Operating system structure

kashiram pokharel

Unit 2 Operating System Structure **2 Hrs.**

Introduction, Layered System, Kernel, Types of Kernel (Monolithic/Macro Kernel and Micro / Exo-Kernel), Client-Server Model, Virtual Machines, Shell.

Operating system Structure:

It covers the following aspects:

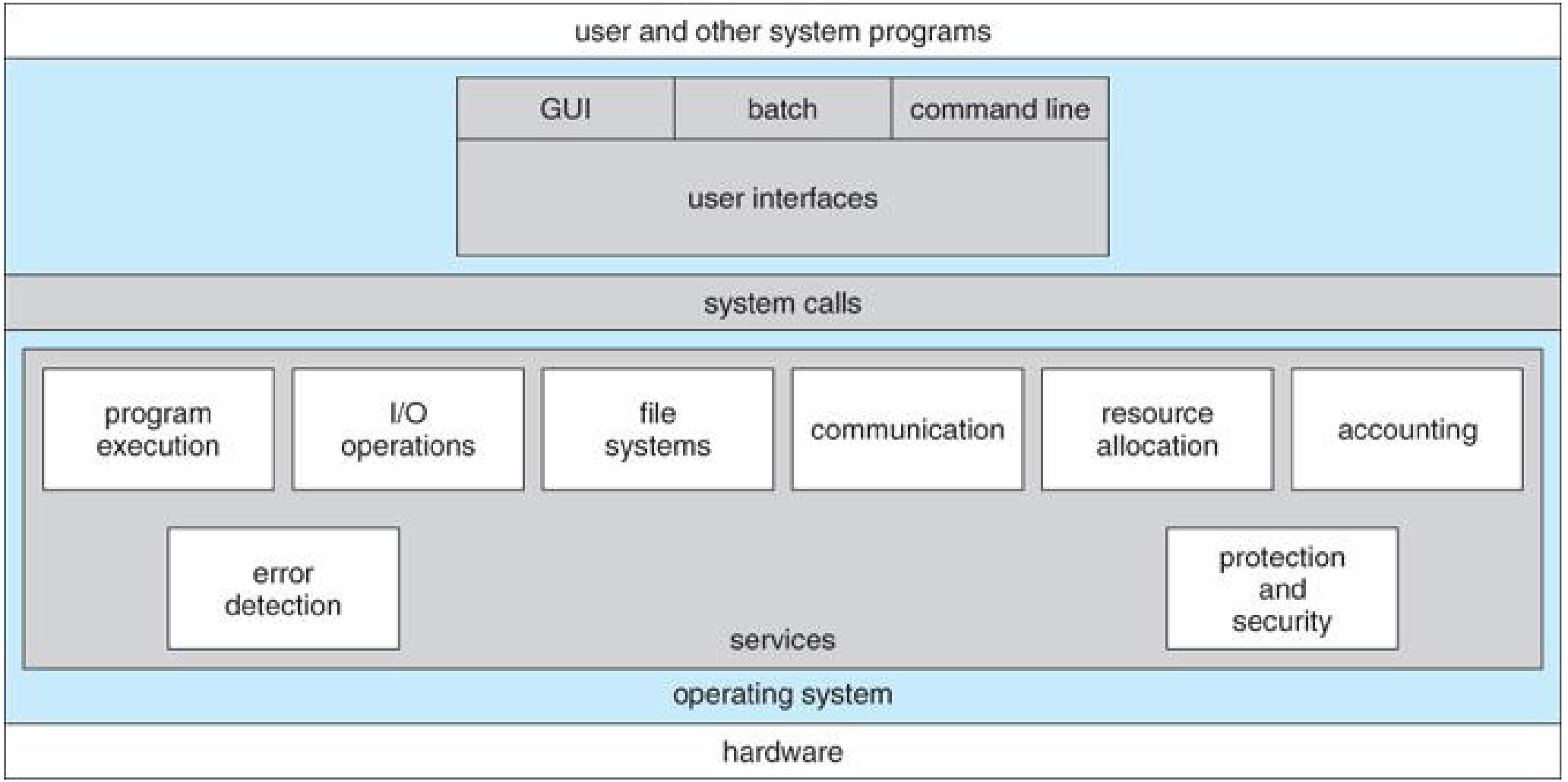
i. System Components
ii. Os services
iii. System Call
iv. System program
v. System structure
vi. System Design and implementation
vii. System Generation.

Operating system components

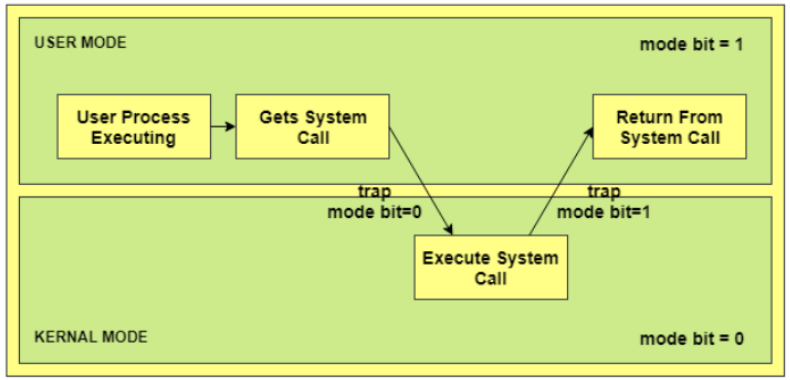
- Not all systems have the same structure many modern operating systems share the same goal of supporting the following types of system components.
  1. process management.
  2. Main memory management
  3. File management
  4. I/o management
  5. Networking
  6. Protection system
  7. Command interpretation system

  etc.

