Operating System

(Chapter-1)

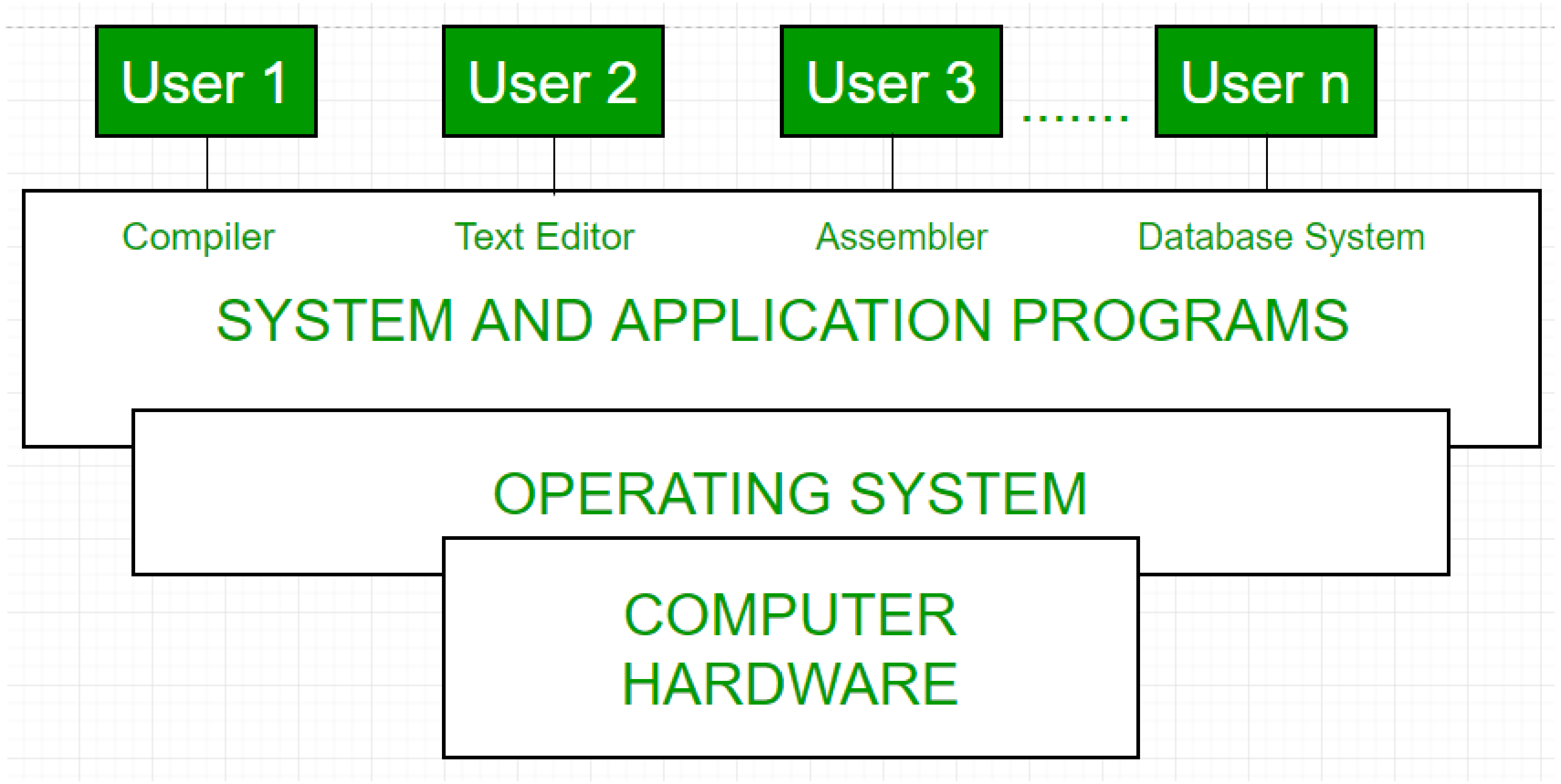
Syllabus Content

- 1.1 Introduction, Objectives, Functions and Evolution of Operating System
- 1.2 Operating system structures: Layered, Monolithic and Microkernel
- 1.3 Linux Kernel, Shell and System Calls

What is an Operating System?

- Computer System = Hardware + Software
- Software = Application Software + System Software(OS)
- An Operating System is a system Software that acts as an intermediary/interface between a user of a computer and the computer hardware.
- 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

What is an Operating System?



What is an Operating System?

- Every computer must have an operating system to run other programs.
- The operating system coordinates the use of the hardware among the various system programs and application programs for various users.
- It simply provides an environment within which other programs can do useful work.
- The operating system is a set of special programs that run on a computer system that allows it to work properly.
- It performs basic tasks such as recognizing input from the keyboard, keeping track of files and directories on the disk, sending output to the display screen, and controlling peripheral devices.

What is an Operating System?

OS is designed to serve two basic purposes:

- **It controls the allocation and use of the computing System's resources among the various user and tasks.**
- **It provides an interface between the computer hardware and the programmer that simplifies and makes it feasible for coding, creation, debugging of application programs.**

What is an Operating System?

- The Operating system must support the following tasks. The tasks are:
- Provides the facilities to create, modification of programs and data files using an editor.
- Access to the compiler for translating the user program from high-level language to machine language.
- Provide a loader program to move the compiled program code to the computer's memory for execution.
- Provide routines that handle the details of I/O programming.

Need of Operating System

- An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.
- A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being application programs.
- An operating system is concerned with the allocation of resources and services, such as memory, processors, devices, and information.
- The operating system correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, a memory management module, I/O programs, and a file system.

Functions of Operating System

Operating system performs three functions:

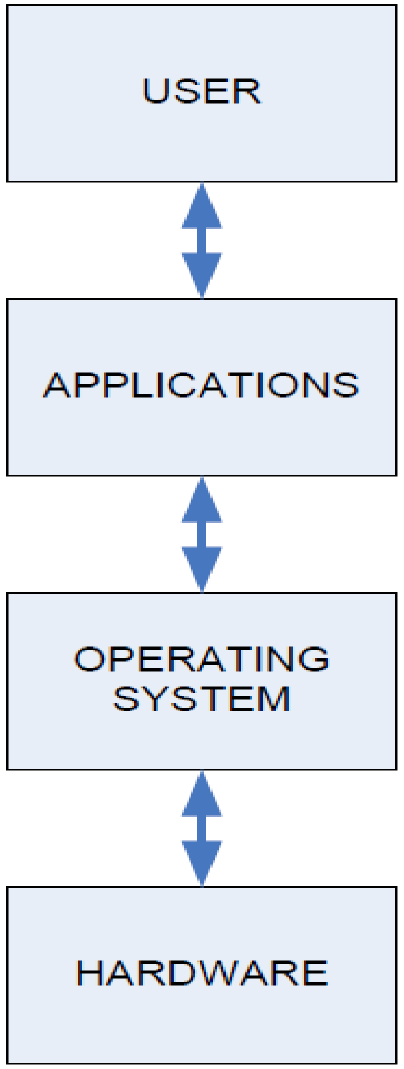
- **Convenience: An OS makes a computer more convenient to use.**
- **Efficiency: An OS allows the computer system resources to be used efficiently.**
- **Ability to Evolve: An OS should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions at the same time without interfering with service.**
- **Throughput: An OS should be constructed so that It can give maximum throughput(Number of tasks per unit time).**

Exam Questions

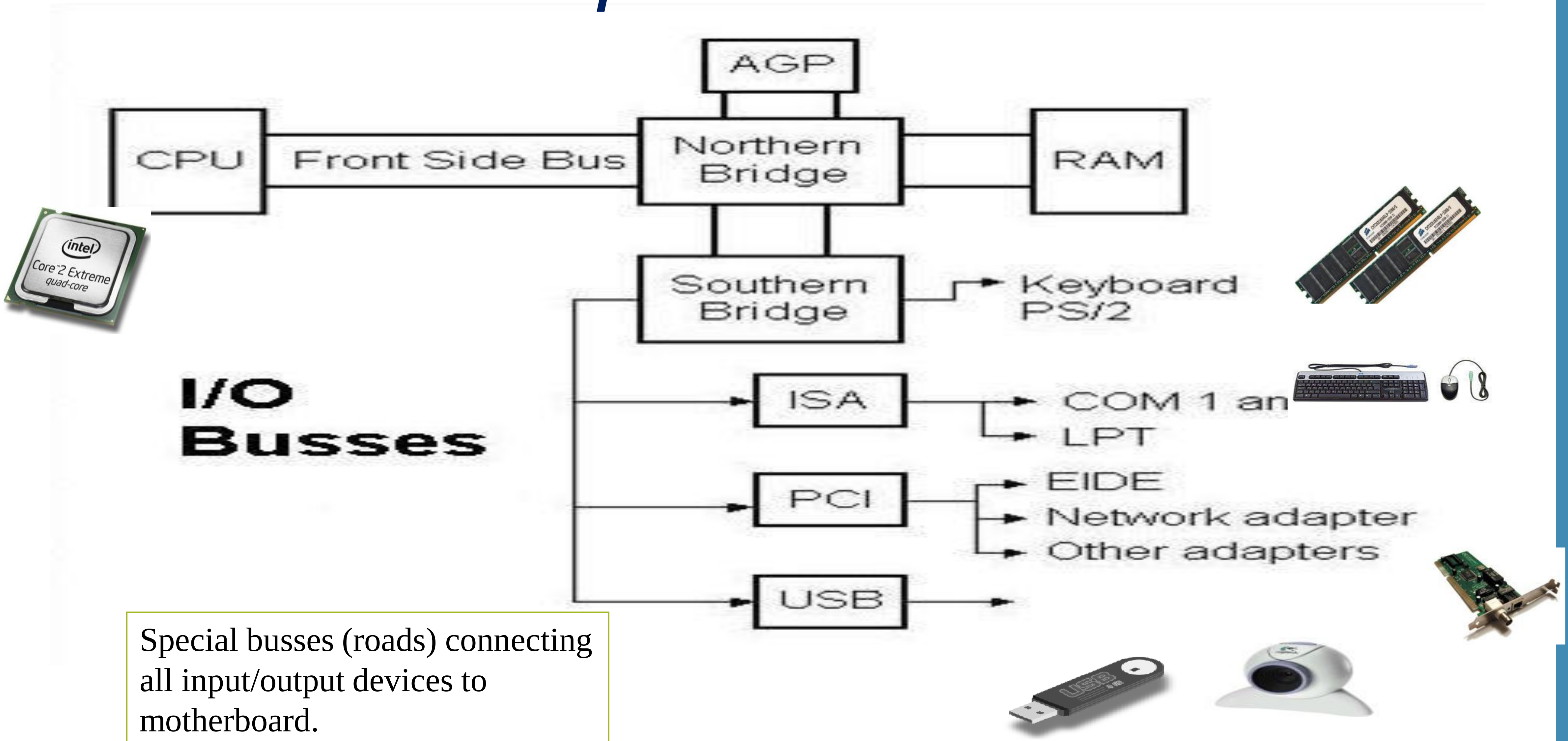
1) **Define OS and its different functionalities**
2) **Brief functions of OS**

The Structure of Computer Systems

- Accessing computer resources is divided into *layers*.
- Each layer is isolated and only interacts directly with the layer below or above it.
- If we install a new hardware device
  - ✓ No need to change anything about the user/applications.
  - ✓ However, you do need to make changes to the operating system.
  - ✓ You need to install the device drivers that the operating system will use to control the new device.
- If we install a new software application
  - ✓ No need to make any changes to your hardware.
  - ✓ But we need to make sure the application is supported by the operating system
  - ✓ user will need to learn how to use the new application.
- If we change the operating system
  - ✓ Need to make sure that both applications and hardware will compatible with the new operating system.



