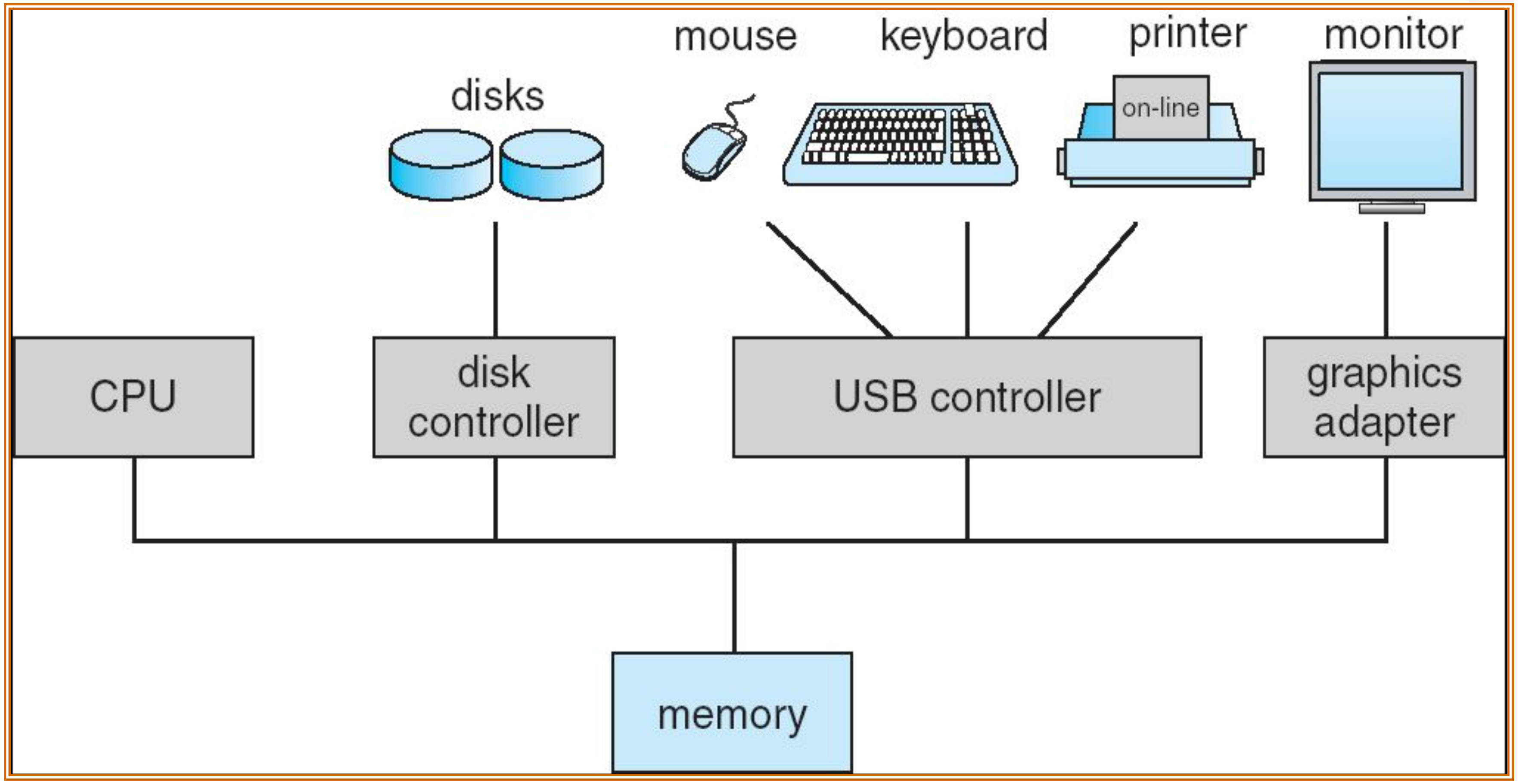
Computer system can be divided into four components

1. Hardware – provides basic computing resources
   - CPU, memory, I/O devices
2. Operating system
   - Controls and coordinates use of hardware among various applications and users
3. Application programs – define the ways in which the system resources are used to solve the computing problems of the users
   - Word processors, compilers, web browsers, database systems, video games
4. Users
   - People, machines, other computers

What is an Operating System?

- A collection of programs that acts as an interface between a user of a computer and the computer hardware.
- Provides an environment within which other programs can do useful work.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
  - Use the computer hardware in an efficient manner.
  - Security and protection to the user activities (prevent illegal accesses)

Computer System Organization



- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles

Operating System Definition

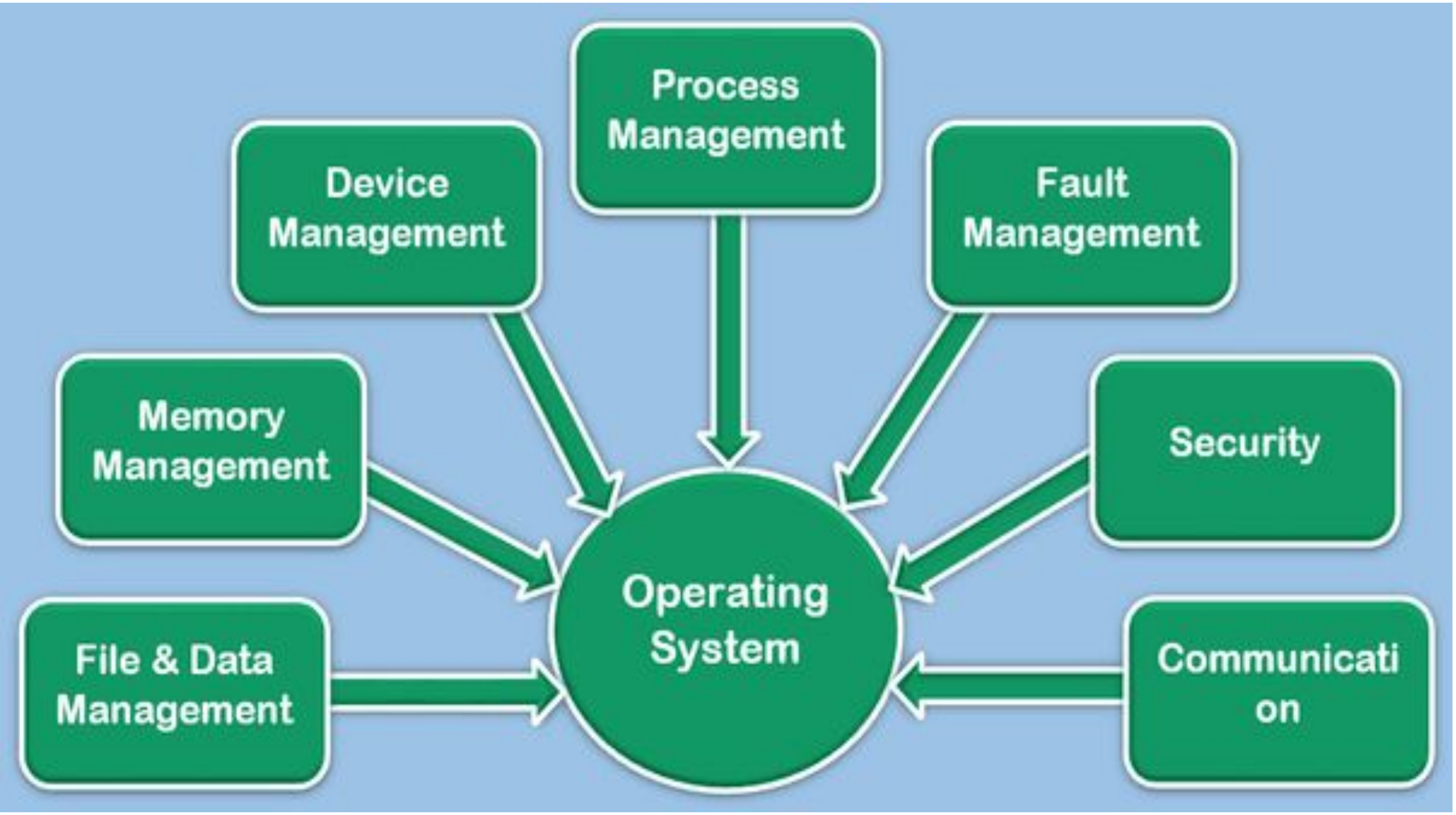
- OS is a **resource allocator**
  - Manages all resources (hardware and software)
  - Decides between conflicting resource requests for efficient and fair resource use.
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer.

OS as a Resource Manager

- Allow multiple programs to run at the same time
- Manage and protect memory, I/O devices, and other resources
- Includes multiplexing (sharing) resources in two different ways:
  - In time
  - In space

OS Functionalities/roles

- OS is responsible for all the functions of hardware as well as software



Function of OS

OS provides the environment within which programs are executed.

Function of OS:

- Process Management
- Main Memory Management
- Secondary-Storage Management
- I/O System Management
- File Management
- Command-Interpreter System

Protection System

Networking

Process Management

- A process is a program in execution. It is a unit of work within the system. A program is a passive entity, while a process is an active entity.
- A process needs resources to accomplish its task
- CPU, memory, I/O, files
- Initialization data
- Process termination requires the reclaiming of any reusable resources
- Typically a system has many processes, some user, some operating system processes, all running concurrently on one or more CPUs
- Concurrency is achieved by multiplexing the CPUs among the processes / threads

Process Management Activities

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

Memory Management

- Main memory is a repository of quickly accessible data shared by the CPU and I/O devices
- The CPU reads instructions from main memory during the instruction fetch-execute cycle and both reads and writes data from main memory
- For a program to be executed, it must be mapped to absolute addresses and loaded into memory
- The operating system is responsible for the following memory management activities:
- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts thereof) and data to move into and out of memory
- Allocating and deallocating memory space as needed

Storage Management

- The operating system provides a uniform, logical view of information storage
- It abstracts from the physical properties of a storage device to a logical storage unit called a file
- The operating system maps files onto physical media and accesses these files by way of the storage devices
- File Management
- Visible component of an OS
- Storage of information possible in magnetic disc, magnetic tapes etc
- Each storage medium has varying properties that include access speed, capacity, data-transfer rate, an access method (sequential or random

- File
- collection of related information by its creator,
- Files can be of any type
- Files are usually organized into directories
- Access control on most systems determines who can access what

- The OS is responsible for the following file management activities:
- Creating and deleting files and directories
- Supporting operations manipulate files and directories
- Mapping files onto secondary storage
- Backing up files onto stable (non-volatile) storage media

Mass Storage Management

- Main memory too small to accommodate all data and programs
- Also data that it holds are lost when power is lost
- Computer must provide secondary storage to back up main memory.
- Most computers use disks as the on-line storage medium for both programs and data.
- Most programs like compilers, word processors etc are stored in disk until loaded into memory.
- So management of disk storage is important.

Mass Storage Management

- The entire speed of computer operation hinges on disk subsystem and its algorithms
- The OS is responsible for the following disk management activities
- Free-space management
- Storage allocation
- Disk scheduling

- Some storage need not be fast-Tertiary Storage
- There are storage that is slower and lower in cost, but sometimes of higher capacity than secondary storage.
- Eg: Backups of disk data, seldom-used data, long term archival storage.
- Tertiary storage is not crucial to system performance, but still must be managed.
- like mounting and unmounting media in devices, migrating data from secondary to tertiary storage etc.