Operating system services:



System call:



- System calls **provide an interface between the process and the operating system**. **System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do.**
- Programming interface to the services provided by the OS i.e. a **system call is** how a program requests a service from an operating system's kernel.
- This may **include hardware related services** (e.g. accessing the hard disk), creating and executing new processes, and communicating with integral kernel services (like scheduling).
- System calls provide the **interface between a process and the operating system**. Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call
- On Unix, Unix-like and other POSIX-compatible operating systems, popular system calls are open, read, write, close, wait, execve, fork, exit, and kill.
- Why use APIs rather than system calls?(Note that the system-call names used throughout this text are generic)

System calls can be roughly grouped into five major categories:

1. Process Control.

- load
- execute
- create process
- terminate process
- get/set process attributes
- wait for time, wait event, signal event
- allocate, free memory

2. **File management**.

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

3. **Device Management**.

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

4. **Information Maintenance**.

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

5. **Communication.**

- create, delete communication connection
- send, receive messages
- transfer status information
- attach or detach remote devices

System Programs

- In logical hierarchy of computer system, we can see system programs lying between OS and application programs.
- System **programs provide OS functionality through separate applications**, which are not part of the kernel or command interpreters. They are also known as system utilities or system applications.
- **System programs provide a convenient environment for program development and execution**. These can be divided into:
  - File manipulation : files/directories operations
  - Status information : Date, time, space, users information
  - Programming language support : Interpreters, compilers, assembler
  - Program loading and execution : Loaders and linkers
  - Communications : Remote login, email handler, file transfer
  - **provide detailed performance**, logging, and debugging information

System program contd..

- Most users' view of the operation system is defined by system programs, not the actual system calls .**Provide a convenient environment for program development and execution.**
- System programs **provide basic functioning to users so that they do not need to write their own environment for program development** (editors, compilers) and program execution (shells). In some sense, they are bundles of useful system calls.
- Typically, these programs format and print the output to the terminal or other output devices Some systems implement a registry - used to store and retrieve configuration information

System structure:

- Interconnection of system components and their interface with kernels is the part of system structure.
- System is partitioned into modules and each module well-defined portion of the system with carefully defined inputs, outputs and functions.
- Some commonly used structures are :
  - Simple Structure
  - Layered structure
  - Monolithic structure
  - Exo-kernel
  - Micro kernel
  - Client server
  - Virtual machine etc

Simple structure:

- Simple structure was **Written to provide the most functionality in the least space and are not not divided into modules**. Although MS-DOS has some structure but do not have well defined structure and are small, simple and limited systems.
- The **interfaces and levels of functionality are not well separated**. MS-DOS is an example of such operating system. In MS-DOS application programs are able to access the basic I/O routines.
- These types of operating system cause the entire system to crash if one of the user programs fails.

Advantages of Simple structure:

- It **delivers better application performance** because of the few interfaces between the application program and the hardware.
- Easy for kernel developers to develop such an operating system.

Disadvantages of Simple structure:

- The structure is very complicated as no clear boundaries exists between modules.
- It does not enforce data hiding in the operating system.

- Started as small but grew beyond the scope
- So errors in applications can cause the whole system to crash.