Computer Architecture



Special busses (roads) connecting all input/output devices to motherboard.

CPU – Central Processing Unit

- ➢ This is the brain of your computer.
- ➢ It performs all of the calculations.
- ➢ In order to do its job, the CPU needs commands to perform, and data to work with.
- ➢ The instructions and data travel to and from the CPU on the system bus.
- ➢ The operating system provides rules for how that information gets back and forth, and how it will be used by the CPU.

RAM – Random Access Memory

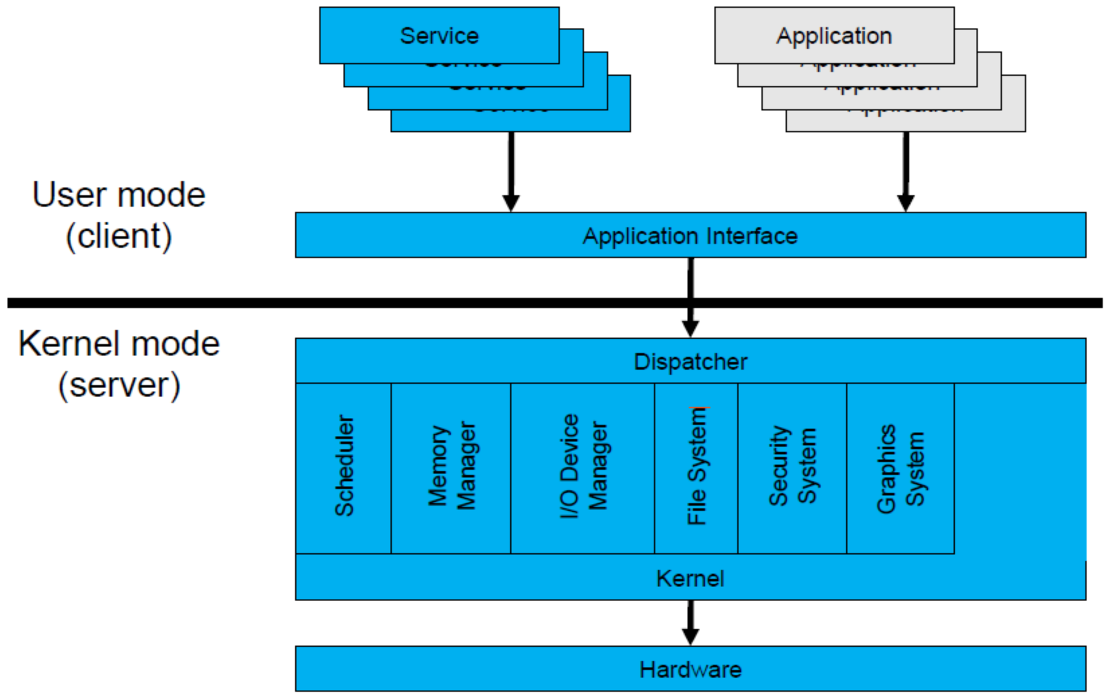
- This is like a desk, or a workspace, where your computer temporarily stores all of the information (data) and instructions (software or program code) that it is currently using.
- Each RAM chip contains millions of address spaces.
- Each address space is the same size, and has its own unique identifying number (address).
- The operating system provides the rules for using these memory spaces, and controls storage and retrieval of information from RAM.
- Device drivers for RAM chips are included with the operating system.

Problem: *If RAM needs an operating system to work, and an operating system needs RAM in order to work, how does your computer activate its RAM to load the operating system?*

Operating System Mode

- The ***User Mode*** is concerned with the actual interface between the user and the system.
- It controls things like running applications and accessing files.

- The ***Kernel Mode*** is concerned with everything running in the background.
- It controls things like accessing system resources, controlling hardware functions and processing program instructions.
- System calls are used to change mode from User to Kernel.



Kernel

- Kernel is a software code that reside in central core of OS. It has complete control over system.
- When operation system boots, kernel is first part of OS to load in main memory.
- Kernel remains in main memory for entire duration of computer session. The kernel code is usually loaded in to protected area of memory.
- Kernel performs it’s task like executing processes and handling interrupts in kernel space.
- User performs it’s task in user area of memory.
- This memory separation is made in order to prevent user data and kernel data from interfering with each other.
- Kernel does not interact directly with user, but it interacts using SHELL and other programs and hardware.

Kernel cont…

- **Kernel includes:-**

  1. **Scheduler**: It allocates the Kernel's processing time to various processes.

  2. **Supervisor**: It grants permission to use computer system resources to each process.

  3. **Interrupt handler** : It handles all requests from the various hardware devices which compete for kernel services.

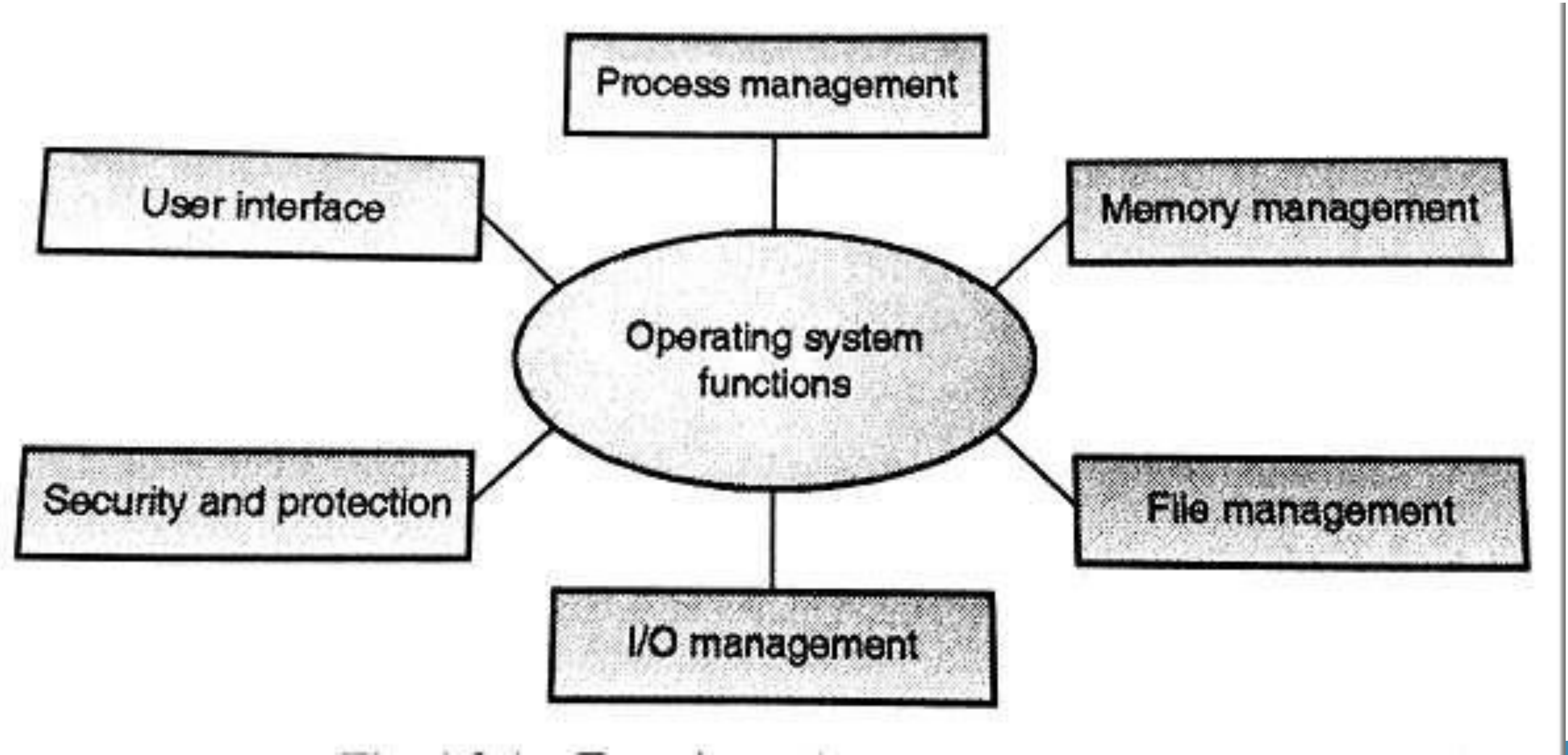
  4. **Memory manager** : allocates space in memory for all users of kernel service.

- kernel provides services for process management, file management, I/O management, memory management.
- System calls are used to provide these type of services.

System Call

- **System call is the programmatic way in which a computer program/user application requests** a service from the kernel of the operating system on which it is executed.
- Application program **is just a user-process**.
- Due to **security reasons , user applications are not given access to privileged resources**(the ones controlled by OS).
- When they need to **do any I/O** or have **some more memory** or **spawn a process** or wait for **signal/interrupt,** it requests operating system to facilitate all these.
- This **request is made through System Call.**
- System calls are also called **software-interrupts.**

Functions of Operating System



1. 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.
- Simultaneous execution leads to multiple processes.
- Hence creation, execution and termination of a process are the most basic functionality of an OS
- If processes are dependent, than they may try to share same resources. thus task of process synchronization comes to the picture.
- If processes are independent, than a due care needs to be taken to avoid their overlapping in memory area.
- Based on priority, it is important to allow more important processes to execute first than others.