I/O Subsystem

- One of the purposes of an operating system is to hide peculiarities of hardware devices from the user
- In UNIX, the peculiarities of I/O devices are hidden from the bulk of the operating system itself by the I/O subsystem
- 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
- Only the device driver knows the peculiarities of the specific device to which it is assigned

Protection and Security

- If a computer system has multiple users and allows for concurrent execution of multiple processes, access to data must be regulated.
- Mechanisms ensure that files, memory segments, CPU and other resources can be operated only by one of the processes that have gained proper authorization from the OS
- Eg:
- A process can execute only within its own address space
- A timer ensures that no process can gain control of CPU without eventually relinquishing control.
- Device control registers are not accessible to users

- Protection – any mechanism for controlling access of processes or users to the resources defined by the operating system
- This must provide means to specify the controls to be imposed and to enforce the controls.
- It can improve reliability by detecting latent errors at the interfaces.

Protection and Security

- Security – defends a system from internal and external attacks
- This is a huge range of threats, including denial-of-service, viruses, identity theft, and theft of service
- Systems generally first distinguish among users, to determine who can do what
- User identities (user IDs) include name and associated number, one per user
- The user ID then is associated with all files and processes of that user to determine access control
- A group identifier (group ID) allows a set of users to be defined an associated with each process, directory and file
- Privilege escalation allows user to change to an effective ID with more rights for a short period of time

Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - process suspension and resumption.
  - Provision of mechanisms for:
    - process synchronization
    - process communication

Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

Secondary-Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

I/O System Management

- The I/O system consists of:
  – A buffer-caching system
  – A general device-driver interface
  – Drivers for specific hardware devices
- I/O system management activities are:
  – I/O interrupt servicing,
  – initiation of I/O operations,
  – optimization of I/O device performance

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

Protection System

- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.

Networking (Distributed Systems)

- A *distributed* system is a collection processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  – Computation speed-up
  – Increased data availability
  – Enhanced reliability

Command-Interpreter System

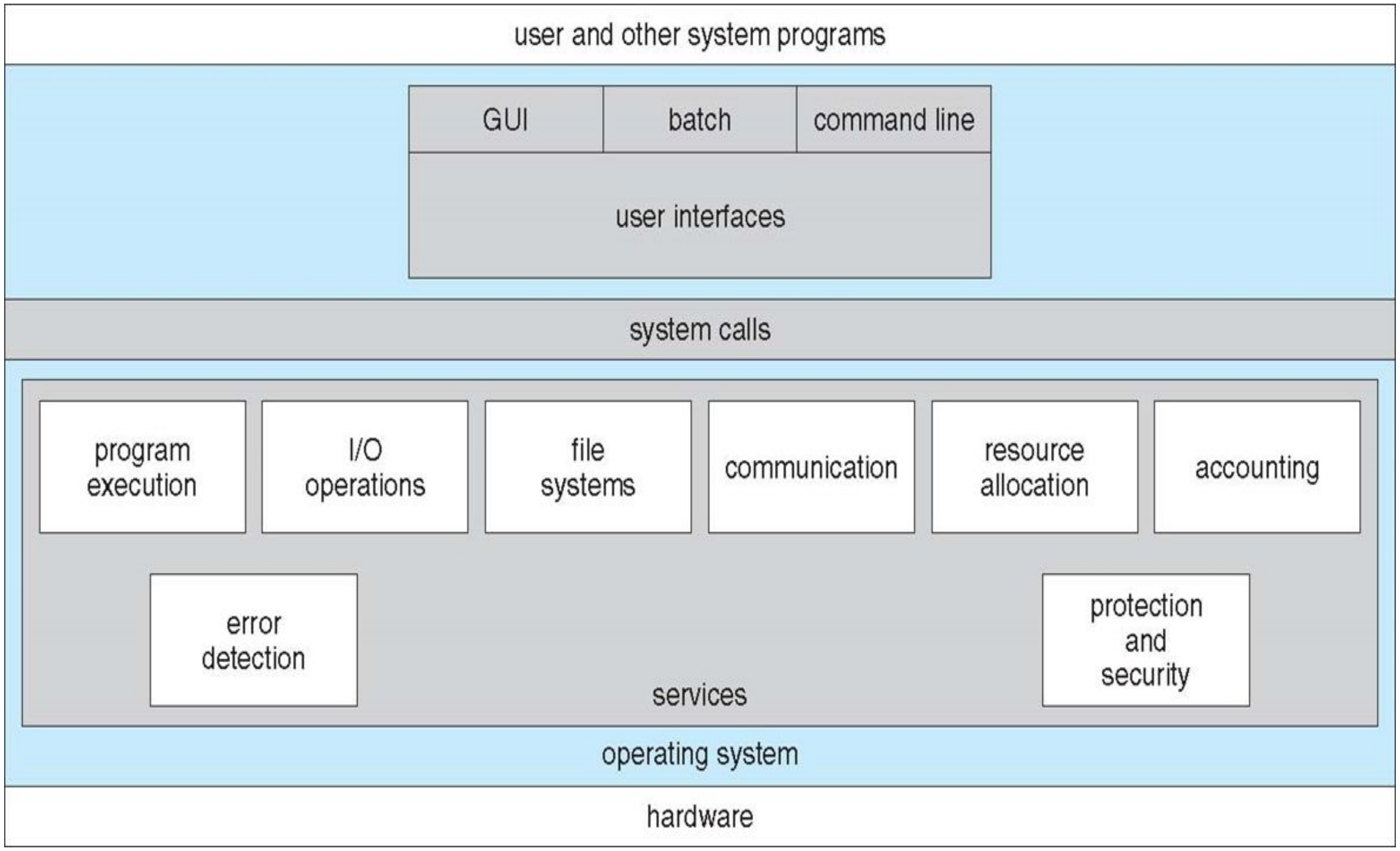
- Many commands are given to the operating system by control statements which deal with:
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection
  - networking

Additional Operating System Functions

Additional functions exist not for helping the user, but rather for ensuring efficient system operations.

- Resource allocation – allocating resources to multiple users or multiple jobs running at the same time.
- Accounting – keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- Protection – ensuring that all access to system resources is controlled.

OS Services


user and other system programs
GUI
batch
command line
user interfaces
system calls
program execution
I/O operations
file systems
communication
resource allocation
accounting
error detection
protection and security
services
operating system
hardware

OS Services

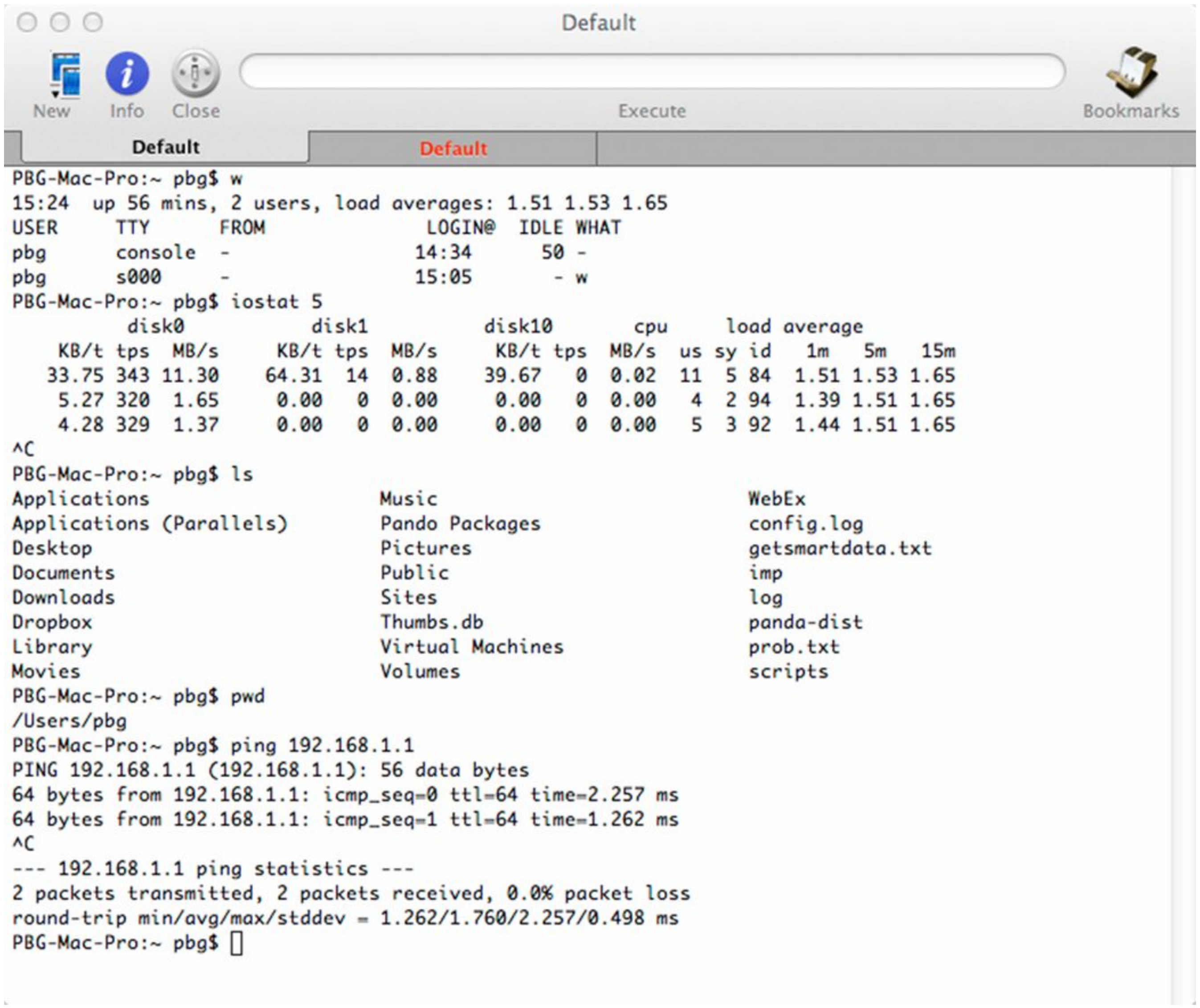
- User Interface
- Program Execution
- I/O operation
- File system manipulation
- Communication between processes
- Error Detection
- Resource allocation
- Accounting: keeping usage statistics
- Protection and security: control resource access

User Operating System Interface - CLI

CLI or **command interpreter** allows direct command entry

- Sometimes implemented in kernel, sometimes by systems program
- Sometimes multiple flavors implemented – **shells**
- Primarily fetches a command from user and executes it
- Sometimes commands built-in, sometimes just names of programs
  - If the latter, adding new features doesn’t require shell modification