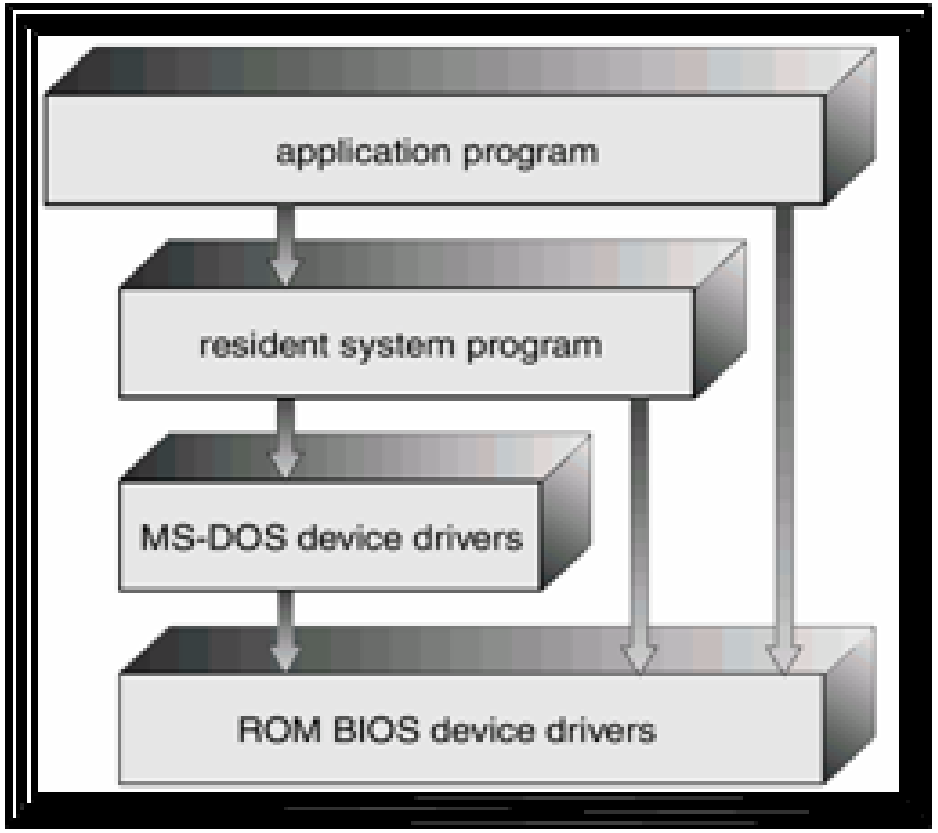


Figure Logical structure of DOS

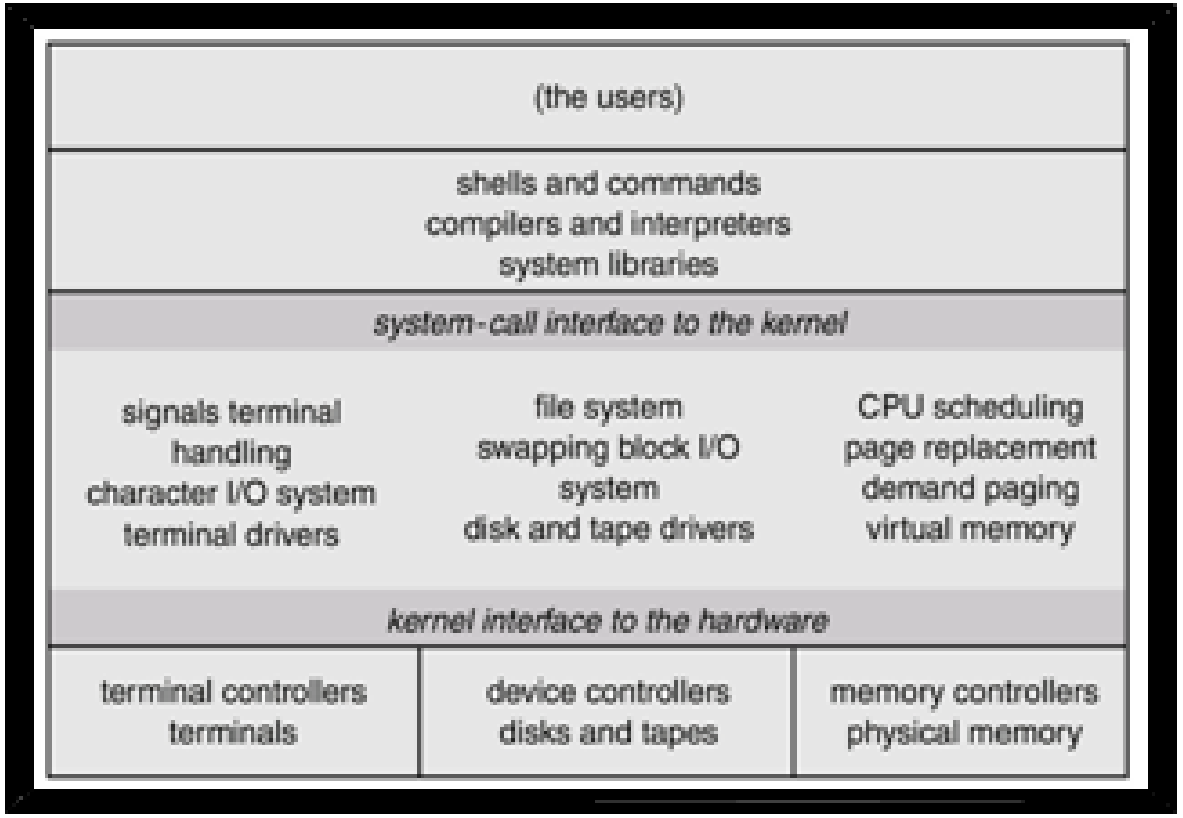