2. 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. When the computer made turn off everything stored in RAM will be erased automatically.
- In addition to the physical RAM installed in your computer, most modern operating systems allow your computer to use a *virtual memory system. Virtual memory allows your computer to use part of a permanent storage device (such as a hard disk) as extra memory.*
- 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 de-allocate memory space as needed.

3. File Management

- A file is a collection of related information defined by its creator.
- *File systems provide the conventions for the encoding, storage and management of data on a storage device such as a hard disk.*
  - FAT12 (floppy disks)
  - FAT16 (DOS and older versions of Windows)
  - FAT32 (older versions of Windows)
  - NTFS (newer versions of Windows)
  - EXT3 (Unix/Linux)
  - HFS+ (Max OS X)
- 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.

4. ***Device Management or I/O Management***

- *Device controllers are components on the motherboard (or on expansion cards) that act as an interface between the CPU and the actual device.*
- *Device drivers, which are the operating system software components that interact with the devices controllers.*
- A special device (inside CPU) called the Interrupt Controller handles the task of receiving interrupt requests and prioritizes them to be forwarded to the processor.
- Deadlocks can occur when two (or more) processes have control of different I/O resources that are needed by the other processes, and they are unwilling to give up control of the device.
- It performs the following activities for device management.
  - ➢ Keeps tracks of all devices connected to system.
  - ➢ Designates a program responsible for every device known as Input/output controller.
  - ➢ Decides which process gets access to a certain device and for how long.
  - ➢ Allocates devices in an effective and efficient way.
  - ➢ Deallocates devices when they are no longer required.

5. ***Security & Protection***

- The operating system uses password protection to protect user data and similar other techniques.
- It also prevents unauthorized access to programs and user data by assigning access right permission to files and directories.
- The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other.

6. *User Interface Mechanism*

- A **user interface** (**UI**) controls how you enter data and instructions and how information is displayed on the screen
- There are two types of user interfaces
  1. Command Line Interface
  2. Graphical user Interface

OS Services

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

OS Services

- Program execution

- Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.
- A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use).

Following are the major activities of an operating system with respect to program management –

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.

OS Services

➢I/O operations

- An I/O subsystem comprises of I/O devices and their corresponding driver software.
- **Drivers hide the peculiarities of specific hardware devices** from the users.
- An Operating System **manages the communication between user and device** drivers.
- I/O operation means **read or write operation** with any file or any specific I/O device.
- Operating system **provides the access to the required I/O device** when required.

OS Services

➢ **File System manipulation**

- File **represents a collection of related information.**
- **Computers can store files on the disk (secondary storage), for long-term storage purpose.**
- Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.
- **A file system is normally organized into directories for easy navigation and usage.** These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management –
- **Program needs to read a file or write a file.**
- **The operating system gives the permission to the program for operation on file.**
- **Permission varies from read-only, read-write, denied and so on.**
- **Operating System provides an interface to the user to create/delete files.**
- **Operating System provides an interface to the user to create/delete directories.**
- **Operating System provides an interface to create the backup of file system.**

OS Services

- **Communication**
- In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.
- The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication –
- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

OS Services

- Error handling
- Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling –
- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing..

OS Services

- **Resource Management**
- In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management –
- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

OS Services

- **Protection**
- Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.
- Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection –
- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

OS Structures

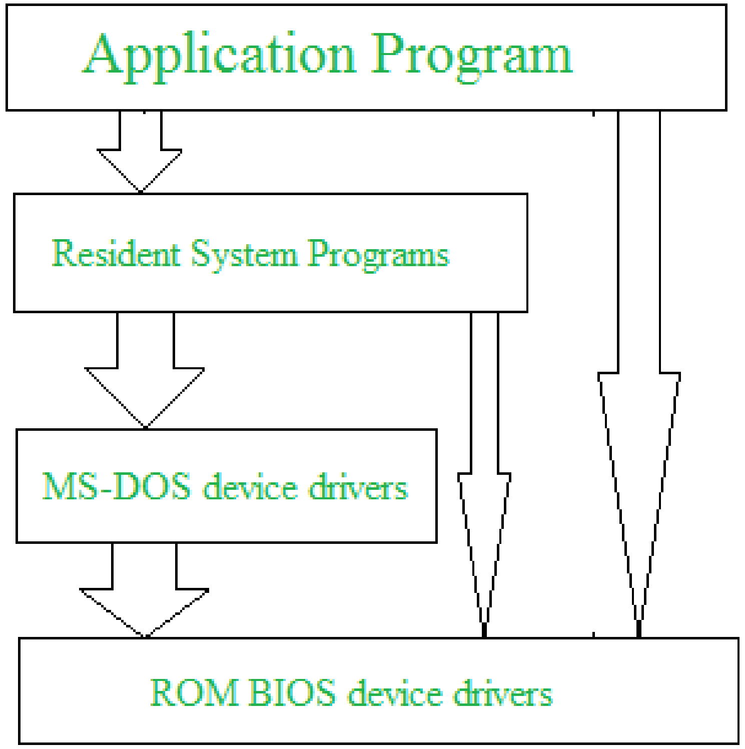
- **Simple structure:**
  Such operating systems do not have well defined structure and are small, simple and limited systems. The interfaces and levels of functionality are not well separated. MS-DOS is an example of such operating system. In MS-DOS application programs are able to access the basic I/O routines. These types of operating system cause the entire system to crash if one of the user programs fails.

Exam Questions

1) **Explain services provided by OS**

OS Structures

- Simple structure:



OS Structures

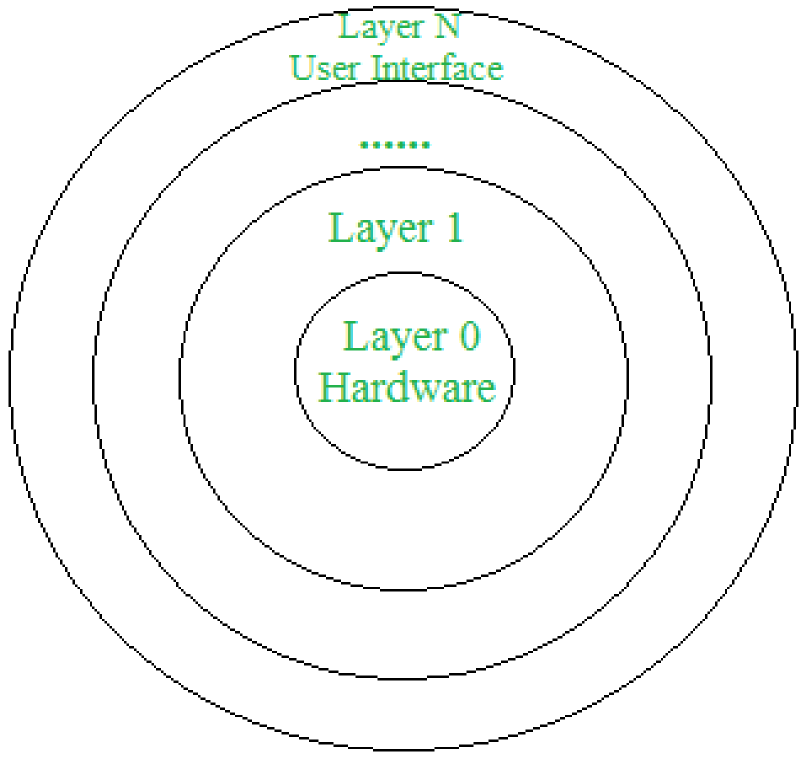
- **Advantages of Simple structure:**
- It delivers better application performance because of the few interfaces between the application program and the hardware.
- Easy for kernel developers to develop such an operating system.
- **Disadvantages of Simple structure:**
- The structure is very complicated as no clear boundaries exists between modules.
- It does not enforce data hiding in the operating system.

OS Structures

- Layered Architecture
- An OS can be broken into pieces and retain much more control on system. In this structure the OS is broken into number of layers (levels). The bottom layer (layer 0) is the hardware and the topmost layer (layer N) is the user interface. These layers are so designed that each layer uses the functions of the lower level layers only. This simplifies the debugging process as if lower level layers are debugged and an error occurs during debugging then the error must be on that layer only as the lower level layers have already been debugged.
- The main disadvantage of this structure is that at each layer, the data needs to be modified and passed on which adds overhead to the system. Moreover careful planning of the layers is necessary as a layer can use only lower level layers. UNIX is an example of this structure.

OS Structures

- Layered Architecture



OS Structures

- **Advantages of Layered structure:**
- Layering makes it easier to enhance the operating system as implementation of a layer can be changed easily without affecting the other layers.
- It is very easy to perform debugging and system verification.
- **Disadvantages of Layered structure:**
- In this structure the application performance is degraded as compared to simple structure.
- It requires careful planning for designing the layers as higher layers use the functionalities of only the lower layers.

OS Structures

- Micro-kernel:
- This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This result in a smaller kernel called the micro-kernel.
- Advantages of this structure are that all new services need to be added to user space and does not require the kernel to be modified. Thus it is more secure and reliable as if a service fails then rest of the operating system remains untouched. Mac OS is an example of this type of OS.

OS Structures

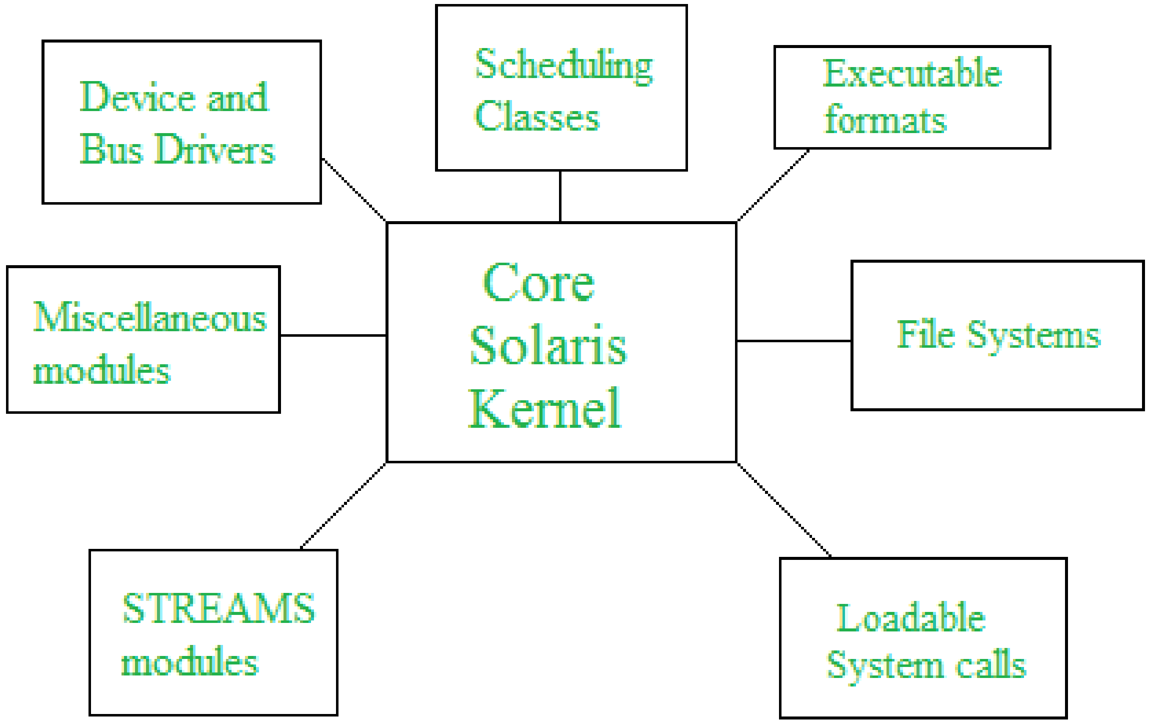
- Micro-kernel:
- Advantages of Micro-kernel structure:
- It makes the operating system portable to various platforms.
- As microkernels are small so these can be tested effectively.
- Disadvantages of Micro-kernel structure:
- Increased level of inter module communication degrades system performance.