Bourne Shell Command Interpreter



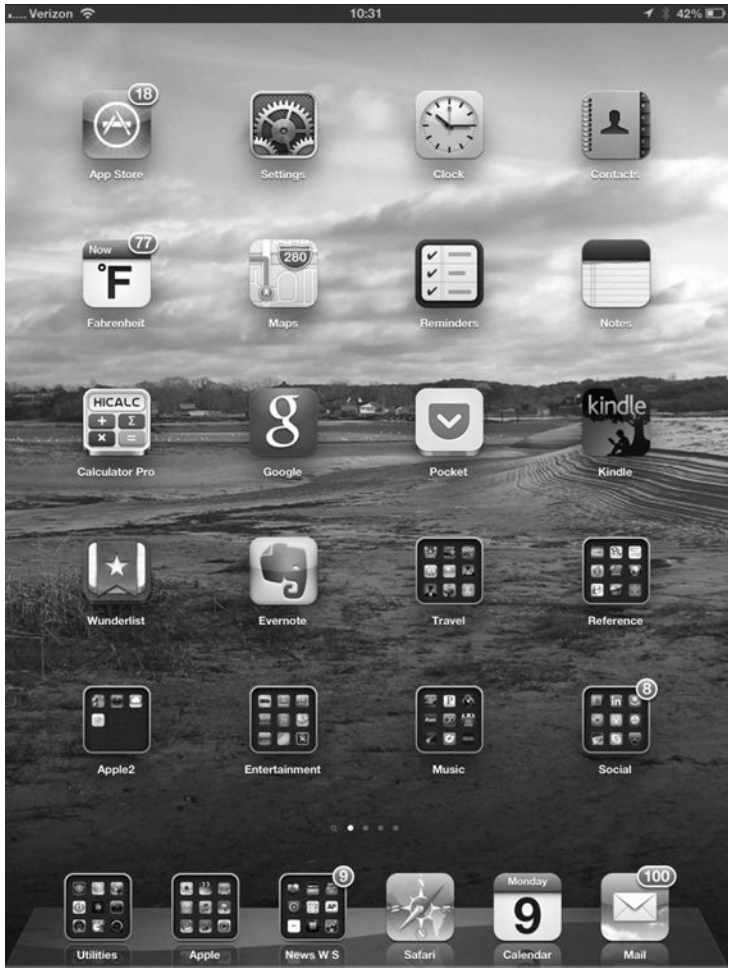
```
PBG-Mac-Pro:~ pbg$ w
15:24  up 56 mins, 2 users, load averages: 1.51 1.53 1.65
USER     TTY      FROM              LOGIN@  IDLE WHAT
pbg      console  -                14:34      50 -
pbg      s000     -                15:05       - w
PBG-Mac-Pro:~ pbg$ iostat 5
          disk0             disk1           disk10         cpu      load average
    KB/t tps  MB/s     KB/t tps  MB/s     KB/t tps  MB/s  us sy id   1m    5m   15m
   33.75 343 11.30    64.31  14  0.88    39.67   0  0.02  11  5 84  1.51 1.53 1.65
    5.27 320  1.65     0.00   0  0.00     0.00   0  0.00   4  2 94  1.39 1.51 1.65
    4.28 329  1.37     0.00   0  0.00     0.00   0  0.00   5  3 92  1.44 1.51 1.65
^C
PBG-Mac-Pro:~ pbg$ ls
Applications                    Music                           WebEx
Applications (Parallels)        Pando Packages                  config.log
Desktop                         Pictures                        getsmartdata.txt
Documents                       Public                          imp
Downloads                       Sites                           log
Dropbox                         Thumbs.db                       panda-dist
Library                         Virtual Machines                prob.txt
Movies                          Volumes                         scripts
PBG-Mac-Pro:~ pbg$ pwd
/Users/pbg
PBG-Mac-Pro:~ pbg$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=2.257 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.262 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.262/1.760/2.257/0.498 ms
PBG-Mac-Pro:~ pbg$ 
```

User Operating System Interface - GUI

- User-friendly desktop metaphor interface
- Usually mouse, keyboard, and monitor
- Icons represent files, programs, actions, etc
- Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)
- Invented at Xerox PARC
- Many systems now include both CLI and GUI interfaces
- Microsoft Windows is GUI with CLI “command” shell
- Apple Mac OS X is “Aqua” GUI interface with UNIX kernel underneath and shells available
- Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)

Touchscreen Interfaces

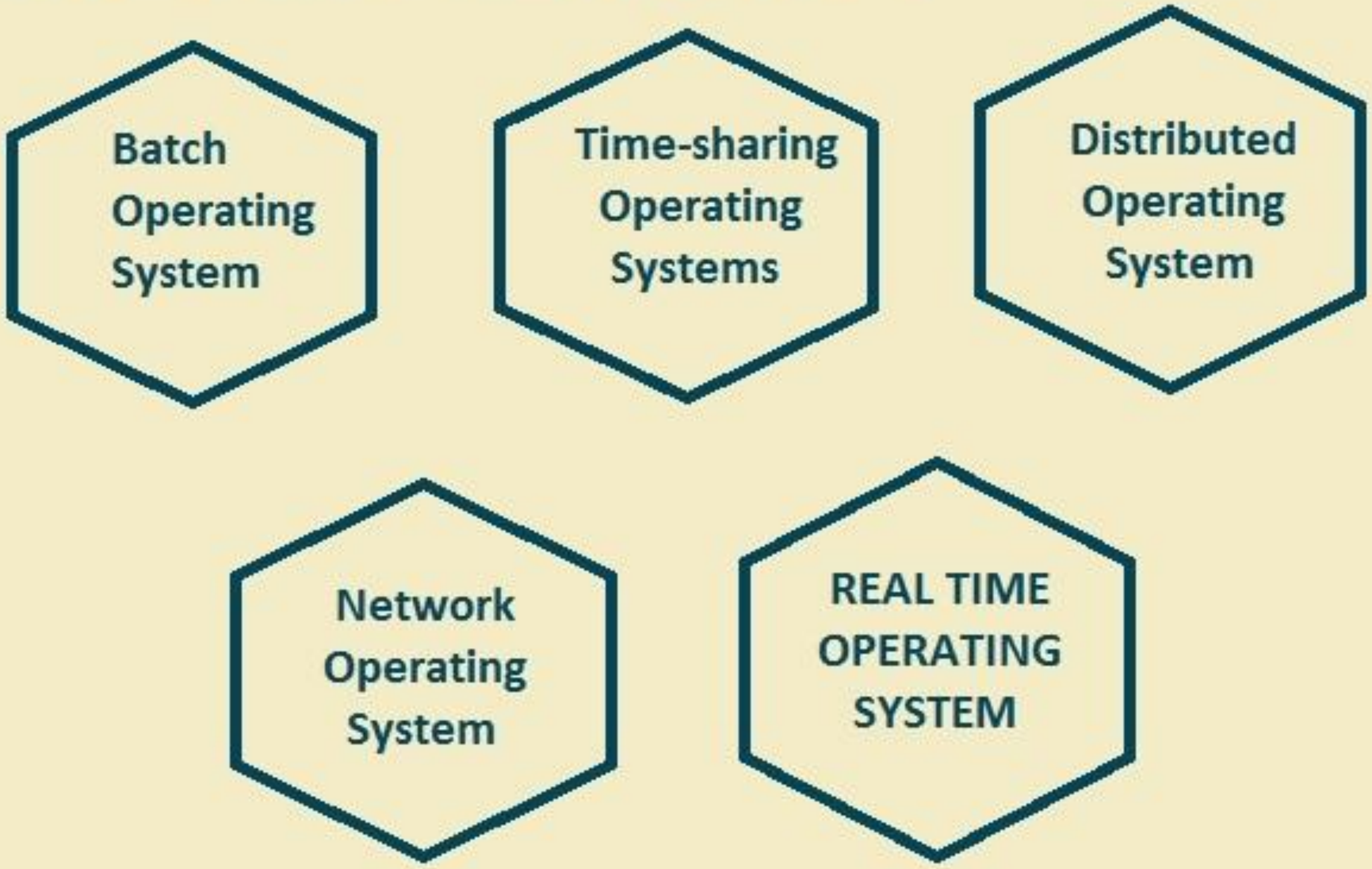
- Touchscreen devices require new interfaces
- Mouse not possible or not desired
- Actions and selection based on gestures
- Virtual keyboard for text entry
- Voice commands.



Operating System Services

- Program execution – system capability to load a program into memory and to run it.
- I/O operations – since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- File-system manipulation – program capability to read, write, create, and delete files.
- Communications – exchange of information between processes executing either on same computer or on different systems tied together by a network. Implemented via *shared memory* or *message passing*.
- Error detection – ensure correct computing by detecting errors in CPU and memory hardware, in I/O devices, or in user programs.

TYPES OF OPERATING SYSTEM


Batch Operating System
Time-sharing Operating Systems
Distributed Operating System
Network Operating System
REAL TIME OPERATING SYSTEM

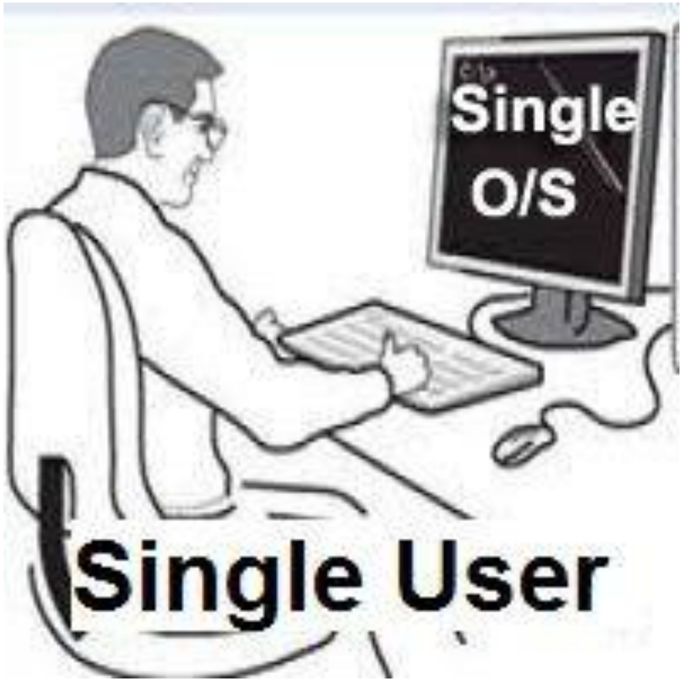
Types of Operating Systems

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

- single user/single task system
- Simple Batch System
- Multiprogramming Batch System
- Real time Operating System
- Time sheard OS

Single-user, single task

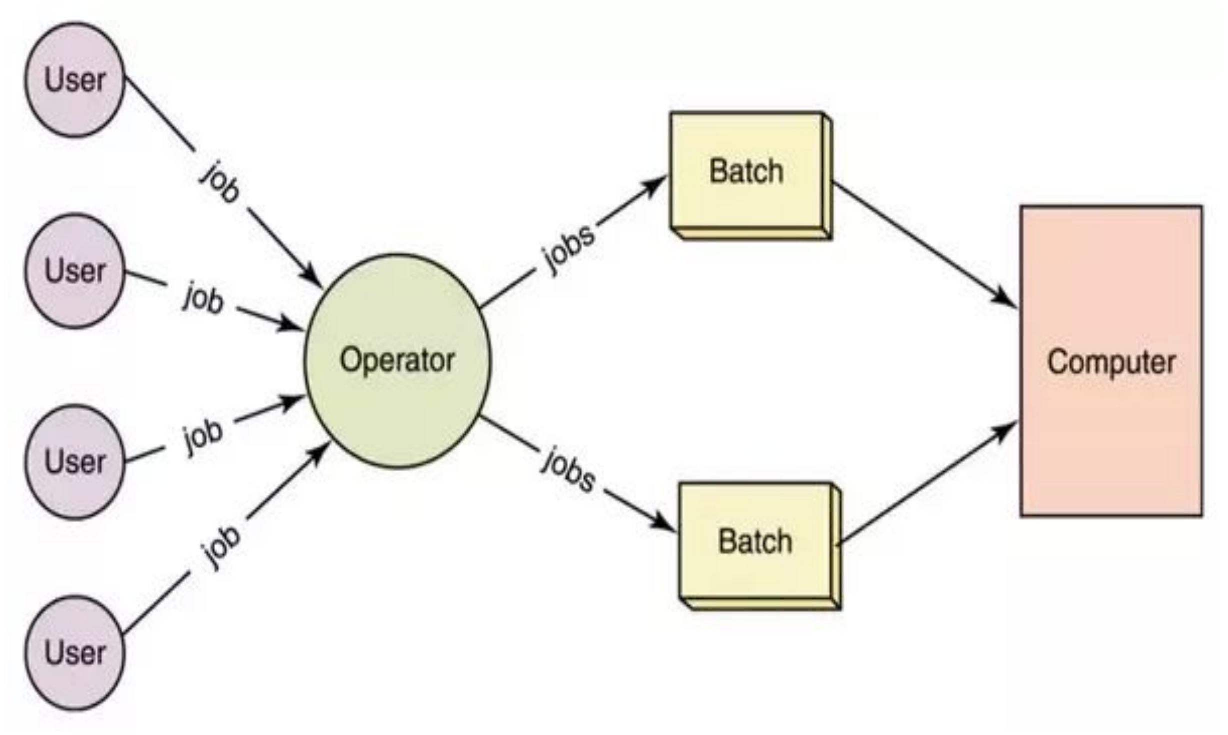
- As the name implies, this operating system is designed to manage the computer so that one user can effectively do one thing at a time. The Palm OS for Palm handheld computers is a good example of a modern single-user, single-task operating system.