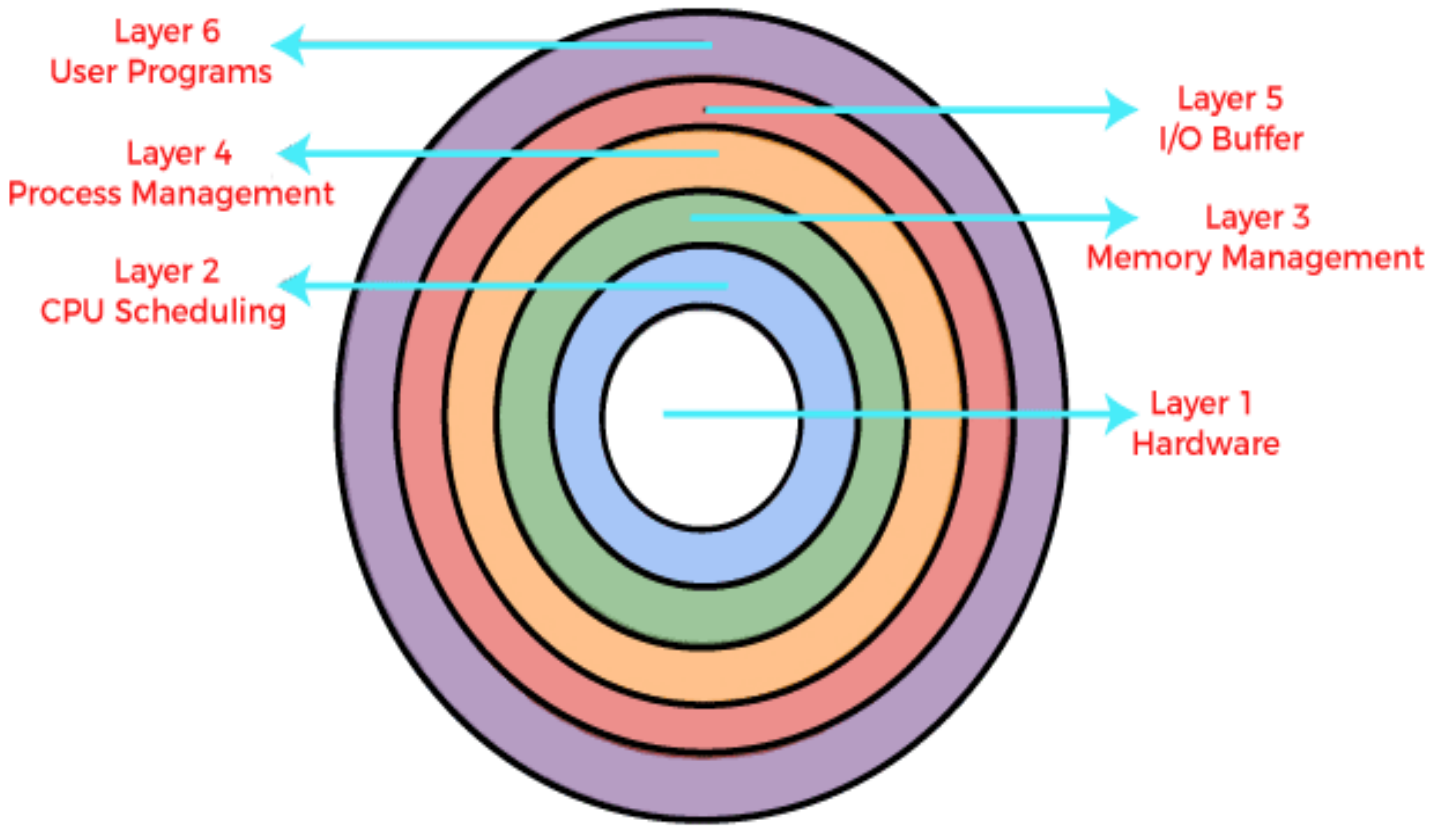
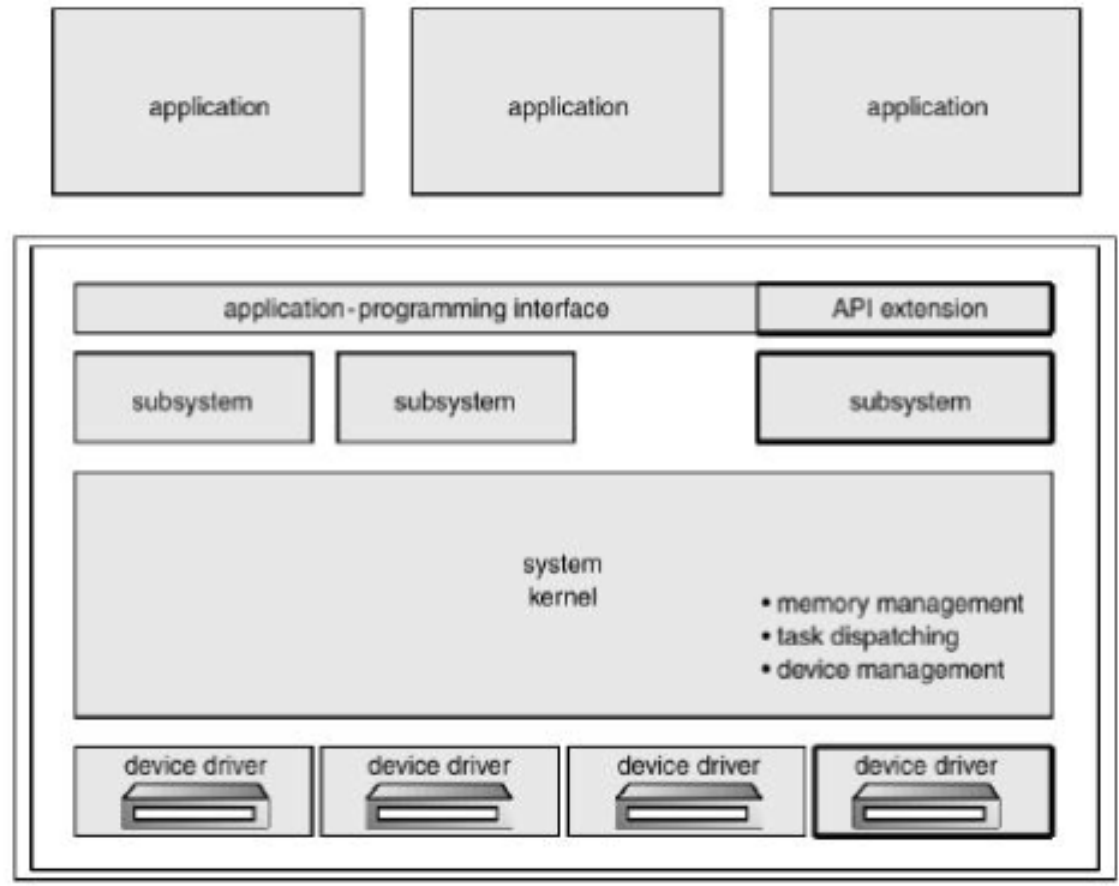