OS Structures

- Modular structure or approach:
- It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set of core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. It resembles layered structure due to the fact that each kernel has defined and protected interfaces but it is more flexible than the layered structure as a module can call any other module.

OS Structures

- **Modular structure or approach:**



Exam Questions

1) **Explain different OS structures**

Microkernel vs Monolithic Kernel

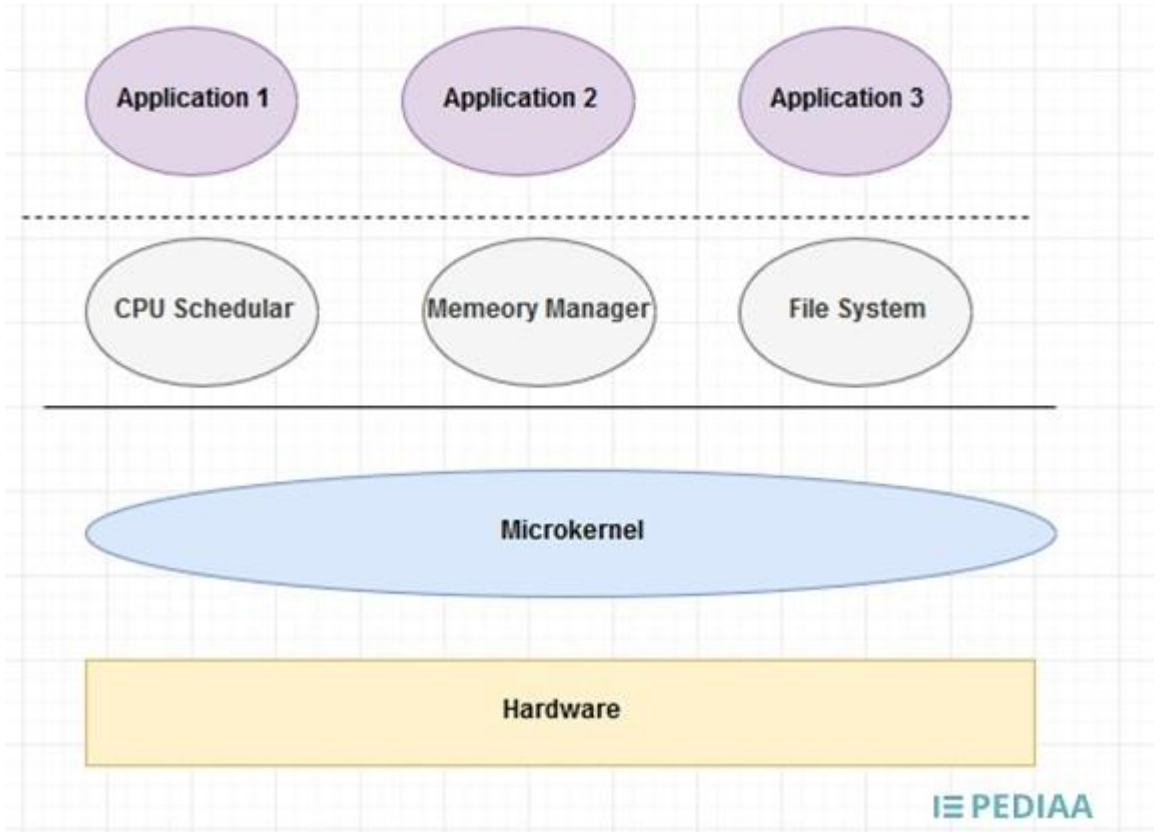
- **The main difference between microkernel and monolithic kernel is that the microkernel-based systems have OS services and kernel in separate address spaces while the monolithic kernel-based systems have OS services and kernel in the same address space.**
- Microkernel and monolithic kernel are two types of kernels. Kernel is the core of the operating system.
- Therefore, there is a special memory area to store the critical code of the kernel.
- The kernel is an important component as it maintains the proper functioning of the entire system.
- It performs hardware and process management, file handling and many other tasks

Microkernel vs Monolithic Kernel

- What is Microkernel
- Microkernel is a type of kernel that allows customization of the operating system. It runs on privileged mode and provides low-level address space management and Inter Process Communication (IPC). Moreover, OS services such as file system, virtual memory manager, and CPU scheduler are on top of the microkernel. Each service has its own address space to make them secure. Besides, the applications also have their own address spaces. Therefore, there is protection among applications, OS Services and kernel.

Microkernel vs Monolithic Kernel

- What is Microkernel



Microkernel vs Monolithic Kernel

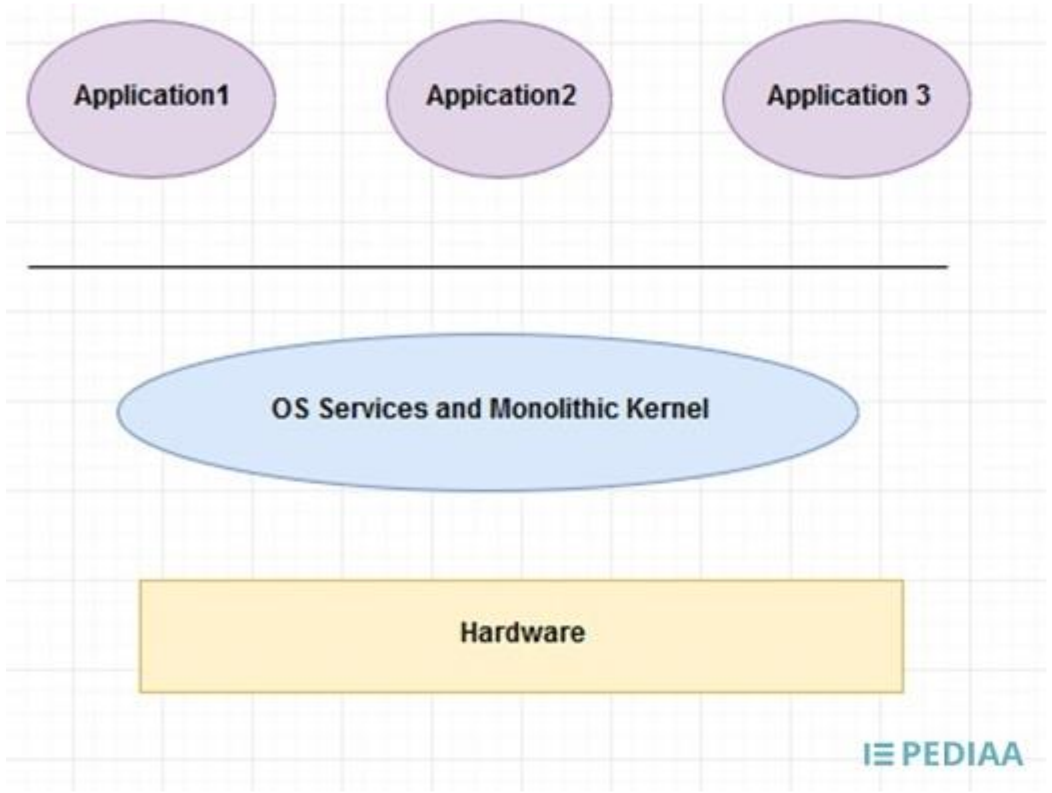
- What is Microkernel
- When the application requests the OS services for a service, the OS services communicate with each other to provide the required service to the application. Here, the Inter Process Communication (IPC) helps to establish this communication. Overall, microkernel-based OS provides a great level of extensibility. It is also possible to customize the services of the operating system depending on the requirements of the application.

Microkernel vs Monolithic Kernel

- What is Monolithic Kernel
- In monolithic kernel based systems, each application has its own address space. Therefore, each application is secure. Also, the kernel contains all the OS services. Therefore, the applications can request services from the kernel. Some OS services are file system, CPU Scheduler, network access, memory manager etc. However, the OS is in a separate address space. Therefore, it is secure from the normal applications and malfunctioned applications.

Microkernel vs Monolithic Kernel

- What is Monolithic Kernel
- .



Microkernel vs Monolithic Kernel

- Difference Between Microkernel and Monolithic Kernel
- A microkernel is a kernel type that provides mechanisms such as low-level address space management, thread management and interprocess communication to implement an operating system. In contrast, a monolithic kernel is a type of kernel in operating systems where the entire operating system works in the kernel space. These definitions explain the main difference between microkernel and monolithic kernel.
- .