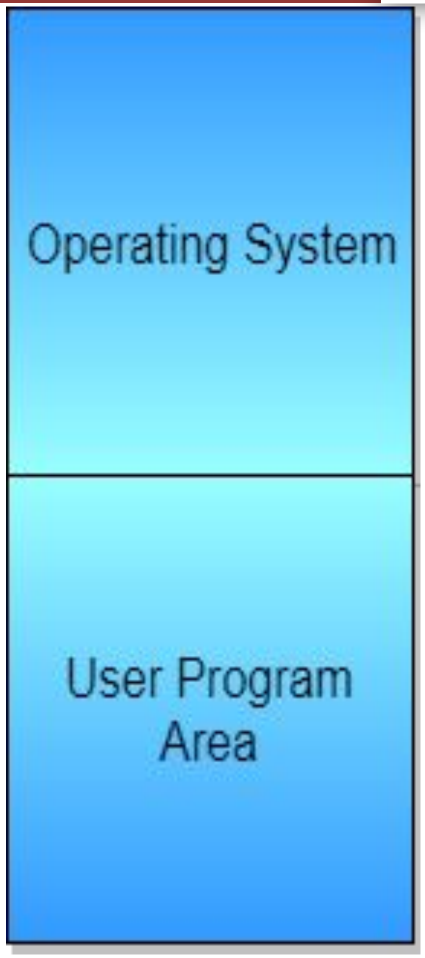


Simple Batch Systems

- In this type of system, there is no direct interaction between user and the computer.
- The user has to submit a job (written on cards or tape) to a computer operator.
- Then computer operator sorts the programs with similar requirements into batches and places a batch of several jobs on an input device.
- Then a special program, called monitor, manages the execution of each program in the batch.
- The output from each job would be sent back to the appropriate programmer.

Simple Batch System Memory layout



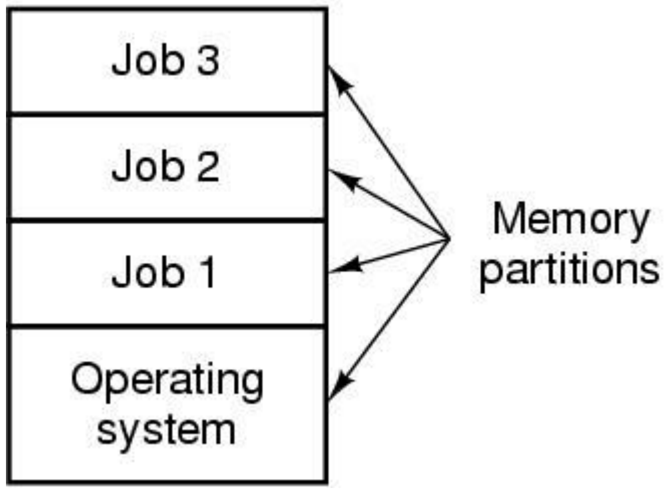


Disadvantages of Simple Batch OS

- No interaction between user and computer.
- CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.
- No mechanism to prioritize the processes.

Multiprogramming batch Systems

- Multiprogramming needed for efficiency
- Single program cannot keep CPU and I/O devices busy always.
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute.
- OS keeps several jobs in memory simultaneously.
- One job selected and run via job scheduling

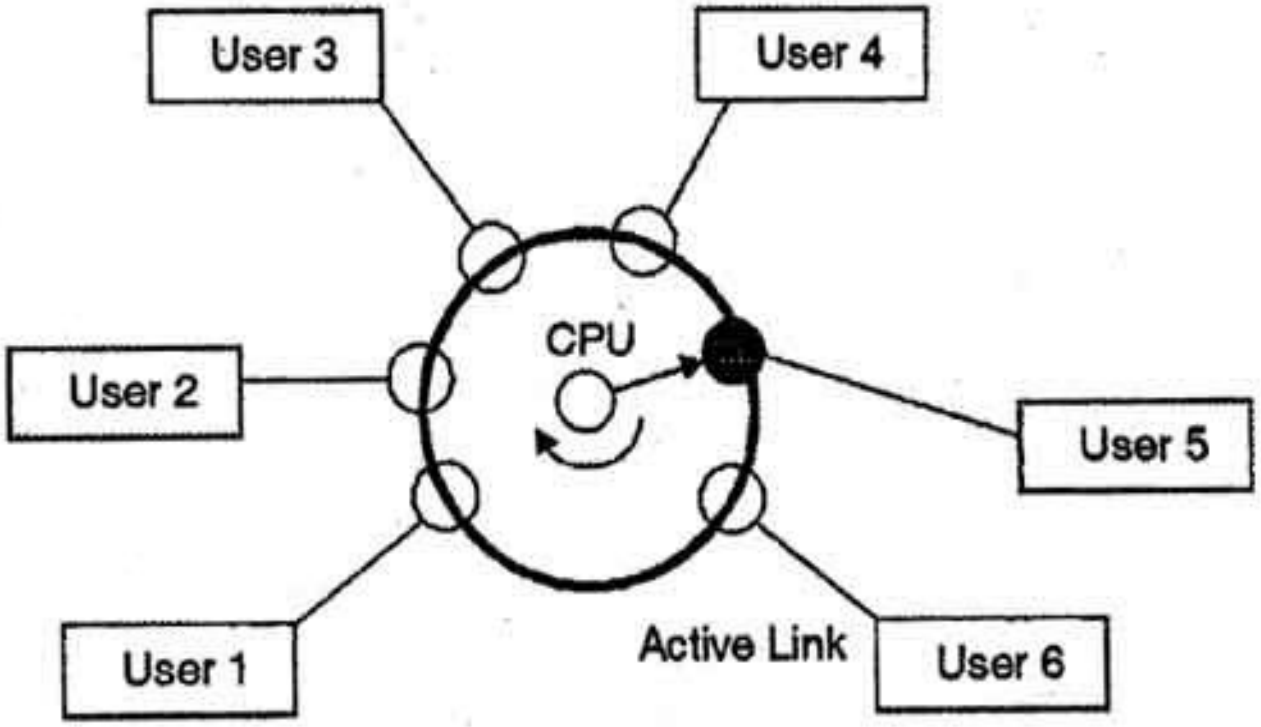


Multiprogramming Batch Systems

- In this the operating system picks up and begins to execute one of the jobs from memory.
- Once this job needs an I/O operation operating system switches to another job (CPU and OS always busy).
- Jobs in the memory are always less than the number of jobs on disk(Job Pool).
- If several jobs are ready to run at the same time, then the system chooses which one to run through the process of CPU Scheduling.
- In Non-multiprogrammed system, there are moments when CPU sits idle and does not do any work.
- In Multiprogramming system, CPU will never be idle and keeps on processing.

Time Sharing/Multitasking Systems

- Time Sharing Systems are very similar to Multiprogramming batch systems. In fact time sharing systems are an extension of multiprogramming systems.
- Also known as multitasking; is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
- In time sharing systems the prime focus is on minimizing the response time, while in multiprogramming the prime focus is to maximize the CPU usage.
- Quick response is achieved by giving fair execution opportunity to each process by two means:
  1. Round-robin scheduling : services all process by turn
  2. Time-slicing: prevents a process from using too much CPU time



Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time.

TIME SHARING SYSTEMS

Advantages

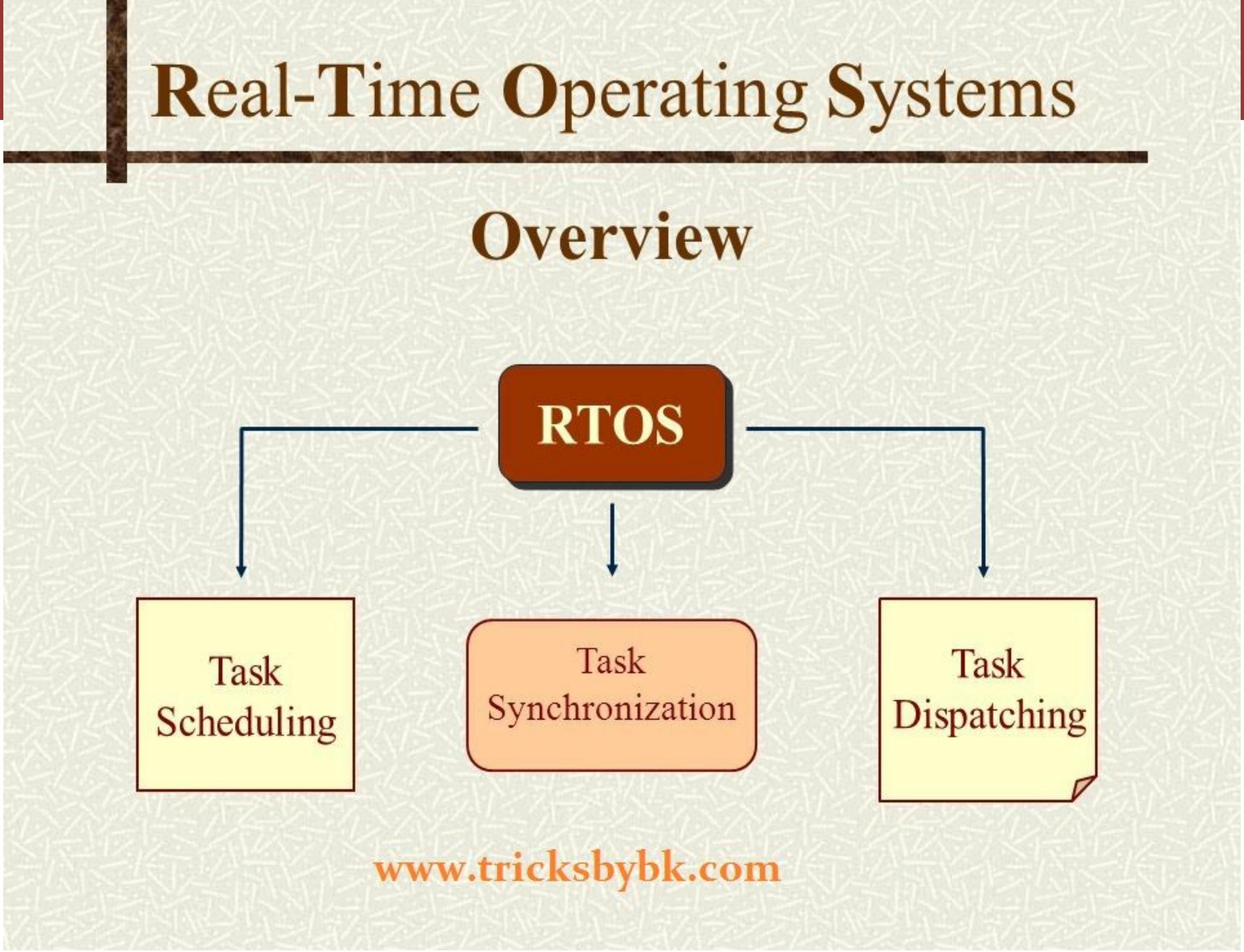
- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Real Time Operating System