Figure Logical structure of UNIX

Layered structure:

- Layered Structure is a type of system structure **in which the different services of the operating system are split into various layers, where each layer has a specific well-defined task to perform.**
- It was created to improve the pre-existing structures like the Monolithic structure (UNIX) and the Simple structure (MS-DOS)
- It provides Hierarchy of layers, each constructed upon another below it .Layer 0 (closest to the hardware) to Layer N (closest to user interface)
- **It provides modularity**. With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.
- But, difficult in differentiation and less efficient


Layer 6
User Programs
Layer 5
I/O Buffer
Layer 4
Process Management
Layer 3
Memory Management
Layer 2
CPU Scheduling
Layer 1
Hardware


application
application
application
application - programming interface
API extension
subsystem
subsystem
subsystem
system
kernel
• memory management
• task dispatching
• device management
device driver
device driver
device driver
device driver

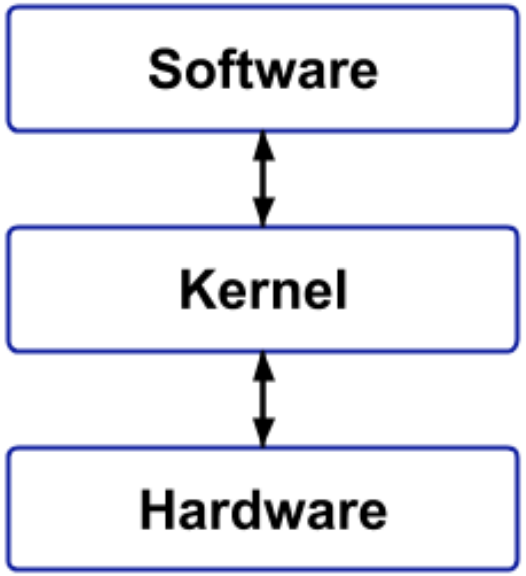
Kernel:

- Kernel is central component of an operating system that **manages operations of computer and hardware.**
- It basically **manages operations of memory and CPU time**. Kernel **acts as a bridge between applications and data processing** performed at hardware level using inter-process communication and system calls.
- Kernel loads first into memory when an operating system is loaded and remains into memory until operating system is shut down again.
- It is responsible for various tasks such as disk management, task management, and memory management.
- It decides which process should be allocated to processor to execute and which process should be kept in main memory to execute.
- It basically acts as an interface between user applications and hardware. The major aim of kernel is to manage communication between software i.e. user-level applications and hardware i.e., CPU and disk memory.

Objectives of Kernel :

- To establish communication between user level application and hardware.
- To decide state of incoming processes.
- To control disk management.
- To control memory management.
- To control task management.

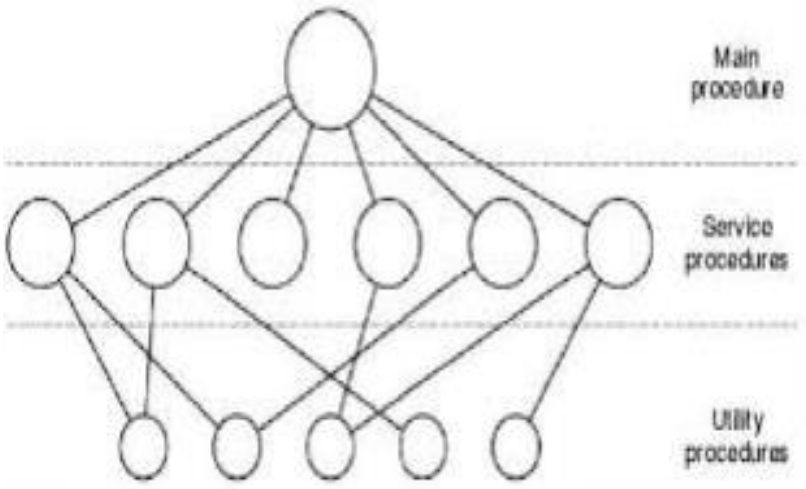
Type of kernel:(Type of OS structure)



3. **Monolithic Kernel structure:**
   - Monolithic Kernels are those Kernels where the **user services and the kernel services are implemented in the same memory space**
   - A **monolithic kernel** is an operating system architecture where the entire operating system is working in kernel space i.e. all operating system services operate in kernel space.
   - 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**.
   - It has dependencies between systems components. **It has huge lines of code which is complex.**
- Example :
   - Unix, Linux, Open VMS, XTS-400 etc.

Monolithic structure:

- The monolithic operating system is a very basic operating system **in which file management, memory management, device management, and process management are directly controlled within the kernel.**
- Here the kernel is a single large program. All the components can **directly communicate with each other and also with the kernel**.
- Functionality of the OS is invoked with simple function calls within the kernel
- Three different procedures are present:
  - **Main procedure**
    - Invokes the requested service procedure
  - **Service Procedures**
    - Carry out the system calls
    - For each system call, there is separate service procedure
  - **Utility Procedures**
    - Help the service procedures
    - Such as fetching data from user programs etc.



A simple structuring model for a monolithic system.

Advantage and disadvantage:

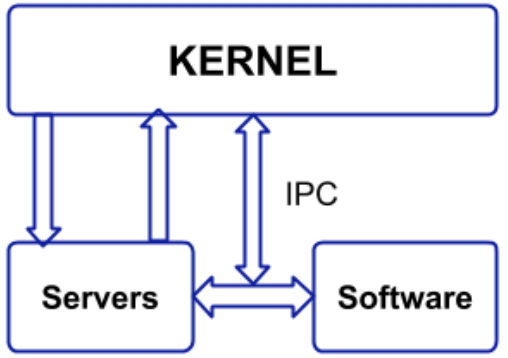
Monolithic architecture has the following advantages, such as:

- Simple and easy to implement structure.
- Faster execution due to direct access to all the services
- It has good performance. It provides CPU scheduling, memory scheduling, file management through System calls only.
- Execution of the process is fast due to direct access to all the services and there is no separate memory space for user and kernel

Disadvantages of Monolithic Architecture:

- The addition of new features or removal of obsolete features is very difficult.
- Security issues are always there because there is no isolation among various servers present in the kernel.
- It has dependencies between system component and lines of code in millions.
- 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