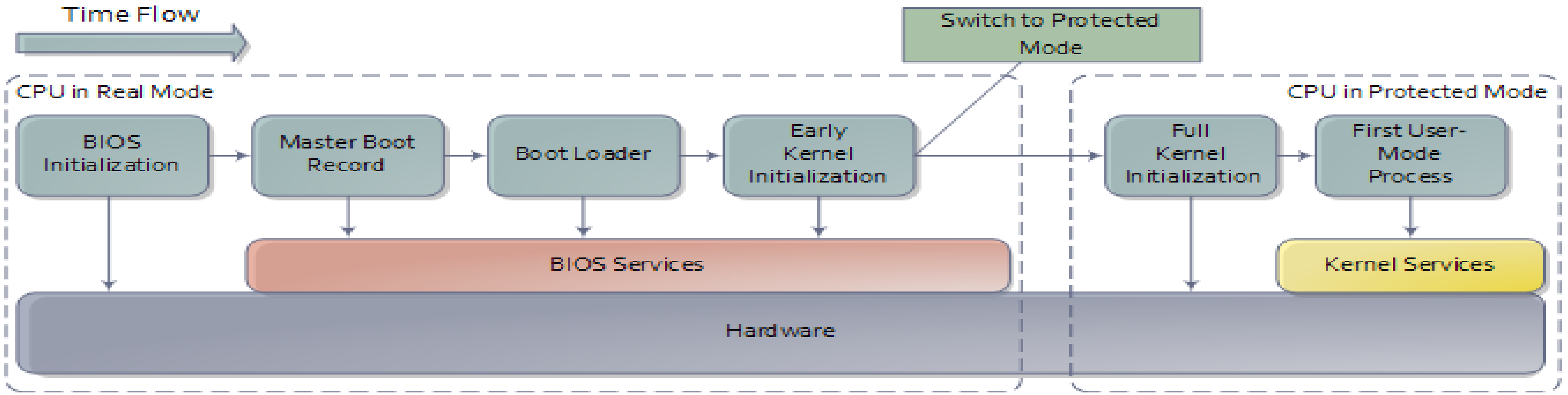
Microkernel vs Monolithic Kernel

Microkernel	Monolithic kernel
In microkernel user services and kernel, services are kept in separate address space.	In monolithic kernel, both user services and kernel services are kept in the same address space.
OS is complex to design.	OS is easy to design and implement.
Microkernel are smaller in size.	Monolithic kernel is larger than microkernel.
Easier to add new functionalities.	Difficult to add new functionalities.
To design a microkernel, more code is required.	Less code when compared to microkernel
Failure of one component does not effect the working of micro kernel.	Failure of one component in monolithic kernel leads to failure of entire system.
Execution speed is low.	Execution speed is high.
It is easy to extend Microkernel.	It is not easy to extend monolithic kernel.
Example : Mac OS X.	Example : Microsoft Windows 95.

Exam Questions

1) **Compare Monolihic kernel vs Microkernel**

Starting an Operating System(Booting)



- ✓ Power On Switch sends electricity to the motherboard on a wire called the *Voltage Good* line.
- ✓ If the power supply is good, then the BIOS (Basic Input/Output System) chip takes over.
- ✓ In Real Mode, CPU is only capable of using approximately 1 MB of memory built into the motherboard.
- ✓ The BIOS will do a Power-On Self Test (POST) to make sure that all hardware are working.

- ✓ BIOS will then look for a small sector at the very beginning of your primary hard disk called MBR.
- ✓ The MBR contains a list, or map, of all of the partitions on your computer‘s hard disk (or disks).
- ✓ After the MBR is found the Bootstrap Loader follows basic instructions for starting up the rest of the computer, including the operating system.
- ✓ In Early Kernel Initialization stage, a smaller core of the Kernel is activated.
- ✓ This core includes the device drivers needed to use computer‘s RAM chips.

BIOS

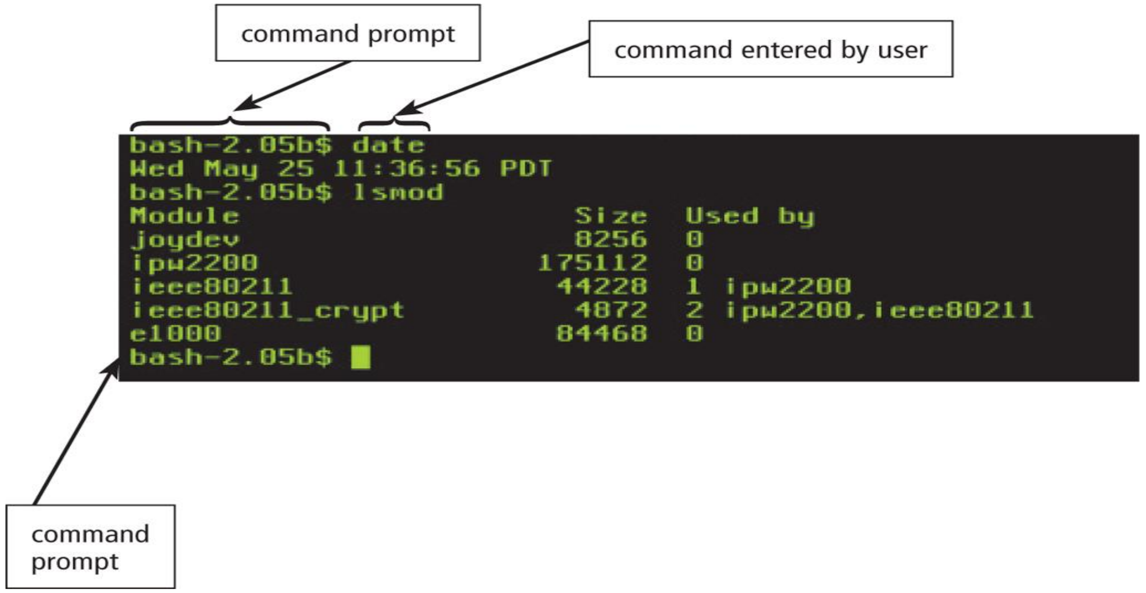
- BIOS firmware was stored in a ROM/EPROM (Erasable Programmable Read-Only Memory) chip known as **firmware** on the PC motherboard.
- BIOS can be accessed during the initial phases of the boot procedure by pressing del, F2 or F10.
- Finally, the firmware code cycles through all storage devices and looks for a boot-loader. (usually located in first sector of a disk which is 512 bytes)
- If the boot-loader is found, then the firmware hands over control of the computer to it.

UEFI

- UEFI stands for Unified Extensible Firmware Interface. It does the same job as a BIOS, but with one basic difference: it stores all data about initialization and startup in an .efi file, instead of storing it on the firmware.
- This .efi file is stored on a special partition called EFI System Partition (ESP) on the hard disk. This ESP partition also contains the bootloader.
- UEFI was designed to overcome many limitations of the old BIOS, including:
  - ➢ UEFI supports drive sizes upto 9 zettabytes, whereas BIOS only supports 2.2 terabytes.
  - ➢ UEFI provides faster boot time.
  - ➢ UEFI has discrete driver support, while BIOS has drive support stored in its ROM, so updating BIOS firmware is a bit difficult.
  - ➢ UEFI offers security like "Secure Boot", which prevents the computer from booting from unauthorized/unsigned applications. This helps in preventing rootkits.
  - ➢ UEFI runs in 32bit or 64bit mode, whereas BIOS runs in 16bit mode. So UEFI is able to provide a GUI (navigation with mouse) as opposed to BIOS which allows navigation only using the keyboard.

1. *Command-line interface*

- In a command-line interface, a user types commands represented by short keywords or abbreviations or presses special keys on the keyboard to enter data and instructions



```
bash-2.05b$ date
Wed May 25 11:36:56 PDT
bash-2.05b$ lsmod
Module                  Size  Used by
joydev                  8256  0
ipw2200               175112  0
ieee80211              44228  1 ipw2200
ieee80211_crypt         4872  2 ipw2200,ieee80211
e1000                  84468  0
bash-2.05b$
```

2. *Graphical User Interface*

- With a graphical user interface (GUI), you interact with menus and visual images

History of Operating System

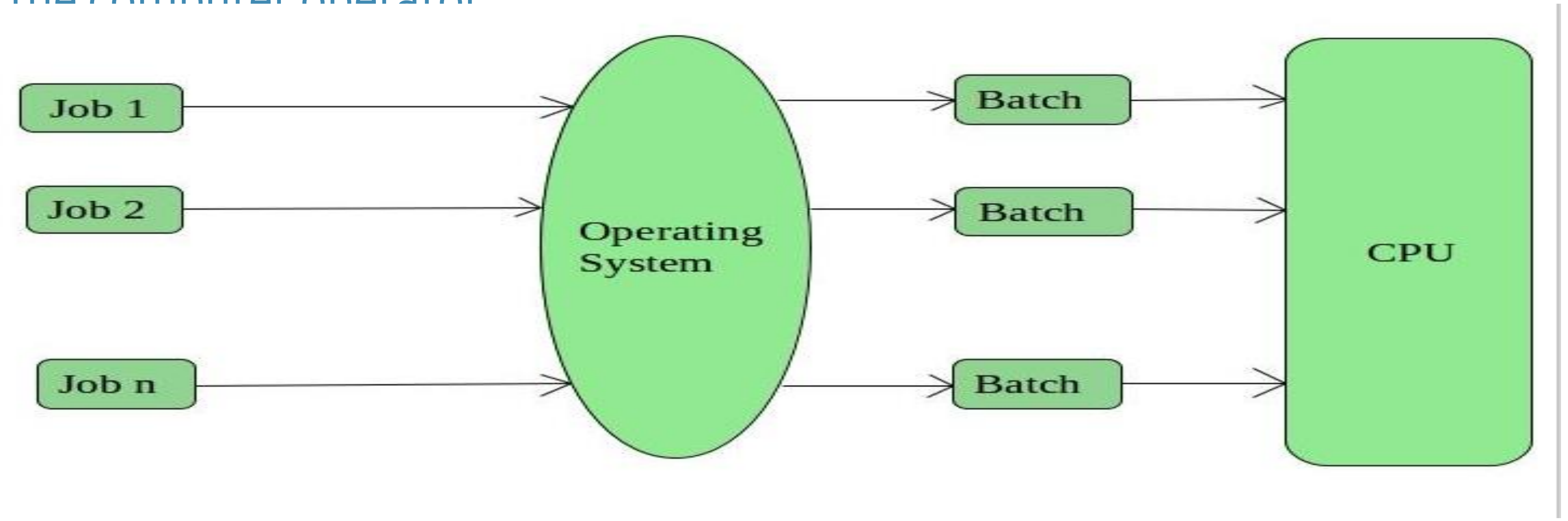
- **The First Generation (1940's to early 1950's)**
  - No Operating System
  - All programming was done in absolute machine language, often by wiring up plug-boards to control the machine's basic functions.
- **The Second Generation (1955-1965)**
  - First operating system was introduced in the early 1950's.It was called GMOS
  - Created by General Motors for IBM's machine the 701.
  - Single-stream batch processing systems
- **The Third Generation (1965-1980)**
  - Introduction of multiprogramming
  - Development of Minicomputer
- **The Fourth Generation (1980-Present Day)**
  - Development of PCs
  - Birth of Windows/MaC OS

Types of Operating Systems

1. Batch Operating System
2. Multiprogramming Operating System
3. Time-Sharing OS
4. Multiprocessing OS
5. Distributed OS
6. Network OS
7. Real Time OS
8. Embedded OS

1. *Batch Operating System*

- The users of this type of operating system does not interact with the computer directly.
- Each user prepares his job on an off-line device like punch cards and submits it to the computer operator
-