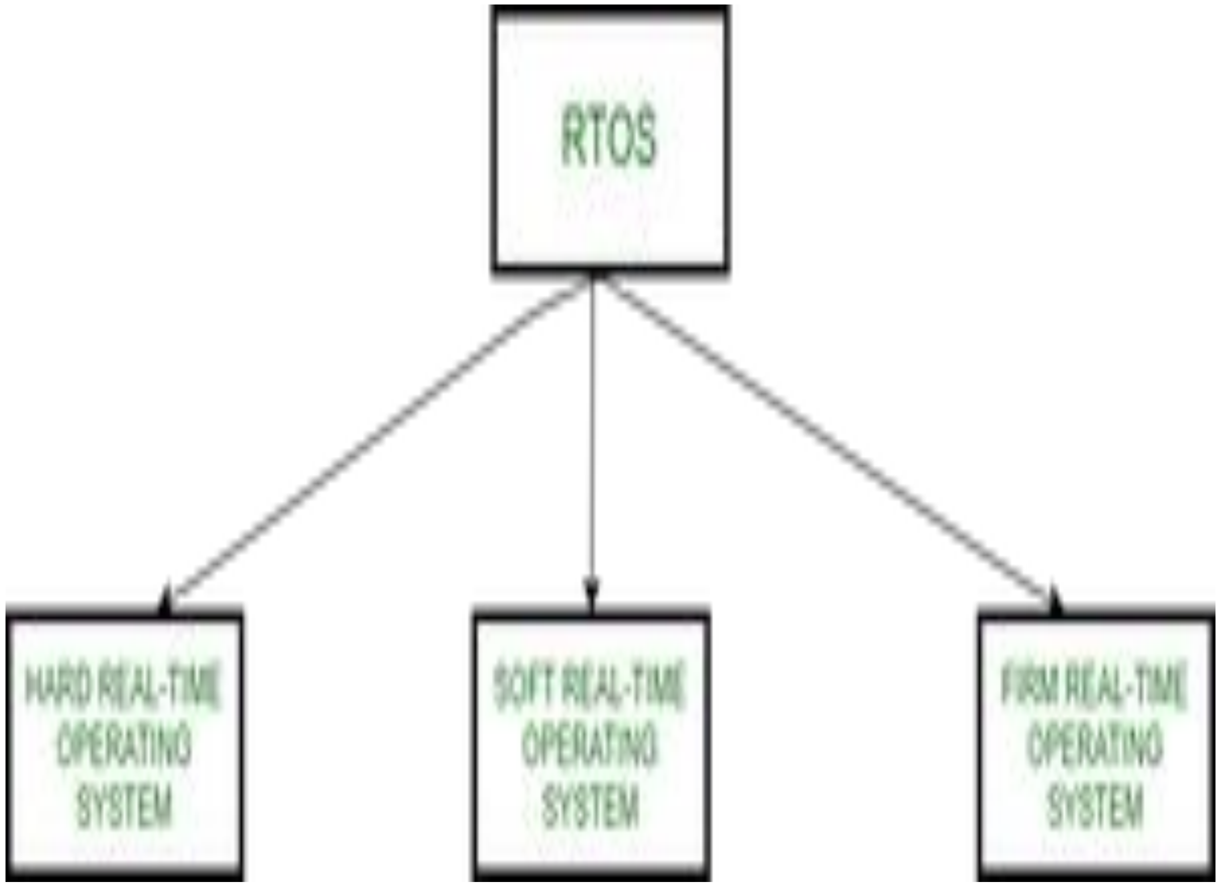
- Used to implement a computer application for controlling or tracking of real-world activities.
- Application needs to complete its computational tasks in a timely manner.
- It is defined as an operating system known to give maximum time for each of the critical operations that it performs, like OS calls and interrupt handling.
- A real-time operating system must have well-defined, fixed time constraints, otherwise the system will fail.

Examples of RTOS

- Embedded systems
- Industrial control systems
- Robots
- Air traffic control systems, etc.,



- **Real-time** systems are systems in which **correctness** of the system depends **not only** on the **logical result** of computation, **but also** on the **time** at which the results are produced.


RTOS
HARD REAL-TIME OPERATING SYSTEM
SOFT REAL-TIME OPERATING SYSTEM
FIRM REAL-TIME OPERATING SYSTEM

Two types of real-time OS

1. Hard Real-Time Operating Systems
2. Soft Real-Time Operating Systems.

- The RTOS which guarantees the maximum time for critical operations and complete them on time are referred to as Hard Real-Time Operating Systems.
- Delays are bounded
- Secon storage limited,or absent data stored in RAM
- A hard real time system is very restrictive and safety is critical.

Example

1. Air Traffic Control
2. Medical System: Telesurgery

Soft Real-Time Operating Systems.

- The RTOS that can only guarantee a maximum of the time, i.e. the critical task will get priority over other tasks, but no assurity of completing it in a defined time. These systems are referred to as Soft Real-Time Operating Systems.
- Soft real time system is a system whose operation is degrade if results are not produce according to the specified timing requirement.
- A Soft real time system is less restrictive and safety is not critical.

For example

- Computer games
- Virtual reality

System Call

- A system call is a way for a user program to interface with the operating system.
- The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request.
- A system call can be written in assembly language or a high-level language like C or Pascal.
- System calls are predefined functions that the operating system may directly invoke if a high-level language is used.

What is a System Call?

- A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running.
- A system call is a method of interacting with the operating system via programs.
- A system call is a request from computer software to an operating system's kernel.

- The Application Program Interface (API) connects the operating system's functions to user programs. It acts as a link between the operating system and a process, allowing user-level programs to request operating system services.
- The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.

Why do you need system calls in Operating System?

- It is must require when a file system wants to create or delete a file.
- Network connections require the system calls to sending and receiving data packets.
- If you want to read or write a file, you need to system calls.
- If you want to access hardware devices, including a printer, scanner, you need a system call.
- System calls are used to create and manage new processes.

System Call

- System call provide an interface to the OS services
- Computer work in two modes:
  - Kernel mode
  - User Mode (safer mode)
- If user need to access some resource, makes a call that switches from user mode to kernel mode (context switch) it is called system call.
- System call is a programmatic way in which a computer program requests a service from the kernel of OS.
- These call are generally available as routines written in C or C++
- Example: Copy one file content to another file.

System Calls: Examples

System Calls are routine services of OS.

Examples

- Create: OS creates a new process with the specified attributes
- Delete: destroy the designated process
- Abort: terminate the process forcibly
- Fork/join: split and merge sequence of instruction
- Suspend: suspended indefinitely
- Resume: resume the target process which is presumably suspended.
- Delay: suspended for a specified time period
- Level of privilege, priority, size, data area, access rights
- Get_attributes
- Change priority

System Call Categories

1. Process control:
2. File manipulation:
3. Device manipulation:
4. Information Maintenance:
5. Communications:

1. Process control System Call

Used in controlling the processes

- end, abort: Running program needs to halt execution normally or abnormally
- load, execute: Process executing one program may need to load and execute another program
- Create process, terminate process:
- Get process attributes, set process attributes (priority, execution time)
- wait event, signal event,
- allocate and free memory

2. File manipulation System call

- Create file, delete file: Requires file name and attributes (type, protection)
- open, close:
- read, write:
- get file attributes, set file attributes

  (Name, type, location, size....)

3. Device manipulation System call

To acquire additional resources needed for a running process. System call manages resource access

- Request device, release device,
- read, write,
- get device attributes, set device attributes,
- logically attach or detach devices (OS understand the device is attached/detached with the system)

4. Information Maintenance System call

Maintaining the information of the system

Transferring information between user program and OS

- get time or date, set time or date
- get system data, set system data
- get process, file, device attributes
- set process, file, device attributes

5. Communications System Call

Used to make Communication between different processes

Two communication models of communication

1. Message-passing model:
   - Information exchanged through interprocess communication
   - Useful for small data transfer.
   - Easy to implement
2. Shared-memory model: Two or more process exchange information by reading and writing data in shared memory area
   - create, delete communication connection
   - send, receive messages
   - transfer status information
   - attach or detach remote devices

System Call Implementation

- In most systems, system calls can only be made from userspace processes, while in some systems, OS/360 and successors for example, privileged system code also issues system calls.
- OS executes at the highest level of privilege, and allows applications to request services via system calls, which are often initiated via interrupts.
- An interrupt automatically puts the CPU into some elevated privilege level, and then passes control to the kernel, which determines whether the calling program should be granted the requested service.
- If service is granted, kernel executes a specific set of instructions to which calling program has no direct control
- Returns privilege level to that of the calling program, and then returns control to the calling program.

Typical System Call Implementations

- RISC Processor:
  - set up some register with the system call number needed, and execute the software interrupt (trap).
  - transfer control to the operating system kernel.
- CISC Processor:
  - instruction set contains the instructions SYSCALL/SYSRET and SYSENTER/SYSEXIT (AMD and Intel).
- "fast" control transfer instructions that are designed to quickly transfer control to the kernel for a system call without the overhead of an interrupt.
- Linux 2.5
- formerly used the INT instruction, where the system call number was placed in the EAX register before interrupt 0x80 was executed

Typical System Call Implementations

- Multics
  - An older mechanism is the call gate;
  - It allows a program to call a kernel function directly using a safe control transfer mechanism, which the OS sets up in advance
- IA-64 architecture
  - EPC (Enter Privileged Code) instruction is used.
  - The first eight system call arguments are passed in registers, and the rest are passed on the stack.
- IBM System/360 mainframe family
  - Supervisor Call instruction (SVC), with the number in the instruction rather than in a register, implements a system call for most of] IBM's OS, and for all system calls in Linux.
  - In IBM's OS, the Program Call (PC) instruction is used for newer facilities. In particular, PC is used when the caller might be in Service Request Block (SRB) mode

System Programs

- System programs provide a convenient environment for program development and execution.
- For example, a compiler is a complex system program.