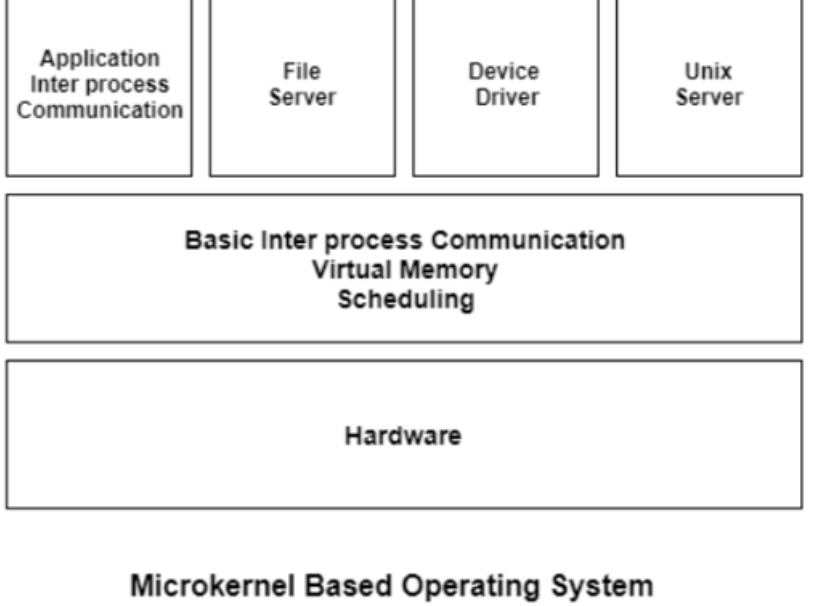
4. Micro Kernel Structure:



- in a Microkernel, the **user services and kernel services are implemented into different spaces** i.e. we use User Space and Kernel Space in case of Microkernels.
- As we are using User Space and Kernel Space separately, so it **reduces the size of the Kernel** and this, in turn, **reduces the size of Operating System**.
- It has **virtual memory and thread scheduling**. It is more stable with less services in kernel space. It puts rest in user space.
- the communication between application and services is done with the help of **message passing** and this, in turn, reduces the speed of execution.

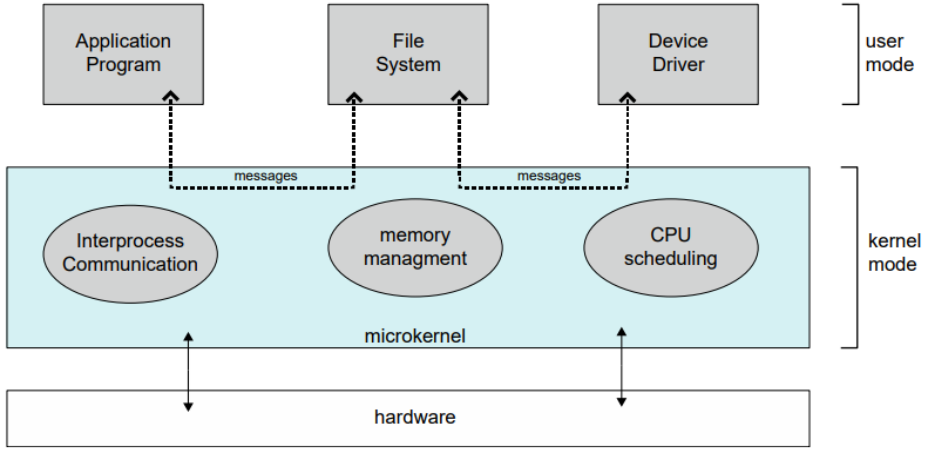
- Example :
  - Mach, L4, AmigaOS, Minix, K42 etc.

Microkernel structure:



Microkernel Based Operating System

- **Kernel** is the **core part of an operating system that manages system resources**.It also acts as a bridge between the application and hardware of the computer. It is one of the first programs loaded on start-up (after the Bootloader).
- A microkernel is the **minimum software that is required to correctly implement an operating system**. This includes memory, process scheduling mechanisms and basic inter-process communication.
- It is an architecture with kernel having the **basic interaction with hardware and the basic Inter-Process Communication mechanisms**.
- All the other Operating System services exist outside the Kernel. Microkernel **provides the flexibilities to add new features or modify existing feature**s while slightly affecting performance as it increases amount of interactions between kernel and user mode features.
- In this module **OS is splitting into small, well defined modules**. One of the modules always resides in memory and always runs on kernel mode – MICROKERNEL and Others runs as user process (device drivers, file system)
- **If a service crashes, it never affects the working of a microkernel.**



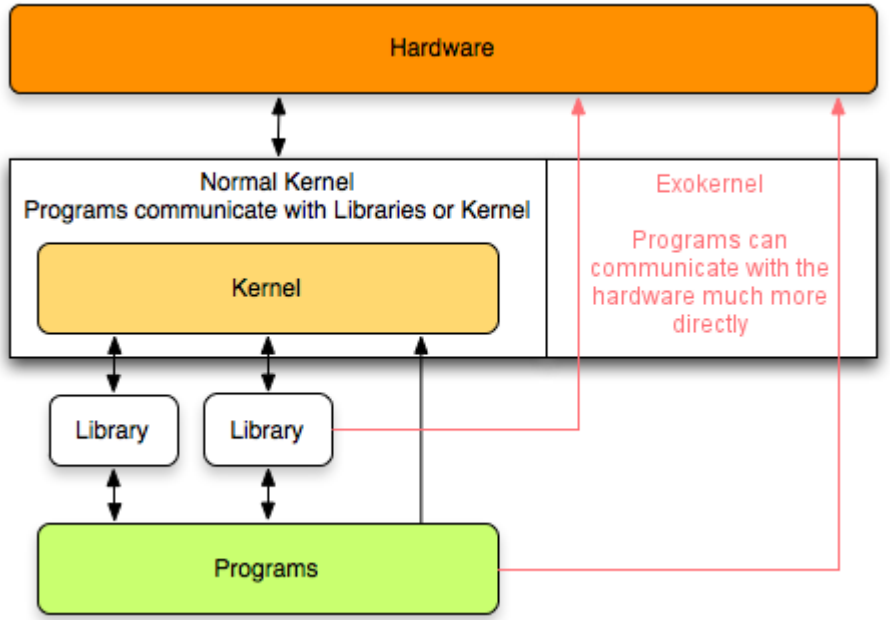
Advantage:

- Easier to extend a microkernel
- It is more stable. If new services are to be added then it can be easily added.
- More reliable (less code is running in kernel mode)
- More secure

Disadvantage:

- Performance overhead of user space to kernel space communication
- There are lots of system calls and context switches.
- Since we are using User Space and Kernel Space separately, so the communication between these can reduce the overall execution time.

5. Exo Kernel structure:



- It provides **application-level management of hardware resources**.
- **This architecture is designed to separate resource protection from management to facilitate application-specific customization.**
- It is the type of kernel which follows **end-to-end principle**. **It has fewest hardware abstractions as possible. It allocates physical resources to applications**.
- It provides **application-level management of hardware resources**.

- **This architecture is designed to separate resource protection from management to facilitate application-specific customization.**
- It is the type of kernel which follows **end-to-end principle**. **It has fewest hardware abstractions as possible. It allocates physical resources to applications**.

- the idea is not to implement all the abstractions. But the idea is to impose as few abstractions as possible and by doing so the abstraction should be used only when needed
- Example :
  - Nemesis, ExOS etc.
- Advantage :
  - It has fewest hardware abstractions.
- Disadvantage :
  - There is more work for application developers.
  - The design of the Exokernel is very complex.

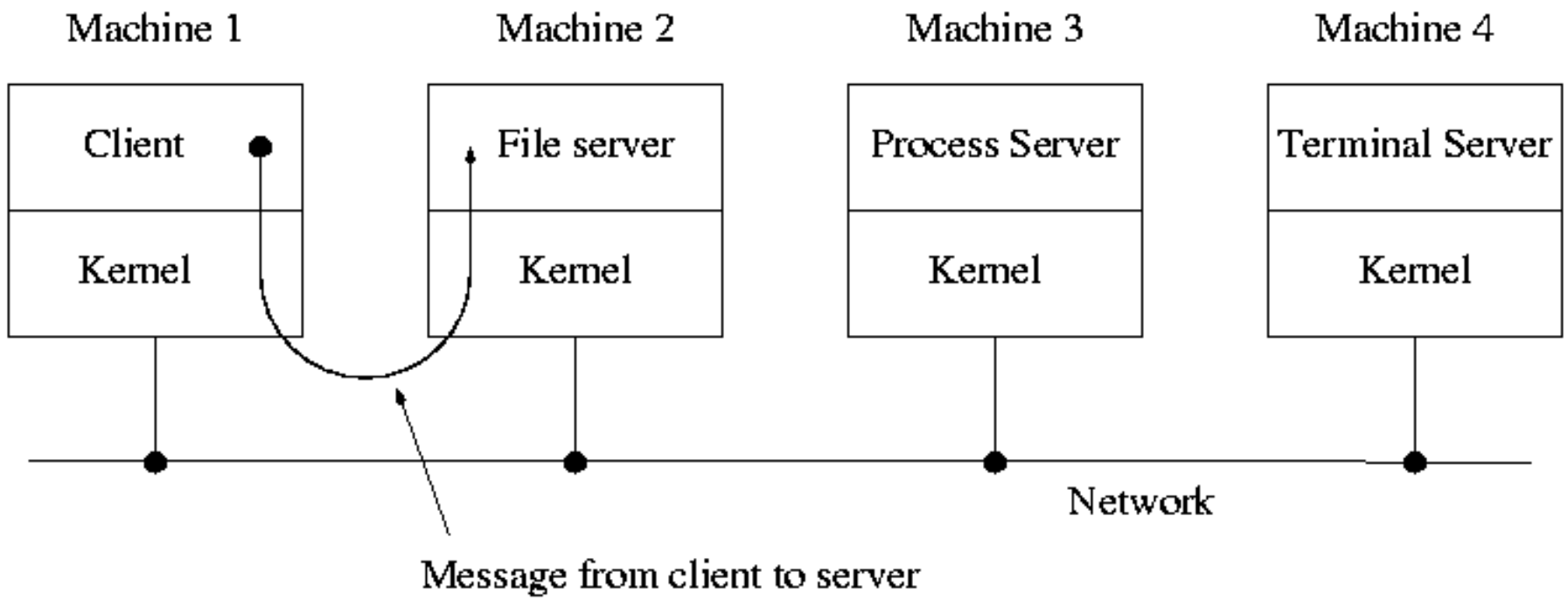
- Nano Kernel –
- It is the type of kernel that offers hardware abstraction but without system services. Micro Kernel also does not have system services therefore the Micro Kernel and Nano Kernel have become analogous.

- Example :

- EROS etc.
- Advantage :
- It offers hardware abstractions without system services.
- Disadvantage :
- It is quite same as Micro kernel hence it is less used.