1. *Batch Operating System* *cont..*

Advantages of Batch Operating System:

- Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

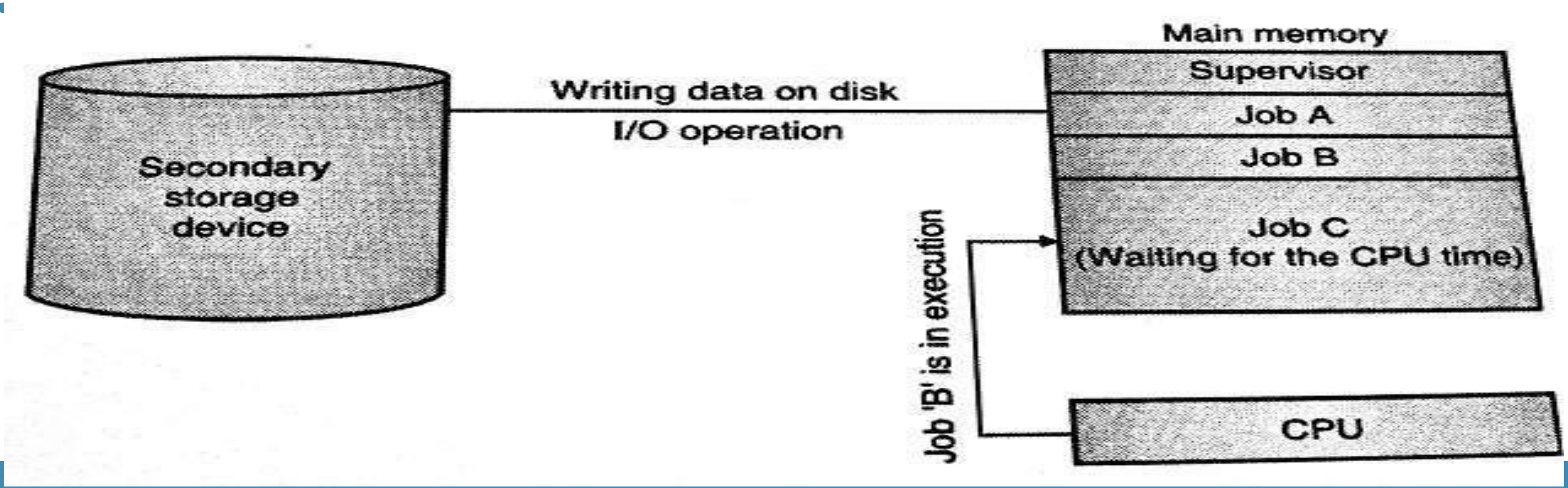
- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System:

IBM's MVS

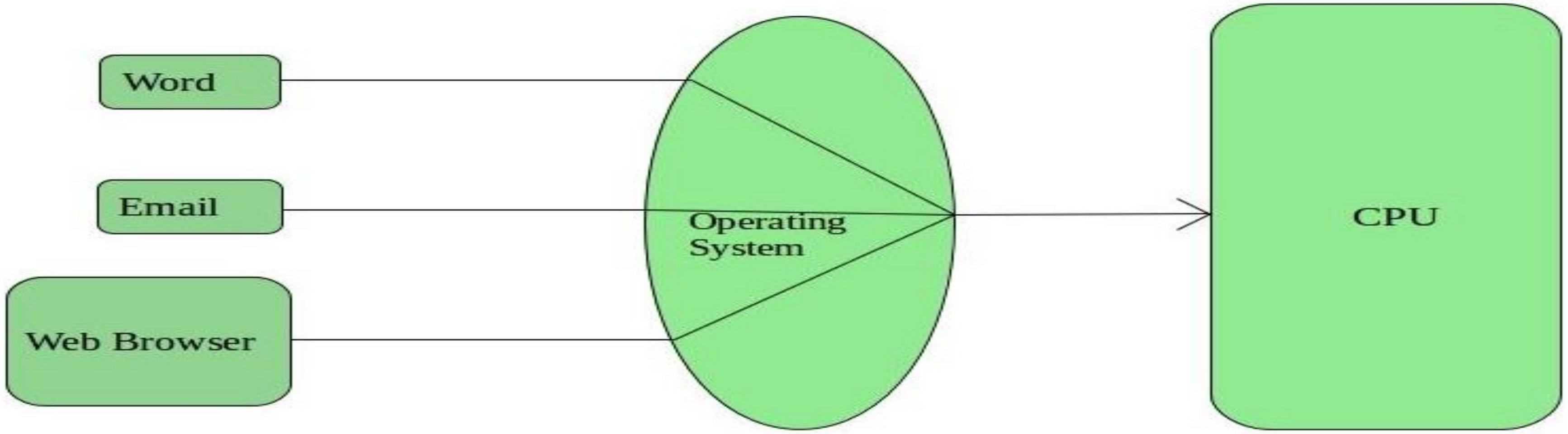
2. *Multiprogramming Operating System:*

- This type of OS is used to execute more than one jobs simultaneously by a single processor.
- It increases CPU utilization by organizing jobs so that the CPU always has one job to execute.
- [illegible]



3. *Time-Sharing Operating Systems*

- Each task is given some time to execute so that all the tasks work smoothly.
- These systems are also known as **Multi-tasking Systems.**
- The task can be from a single user or different users also.
- The ti
- After



3. *Time-Sharing Operating Systems cont..*

- **Advantages of Time-Sharing OS:**
  - ➢Each task gets an equal opportunity
  - ➢Fewer chances of duplication of software
  - ➢CPU idle time can be reduced
- **Disadvantages of Time-Sharing OS:**
  - ➢Reliability problem
  - ➢One must have to take care of the security and integrity of user programs and data
  - ➢Data communication problem
- **Examples of Time-Sharing Oss**

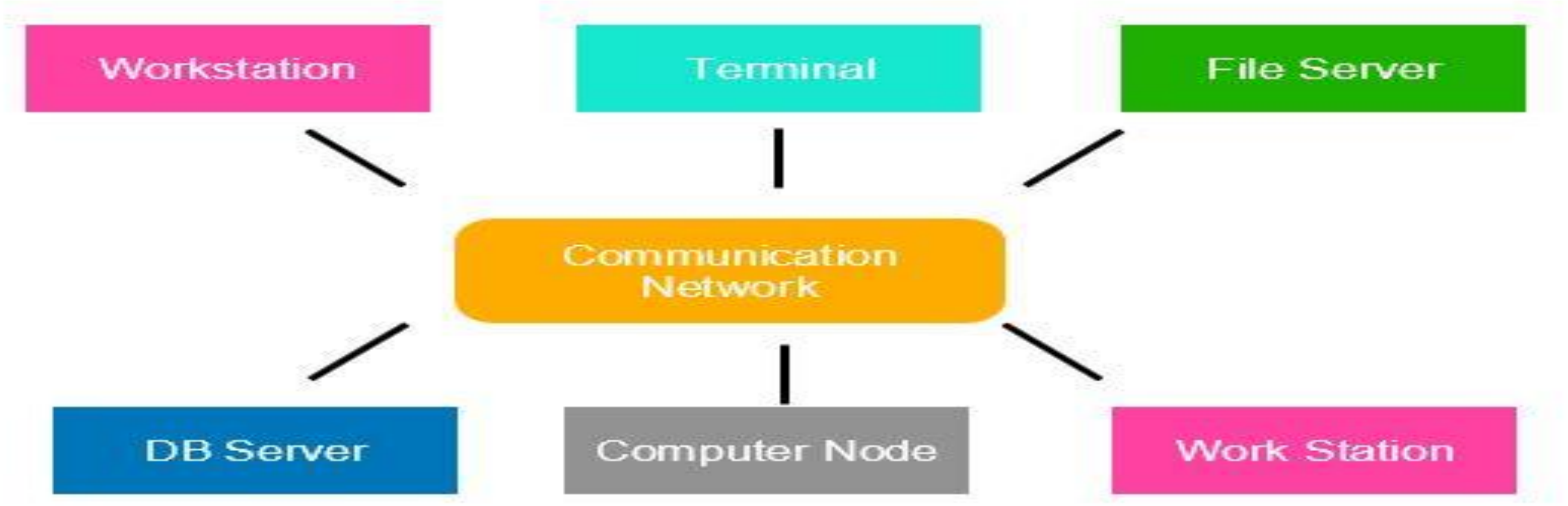
  Multics, Unix, etc.

4. ***Multiprocessor operating systems***

- Multiprocessor operating systems are also known as parallel OS or tightly coupled OS.
- Such operating systems have more than one processor in close communication that sharing the computer bus, the clock and sometimes memory and peripheral devices.
- It executes multiple jobs at the same time and makes the processing faster.
- It supports large physical address space and larger virtual address space.
- If one processor fails then other processor should retrieve the interrupted process state so execution of process can continue.
- Inter-processes communication mechanism is provided and implemented in hardware.

5. *Distributed Operating System*

- Various autonomous interconnected computers communicate with each other using a shared communication network.
- Independent systems possess their own memory unit and CPU.
- These are referred to as **loosely coupled systems**.
- Examples:- Locus, DYSEAC



6. ***Network Operating System***

- These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions.
- These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network.
- The " other" computers arc called client computers, and each computer that connects to a network server must be running client software designed to request a specific service.
- popularly known as **tightly coupled systems**.

6. Network Operating System

Advantages of Network Operating System:

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated into the system
- Server access is possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on a central location for most operations
- Maintenance and updates are required regularly

Examples of Network Operating System are:

Microsoft Windows Server 2003/2008/2012, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

7. *Real-Time Operating System*

- These types of OSs serve real-time systems.
- The time interval required to process and respond to inputs is very small.
- This time interval is called **response time**.
- **Real-time systems** are used when there are time requirements that are very strict like
  - ➢ missile systems,
  - ➢ air traffic control systems,
  - ➢ robots, etc.