Categories:

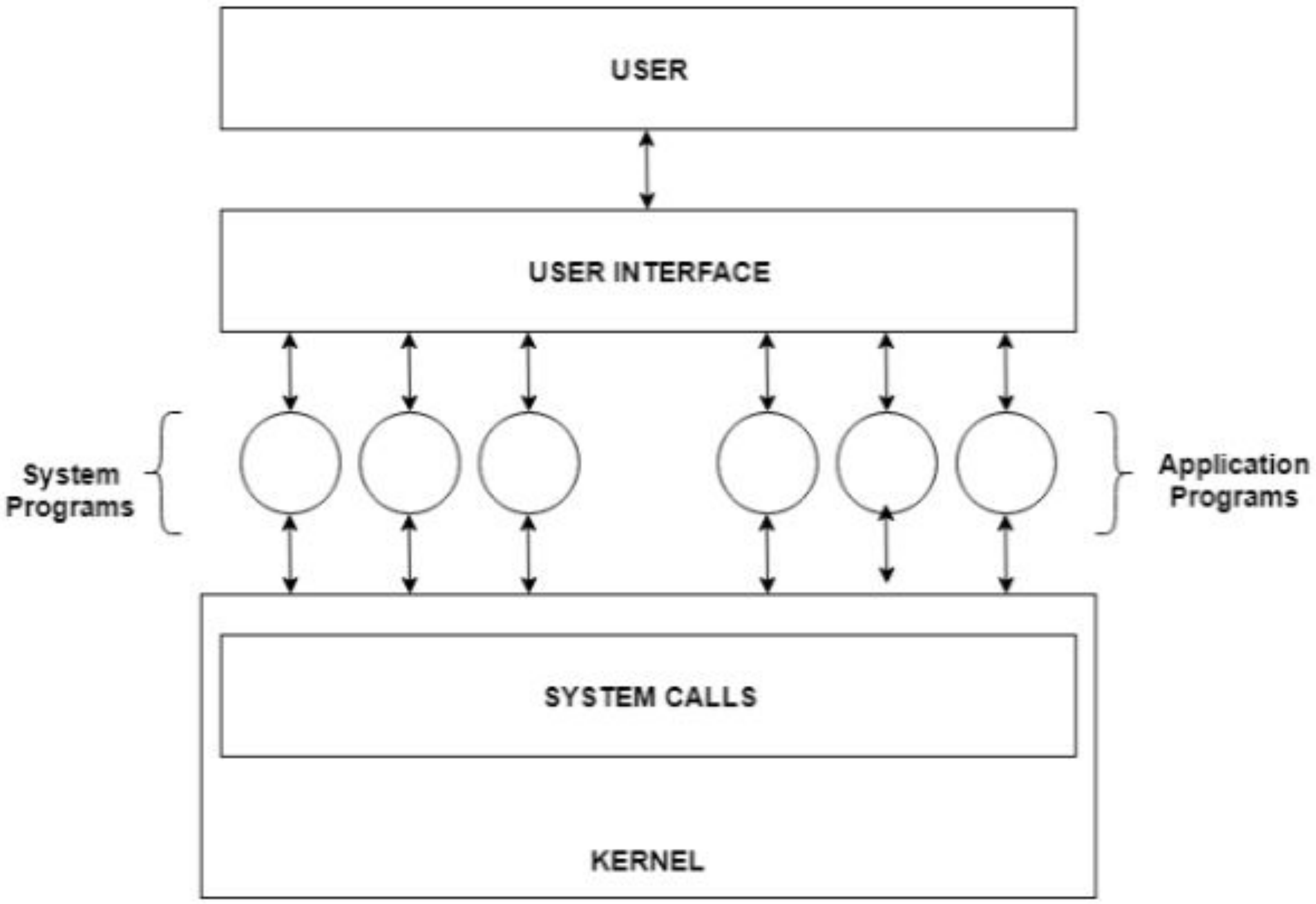
- File management: create, delete, copy, rename, print, dump, list
- Status information: no. of users, available memory or disk space
- File modification: modify content of file using text editors
- Programming language support: compilers, assemblers, interpreters for C, C++, Java etc.
- Program loading and execution: provide loaders and linkers
- Communications: provide virtual connection among processes, users and computers

- Most users' view of OS is defined by system programs, not the actual system calls.

System Calls

- Provide interface between a running program and OS.
  - Generally available as assembly-language instructions.
  - Some system allow system calls directly from HLL programs resemble predefined functions or subroutine calls
- Three general methods are used to pass parameters between a running program and OS.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the *stack* by the program, and *pop* off the stack by OS.

Operating system hierarchy



System Call Vs System Program

- User program receives OS services through system calls while system program provide a convenient environment for program development and execution.
- System calls, are the calls made by applications or the processors for a particular execution of a code block;
- System Program is the program that is run in the kernel space. System call is the API exposed from kernel space to user space.
- So, your program running in user space calls to system program via API which is called as system call.
- These are the programs that are used and required to run the system - machine, input output devices and other connected peripherals. They are also known as System softwares.

OS Structure

Structure of OS concerns the nature of OS core (Kernel) and other parts of OS and their interaction.

1. Monolithic structure:
   - OS formed a single software layer between user and the bare machine.
   - User interface was provided by command interpreter.
   - Both command interpreter and user processes invoked OS functionalities and services through system calls
   - Poor portability, high cost of maintenance and enhancement.
2. Layered structure: Structuring OS into no. of layers
3. Kernel-based structure:
   - Small portion of OS code constitutes Kernel
   - Portability increased
4. Microkernel-based OS structure
5. Modular structure

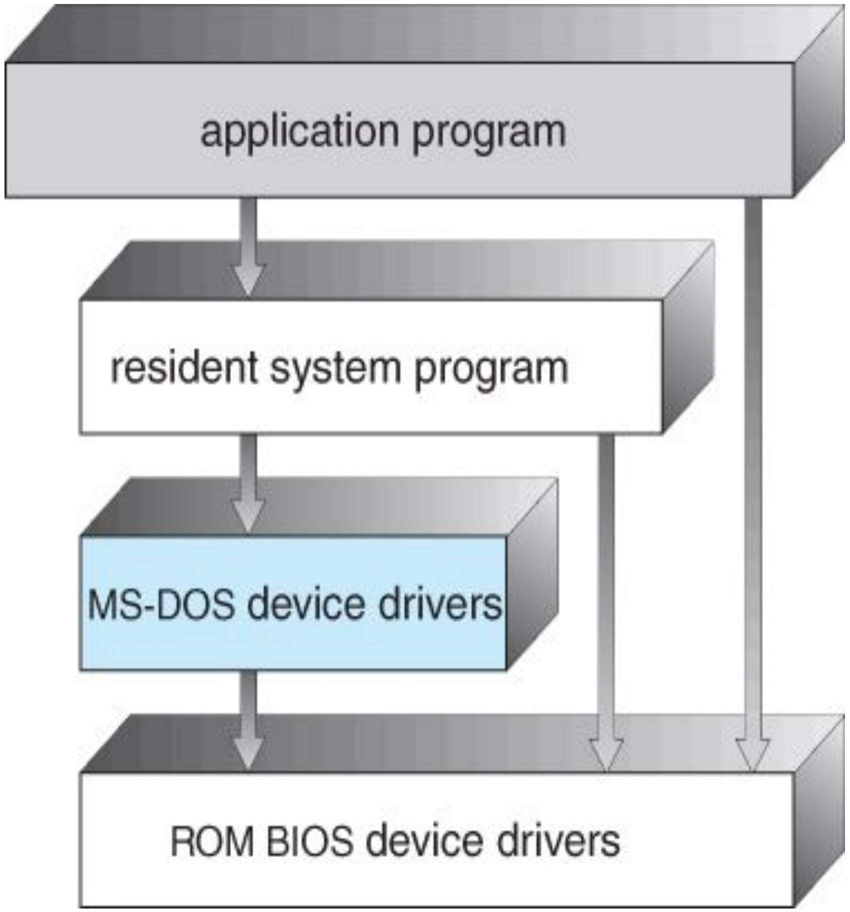
System Structure – Simple Approach

A small, simple and limited systems that do not have a well defined structure

All layers has direct access to base hardware.

Example

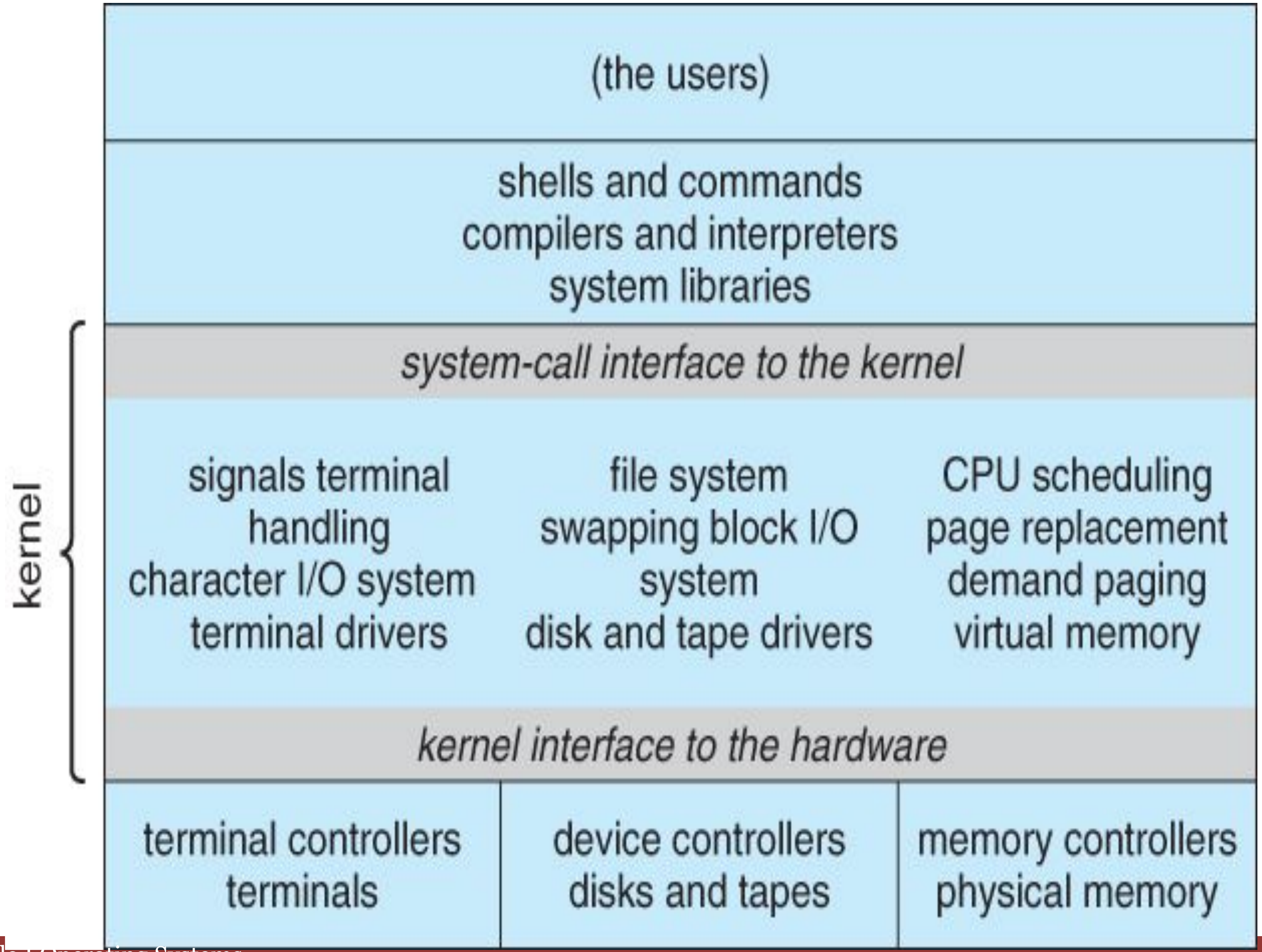
- MS-DOS – written to provide most functionality in least space
  - not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