Client-server structure:

- **client-server architecture**, architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns.
- The Client-server model is a distributed application structure that partitions task or workload between the providers of a resource or service, called servers, and service requesters called clientsSource:
- Two classes of processes - Server and Clients
- Communication between client and server is via message passing
- Client and server can run on different computers connected by LAN/WAN
- Servers run as user mode. Hence, no system down even if the server crashed
- Well adopted in distributed system

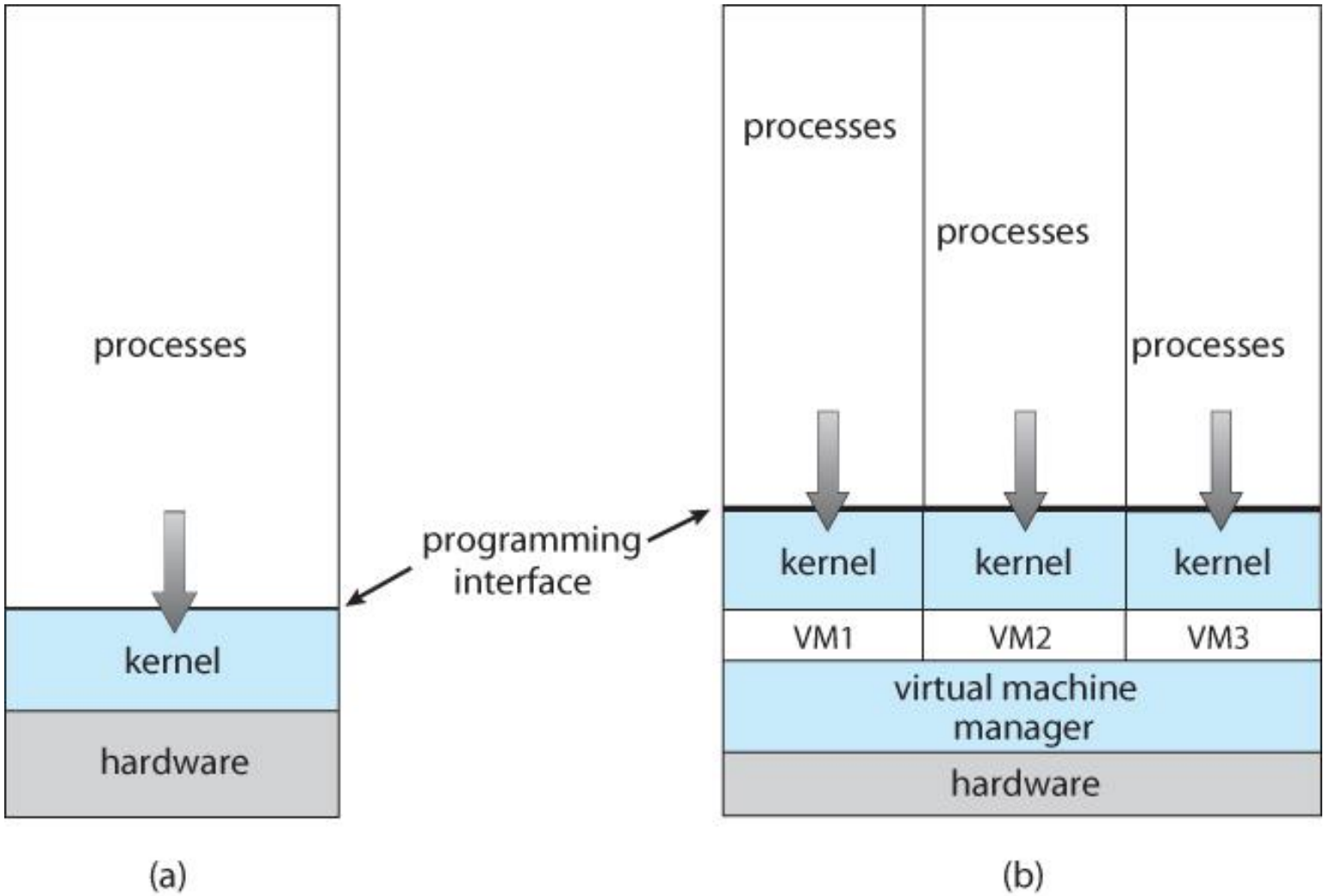

Machine 1
Machine 2
Machine 3
Machine 4
Client
File server
Process Server
Terminal Server
Kernel
Kernel
Kernel
Kernel
Network
Message from client to server

Virtual Machine:

- Virtualization is the process of creating a software-based, or "virtual" version of a computer, with dedicated amounts of CPU, memory, and storage that are "borrowed" from a physical host computer.
- A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface identical to the underlying bare hardware.
- The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory.

Advantage:

- **The virtual-machine concept provides complete protection of system resources.**
- **This isolation, however, permits no direct sharing of resources**
- **System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.**



- Non virtual machine system with VM

Shells:

- Your **interface to the operating system** is called a *shell*.
- The shell is the **outermost layer of the operating system**.
- Shells incorporate a programming language to control processes and files, as well as to start and control other programs.
- The shell manages the **interaction between actor and the operating system by prompting you for input, interpreting that input for the operating** system, and then handling any resulting output from the operating system.
- **Shells provide a way for you to communicate** with the operating system. This communication is carried out either interactively (input from the keyboard is acted upon immediately) or as a shell script.
- A *shell script* is a sequence of shell and operating system commands that is stored in a file.

- A shell is software that provides an interface between users and operating system of a computer to access the services of a kernel.
- There are two types of shell:
  - Command-line shell (eg. Bash(sh), Command Prompt(cmd), C shell, Bourne shell, Korn shell(ksh) etc.)
  - GUI Shell (eg. Windows Explorer or Windows Shell)
- A third type of shell is recently developed – a GCLI(Graphical Command Line Interface) shell. A GCLI shell combines the features of both CLI and GUI shell and provides an interface which is both user-friendly and powerful.

Thank you