8. Embaded Operating System

- An embedded operating system is one that is built into the circuitry of an electronic device.
- Embedded operating systems are now found in automobiles, bar-code scanners, cell phones, medical equipment, and personal digital assistants.
- The most popular embedded operating systems for consumer products, such as PDAs, include the following:
  - Windows XP Embedded
  - Windows CE .NET:- it supports wireless communications, multimedia and Web browsing. It also allows for the use of smaller versions of Microsoft Word, Excel, and Outlook.
  - Palm OS:- It is the standard operating system for Palm-brand PDAs as well as other proprietary handheld devices.
  - Symbian:- OS found in " smart" cell phones from Nokia and Sony Ericsson

Popular types of OS

- Desktop Class
  - ❖Windows
  - ❖OS X
  - ❖Unix/Linux
  - ❖Chrome OS
- Server Class
  - ❖Windows Server
  - ❖Mac OS X Server
  - ❖Unix/Linux
- Mobile Class
  - ❖Android
  - ❖iOS
  - ❖Windows Phone

Desktop Class Operating Systems:-

- **Platform:** the hardware required to run a particular operating system
  - Intel platform (IBM-compatible)
    - Windows
    - DOS
    - UNIX
    - Linux
  - Macintosh platform
    - Mac OS
  - iPad and iPhone platform
    - iOS

Ms-DOS

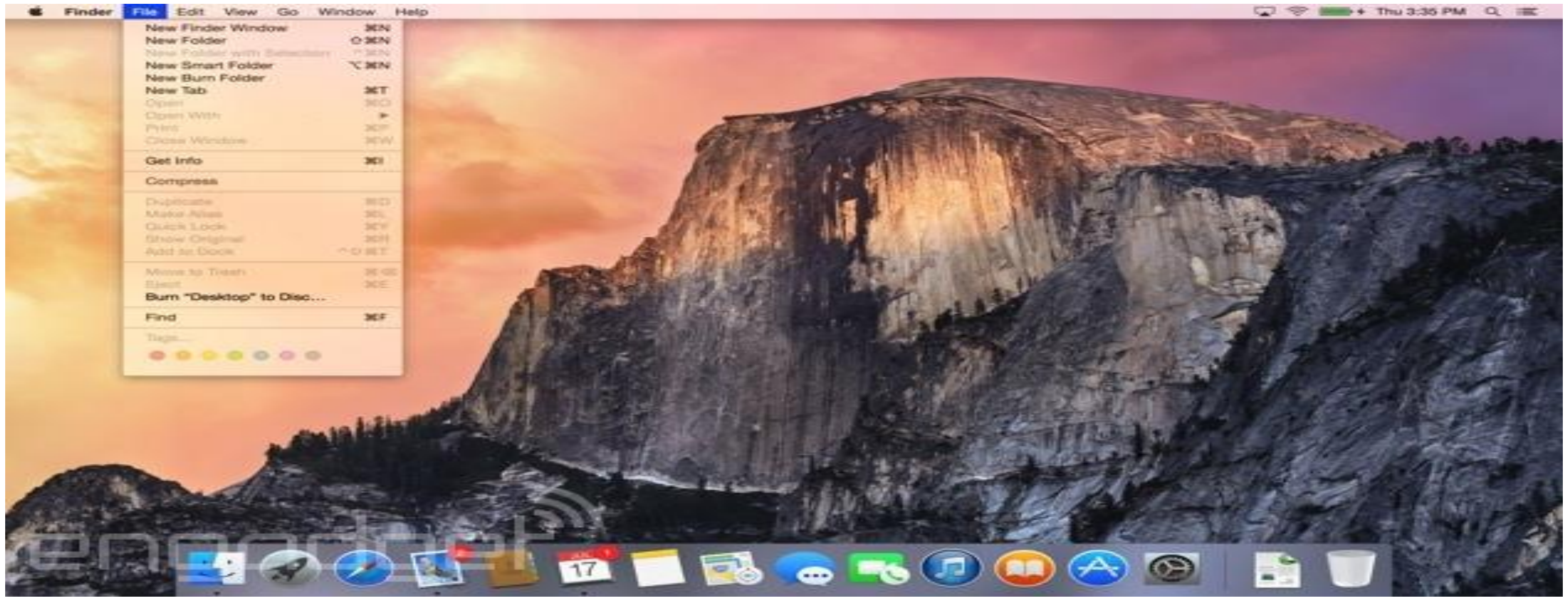
- Single User Single Tasking OS.
- It had no built-in support for networking, and users had to manually install drivers any time they added a new hardware component to their PC.
- DOS supports only 16-bit programs.
- Command line user interface.
- So, why is DOS still in use? Two reasons are its size and simplicity. It does not require much memory or storage space for the system, and it docs not require a powerful computer.

Microsoft Windows

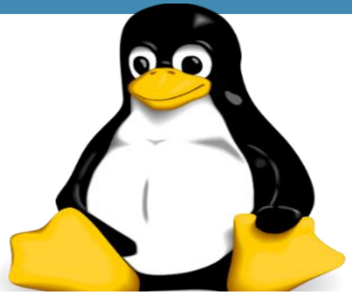
- The graphical Microsoft operating system designed for Intel-platform desktop and notebook computers.
- Best known, greatest selection of applications available.
- Current editions include Windows 7, 8, 8.1 and 10.

Mac OS

- User-friendly, runs on Mac hardware. Many applications available.
- Current editions include: Sierra, High Sierra, Mojave, Catalina & Big Sur—Version XI(Released in Nov 2020)



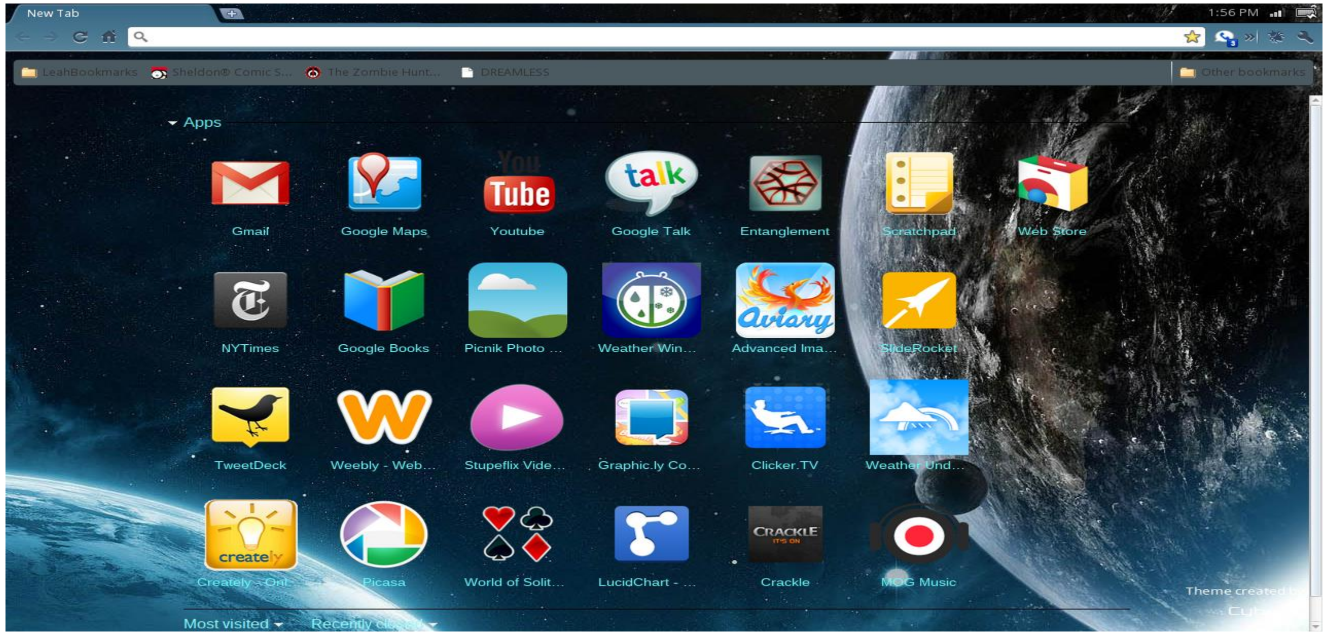
Linux



- **Linux:** An open-source, cross-platform operating system that runs on desktops, notebooks, tablets, and smartphones.
  - The name *Linux* is a combination *Linus* (the first name of the first developer) and *UNIX (another operating system*.
- Users are free to modify the code, improve it, and redistribute it,
- Developers are not allowed to charge money for the Linux kernel itself (the main part of the operating system), but they can charge money for **distributions** (**distros** for short).

Google Chrome OS

- **Chrome OS**. Is a popular thin client operating system.
- **Thin client** A computer with minimal hardware, designed for a specific task. For example, a thin web client is designed for using the Internet.