MS-DOS Layer structure

1. Monolithic Structure (UNIX Structure)

Too many functions packed into one level called kernel



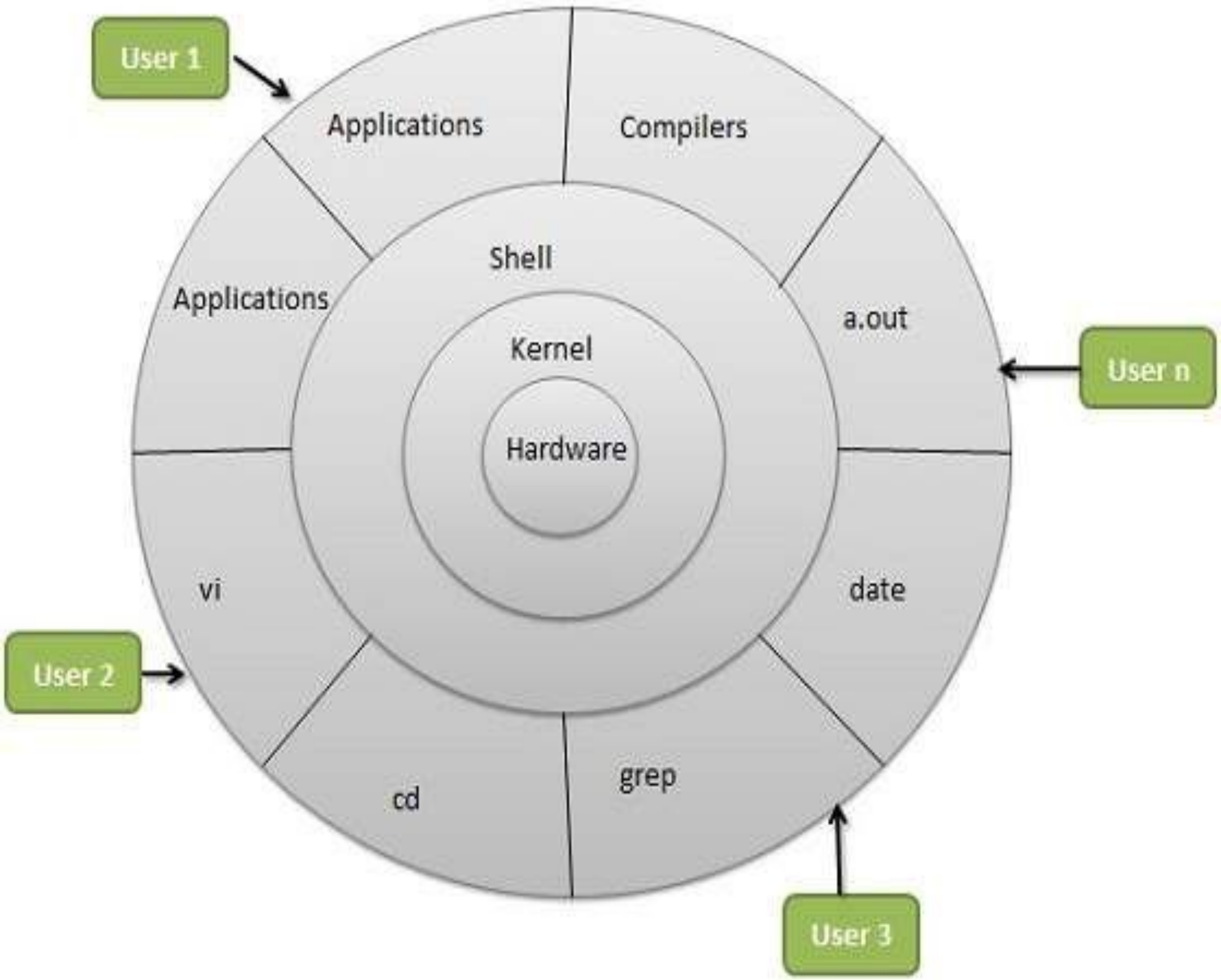
UNIX System Structure

- UNIX – limited by hardware functionality, the original UNIX OS had limited structuring. The UNIX OS consists of two separable parts.
  - Systems programs: use kernel supported system calls, provide functions such as compilation, file manipulation.
  - The kernel: separated into interfaces and device drivers
     Consists of everything below the system-call interface and above the physical hardware
     Provides the file system, CPU scheduling, memory management, and other OS functions through system call; a large number of functions for one level.

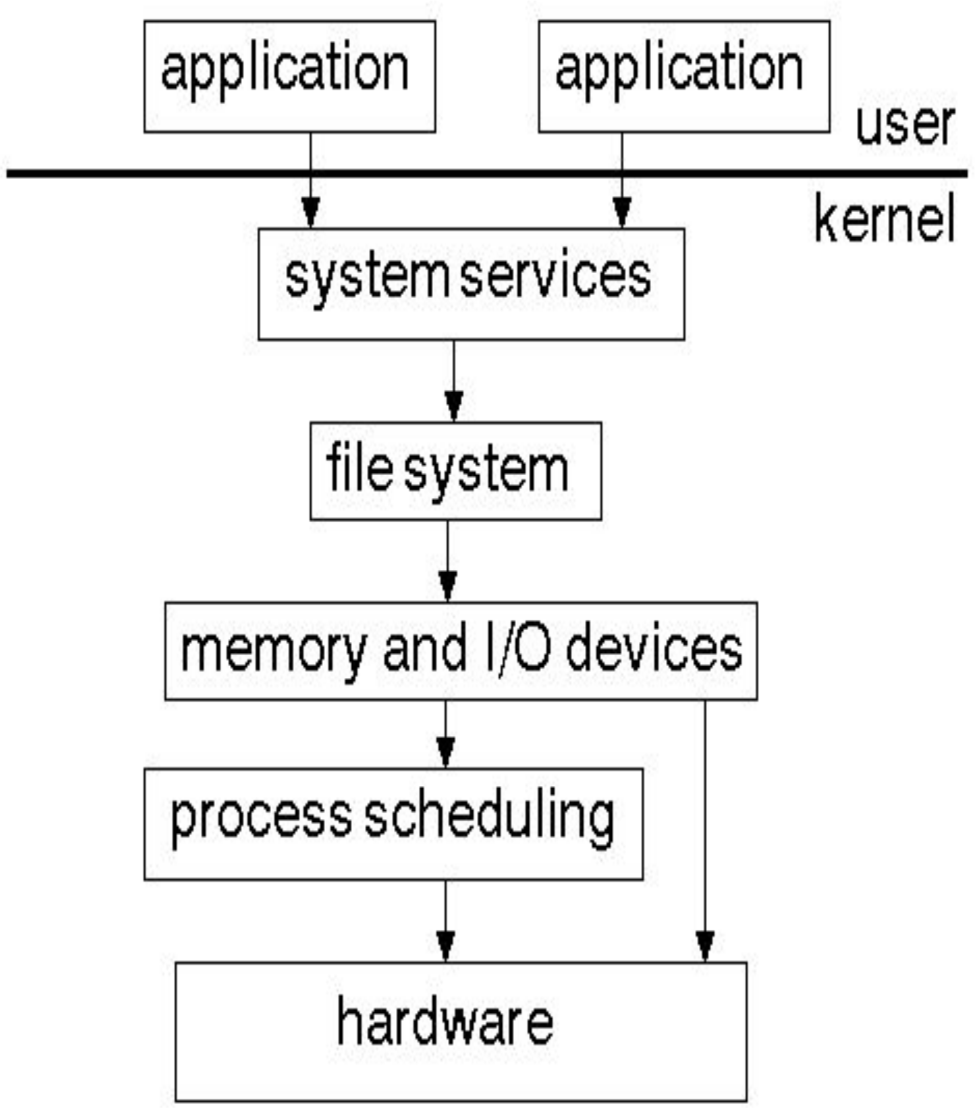
2. System Structure – Layered Approach

- The OS is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.
- Design and implementation of system is simplified.
- Simplifies debugging and system verification
- Each layer hides the data structures, operations and hardware from higher-level layers (ensures protection)
- Disadvantages:
- Difficulty in deciding the layer position
- Slow in getting services (system calls has to pass to each lower layer)

An Operating System Layer



Layered Model



Advantages

- Simplicity of Construction
- Debugging
- With modularity, layers are selected such that each layer uses functions (operations) and services of only lower-level layers. This simplifies debugging and system verification.
- 1st layer can be debugged without any concern from the rest of the system since layer 1 uses only the basic hardware to implement its functions.
- The correct functioning of layer 1 can be assumed while debugging layer 2 and so on…
- If an error is found during the debugging of a particular layer, the error must be on that layer since the layers below are already debugged,
- Thus the design and implementation of the system are simplified

Disadvantages

- Proper definition of layers is difficult
- Since a layer can use only lower-level layers, careful planning is necessary.
- Eg: The device driver for the disk space and virtual memory should be at a lower level than the memory management routines since memory management requires the ability to use the disk space.
- Less efficient than other types.
- When a user program executes an I/O operation, it executes a system call, which calls the memory management layer, which in turn calls the CPU scheduling layer, which is then passed to the hardware.
- At each layer, the parameters may be modified, data may need to be passed etc.
- Each layer adds overhead to the system call . Hence system calls take longer time than ones in non-layered system.

4. Kernel

- It is a bridge between applications and the actual data processing done at the hardware level.
- It manages operations of computer and hardware - most notably memory and CPU time.
- Manages system's resources (the communication between hardware and software components)

There are two types of kernels:

  - A microkernel, which only contains basic functionality;
  - A monolithic kernel, which contains entire OS.

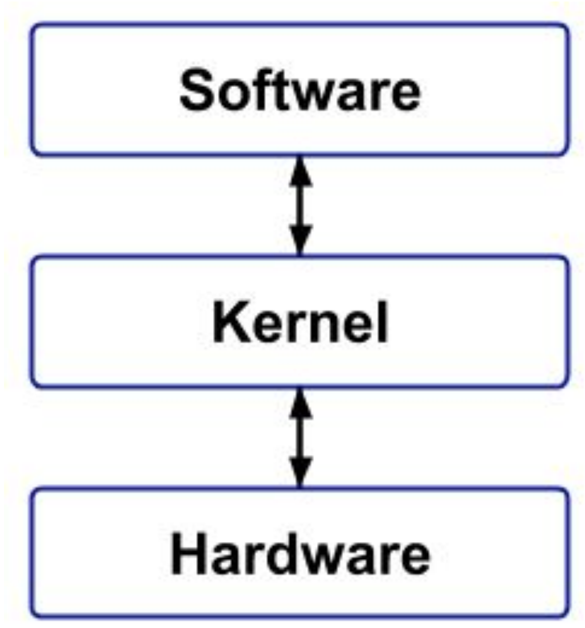
Functions of kernel

- Access Computer resource
- Resource Management
- Memory Management
- Device Management

Monolithic kernel

- Monolithic Kernels are those Kernels where the user services and the kernel services are implemented in the same memory space i.e. different memory for user services and kernel services are not used in this case.
- By doing so, the size of the Kernel is increased and this, in turn, increases the size of the Operating System. As there is no separate User Space and Kernel Space, so the execution of the process will be faster in Monolithic Kernels.

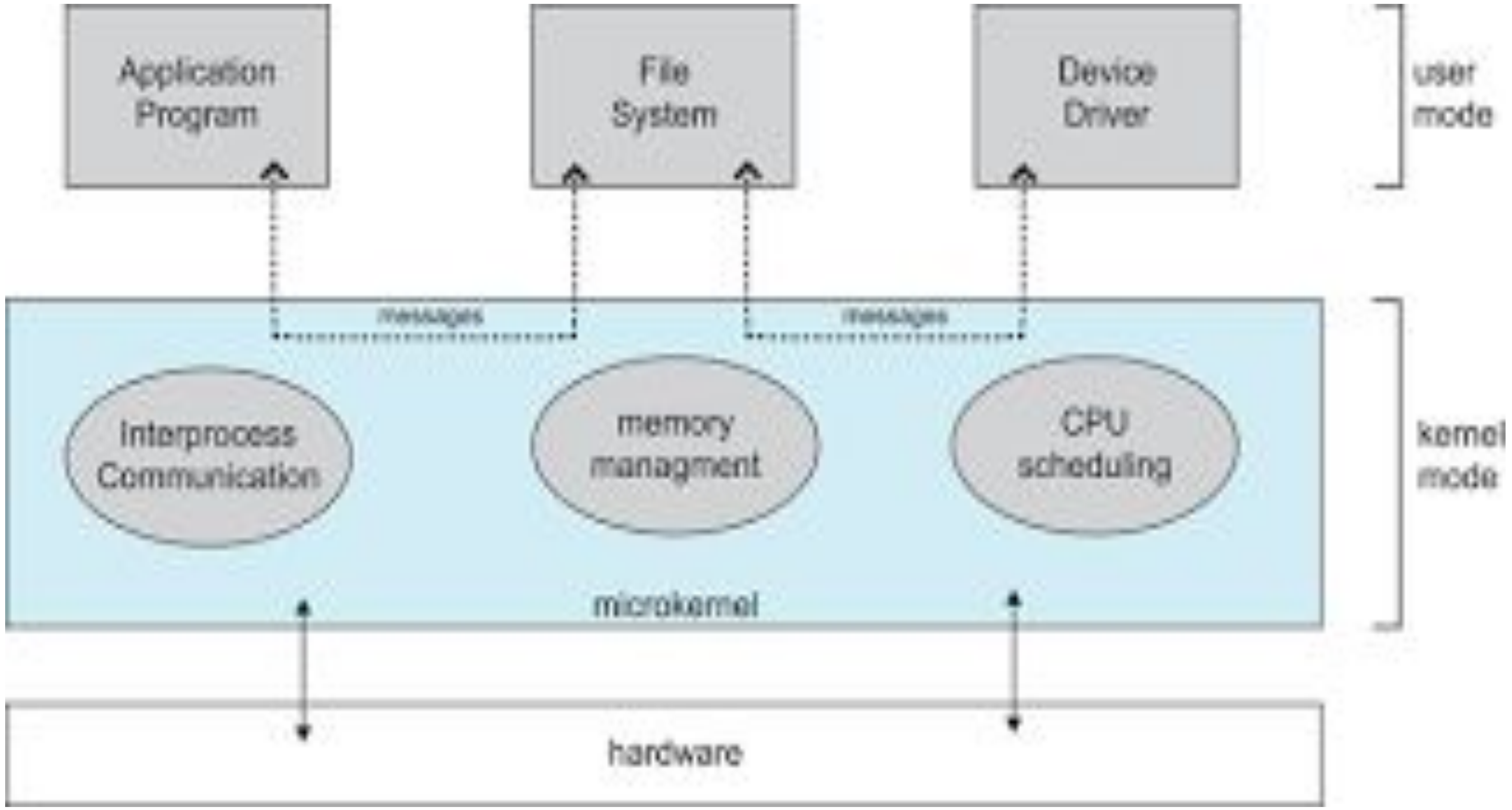
Monolithic kernel



- **Advantages:**
- It provides CPU scheduling, memory scheduling, file management through System calls only.
- Execution of the process is fast because there is no separate memory space for user and kernel.
- **Disadvantages:**
- If any service fails, then it leads to system failure.
- If new services are to be added then the entire Operating System needs to be modified.

Microkernel (Layered)

- To reduce kernel complexity, nonessential components removed from the kernel and implement as system and user-level programs; results smaller kernel.
- Main function is to provide communication facility between client program and various services running in user space as message passing.
- Microkernels provide minimal process and memory management, in addition to communication facility.
- Benefit is the ease of extending the OS i.e. all new services are added to user space, hence no kernel modification required.
- Easier to port one hardware design to another.
- Provides more security and reliability; if a service fails rest of the OS remain untouched.
- Performance issue


Application Program
File System
Device Driver
user mode
messages
messages
Interprocess Communication
memory managment
CPU scheduling
kernel mode
microkernel
hardware

- Communication provided through message passing.
- Eg: If the client pgm needs to access a file, it must interact with the file server.
- The client program and service never interact directly, they communicate indirectly by exchanging messages with the microkernel.
- Disadvantage: The performance of micro kernels can suffer due to increase in system-function overhead.

Advantage

(i) Extension of OS is easier.

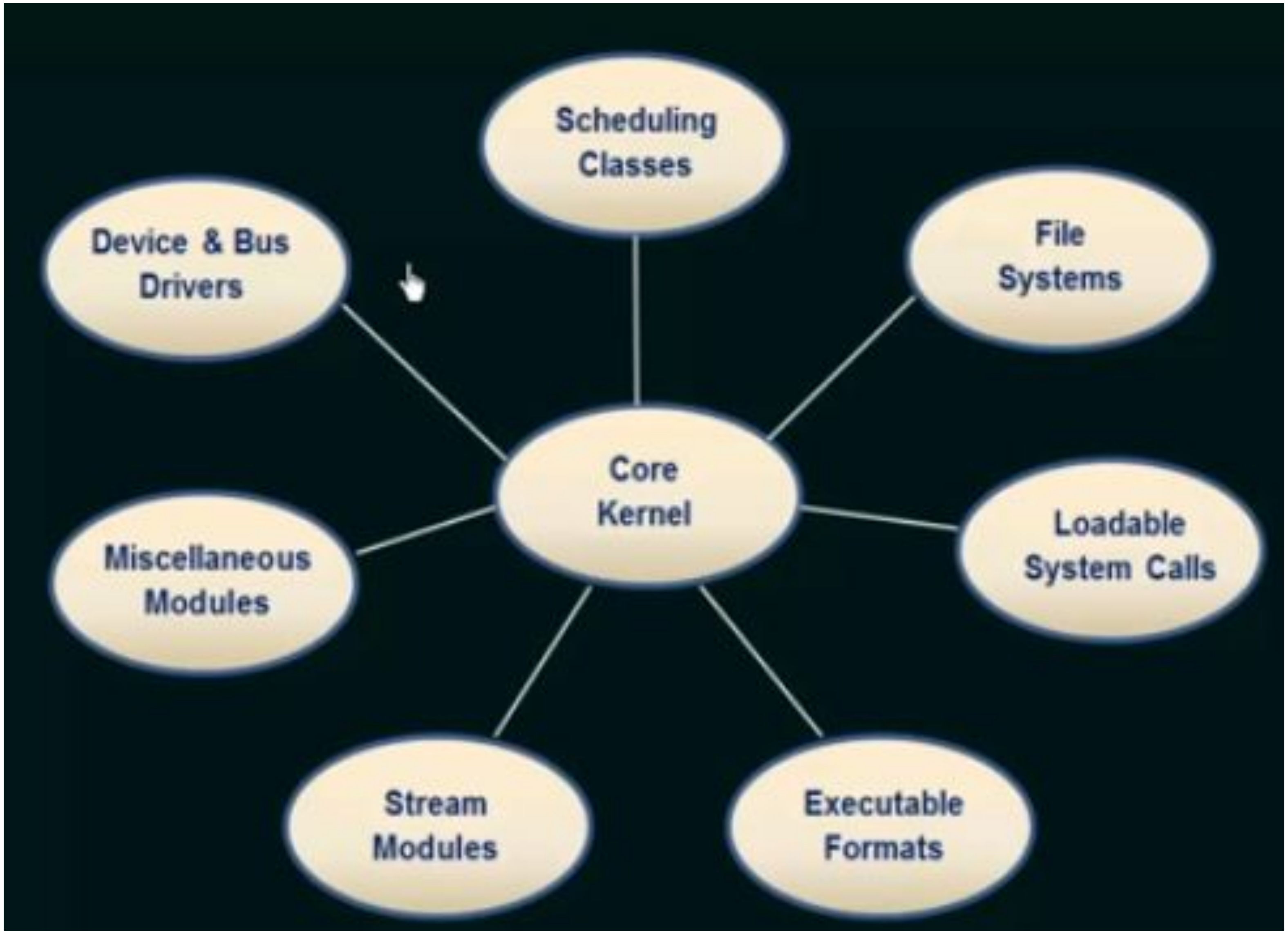
- All new services are added to the user space and do not require modification to the kernel.
- If the kernel needs to be modified, the microkernel tends to be small. So changes made will also be small.
- The kernel is easy to port from one hardware design to another.

(ii) Microkernel provides more security and reliability. Since most services are running as user process rather than kernel process.

- If a service fails, most of the OS remains untouched.

Modular kernel

- Better than layered structure
- Each module can be directly loaded and interacted
- Provides security
- Microkernel requires message passing


Scheduling Classes
Device & Bus Drivers
File Systems
Core Kernel
Miscellaneous Modules
Loadable System Calls
Stream Modules
Executable Formats

- The best current methodology for OS design involves using loadable kernel modules.
- Here, the kernel has a set of core components and links to additional services via modules, either at boot time or at run time.
- This is most common in UNIX as well as Windows.
- The idea is for the kernel to provide core services while other services are implemented dynamically, as the kernel is running.
- Linking services dynamically is preferable to adding new features directly to the kernel. Otherwise the kernel needs to be recompiled every time a change was made.

Shell

- The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell.
- OS shells use either a command-line interface (CLI) or GUI, depending on a computer's role and particular operation.
- It interprets commands the user types in and arranges for them to be carried out.
- It is named a shell because it is the outermost layer around the OS kernel.

CLI shell and GUI shell

CLI shells

- Uses alphanumeric characters typed on a keyboard to provide instructions and data to the operating system, interactively.
- Require user to be familiar with commands and their calling syntax, and to understand concepts about shell-specific scripting language.

Graphical shells

- Allowing for operations such as opening, closing, moving and resizing windows, as well as switching focus between windows
- Place a low burden on beginning computer users, and are characterized as being easy to use. Most GUI-enabled operating systems also provide CLI shells

Summary

- An operating system is software that manages the computer hardware, as well as providing an environment for application programs to run.
- Perhaps the most visible aspect of an operating system is the interface to the computer system it provides to the human user.

References

1. Andrew S. Tanenbaum, “Modern Operating Systems”, Prentice Hall
2. 2. J. L. Peterson and A. Silberschatz , Operating System Concepts, Addison Wesley.

Thank you