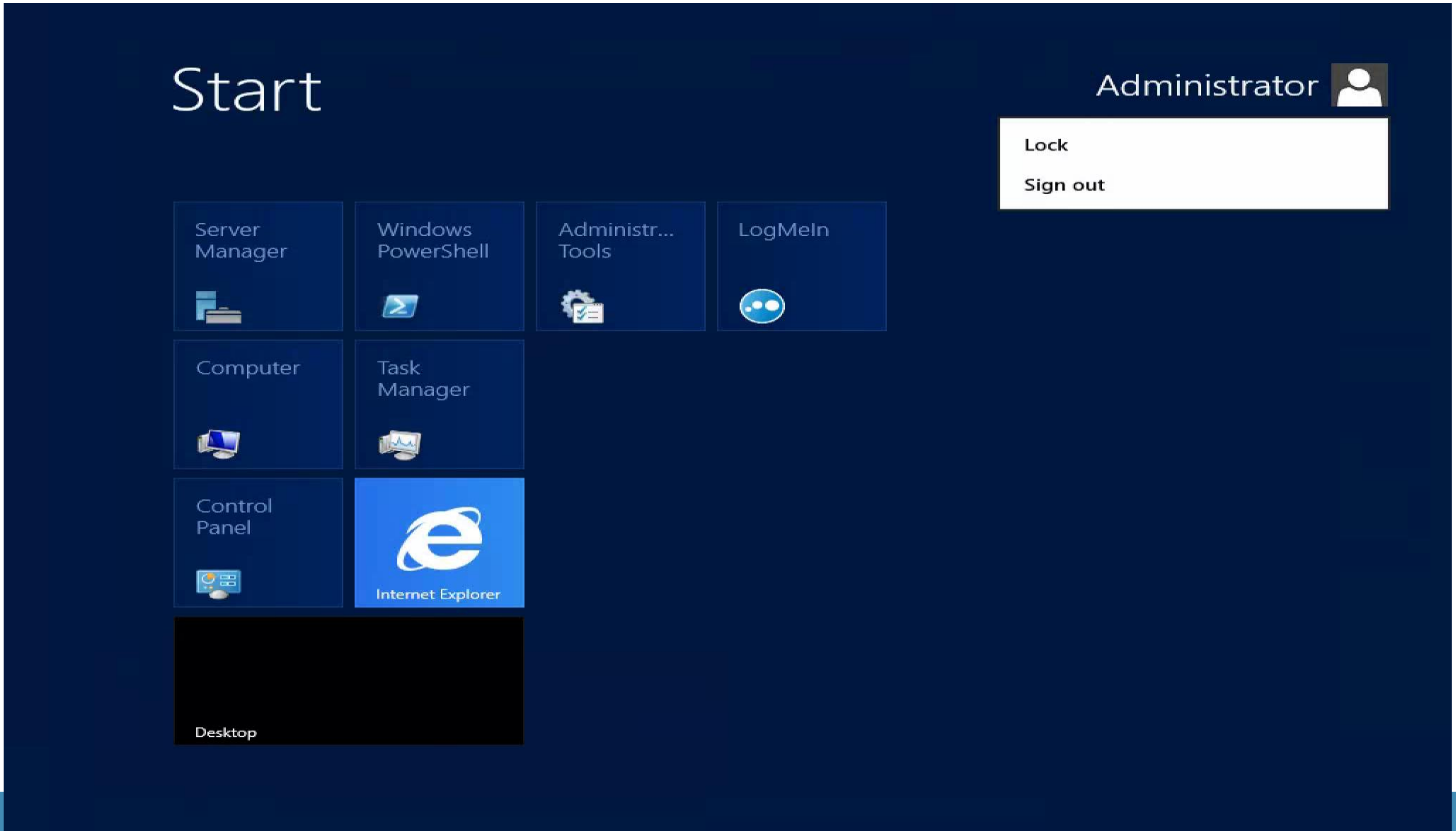


chromebook

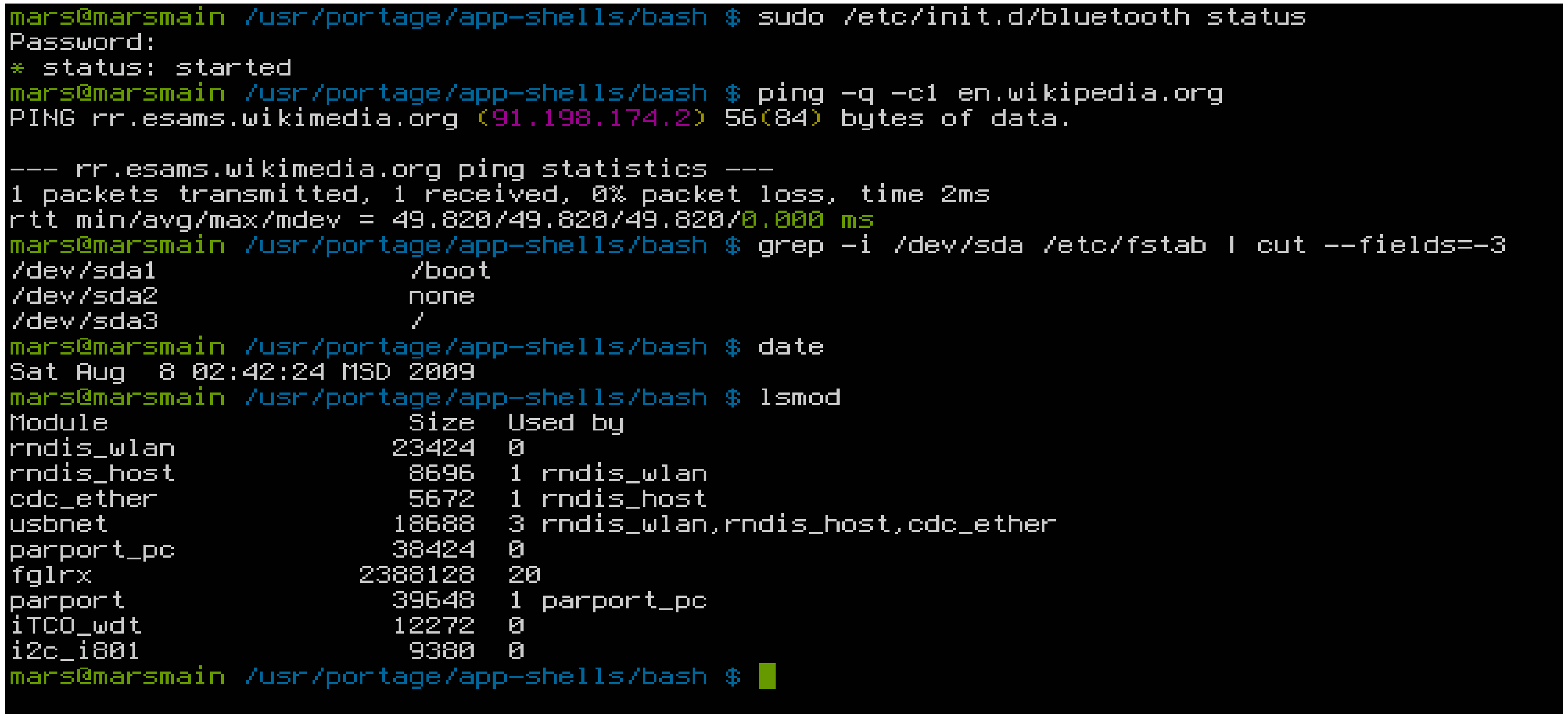
Server Operating Systems

- Windows Server
  - Familiar GUI interface for those experienced with Windows
- UNIX
  - Very mature server capabilities, time-tested, large user community, stable
- Linux
  - Free, customizable, many free services and utilities available

Windows Server



UNIX



```
mars@marsmain /usr/portage/app-shells/bash $ sudo /etc/init.d/bluetooth status
Password:
 * status: started
mars@marsmain /usr/portage/app-shells/bash $ ping -q -c1 en.wikipedia.org
PING rr.esams.wikimedia.org (91.198.174.2) 56(84) bytes of data.

--- rr.esams.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 49.820/49.820/49.820/0.000 ms
mars@marsmain /usr/portage/app-shells/bash $ grep -i /dev/sda /etc/fstab | cut --fields=-3
/dev/sda1               /boot
/dev/sda2               none
/dev/sda3               /
mars@marsmain /usr/portage/app-shells/bash $ date
Sat Aug  8 02:42:24 MSD 2009
mars@marsmain /usr/portage/app-shells/bash $ lsmod
Module                  Size  Used by
rndis_wlan             23424  0
rndis_host              8696  1 rndis_wlan
cdc_ether               5672  1 rndis_host
usbnet                 18688  3 rndis_wlan,rndis_host,cdc_ether
parport_pc             38424  0
fglrx                2388128  20
parport                39648  1 parport_pc
iTCO_wdt               12272  0
i2c_i801                9380  0
mars@marsmain /usr/portage/app-shells/bash $ 
```

Tablet and Phone Operating Systems

- **System-on-chip (SoC):** An operating system that comes preinstalled on a chip on a portable device such as a smartphone.
- Popular SoC operating systems:
  - iOS: for iPad, iPhone
  - Android: for a variety of tablets and phones
- Downloadable applications (apps) from an App store, for example:
  - Apple App Store
  - Google Play Store

ANDROID APP ON Google play

Available on the App Store

iOS on the iPhone and iPad

- The Apple-created operating system for Apple tablets and phones.
- The current stable version, iOS 14, was released to the public on September 16, 2020

"Remind me about this when I get home"
tap to edit
OK, I'll remind you when you arrive at home:
Reminder
11 Hot Beach Escapes - Jetsetter
Arriving: Home
Remove
Search
SIRI SUGGESTIONS
Show More
Nisha
David
Erin
Steve
Twitter
KAYAK
Instagram
OpenTable
NEARBY
Food
Drink
Shopping
Fun
NEWS
NASA Curiosity Rover Captures Sunset On Mars
Sunsets on Earth can be some of the most beautiful and breathtaking natural phenomena to watch and photograph. But...
huffingtonpost.com - today
Healthy diet may improve memory, says study - CNN.com
"Show me photos from Utah last August"
tap to edit

Android

- Android, a popular OS for smartphones and tablets, is based on Linux Kernel.
  - Developed by Google
- Current versions include:
  - Android 8 Oreo
  - Android 9 Pie
  - Android 10
  - Android 11 (released on Sep, 2020)

Advantage of Linux Operating System

1. Open Source

As it is open-source, its source code is easily available.
Anyone having programming knowledge can customize the operating system.
One can contribute, modify, distribute, and enhance the code for any purpose.

2. Security

The Linux security feature is the main reason that it is the most favourable option for developers.
It is not completely safe, but it is less vulnerable than others.
Each application needs to authorize by the admin user.
Linux systems do not require any antivirus program.

3. Free

Certainly, the biggest advantage of the Linux system is that it is free to use.
We can easily download it, and there is no need to buy the license for it.
It is distributed under GPL (General Public License).
Comparatively, we have to pay a huge amount for the license of the other OS

Advantage of Linux Operating System

4. Lightweight

The requirements for running Linux are much less than other operating system

In Linux, the memory footprint and disk space are also lower.

Generally, most of the Linux distributions required as little as 128MB of RAM around the same amount for disk space.

5. Stability

Linux is more stable than other operating systems.

Linux does not require to reboot the system to maintain performance levels.

It rarely hangs up or slow down. It has big up-times.

Advantage of Linux Operating System

6. Performance

Linux system provides high performance over different networks.

It is capable of handling a large number of users simultaneously.

7. Flexibility

Linux operating system is very flexible.

It can be used for desktop applications, embedded systems, and server applications too.

It also provides various restriction options for specific computers.

We can install only necessary components for a system.

8. Software Updates

In Linux, the software updates are in user control.

We can select the required updates.

There a large number of system updates are available.

These updates are much faster than other operating systems.

So, the system updates can be installed easily without facing any issue.

Advantage of Linux Operating System

9. Distributions/ Distros

There are many Linux distributions available in the market.

It provides various options and flavors of Linux to the users.

We can choose any distros according to our needs.

Some popular distros are **Ubuntu, Fedora, Debian, Linux Mint, Arch Linux,**

For the beginners, Ubuntu and Linux Mint would be useful.

Debian and Fedora would be good choices for proficient programmers.

10. Live CD/USB

Almost all Linux distributions have **a Live CD/USB** option.

It allows us to try or run the Linux operating system without installing it.

11. Graphical User Interface

Linux is a command-line based OS but it provides an interactive user interface like Windows.

Advantage of Linux Operating System

12. Suitable for programmers

It supports almost all of the most used programming languages such as C/C++, Java, Python, Ruby, and more.

Further, it offers a vast range of useful applications for development.

The programmers prefer the Linux terminal over the Windows command line.

The package manager on Linux system helps programmers to understand how things are done.

Bash scripting is also a functional feature for the programmers.

It also provides support for SSH, which helps in managing the servers quickly.

13. Community Support

Linux provides large community support.

We can find support from various sources.

There are many forums available on the web to assist users.

Further, developers from the various open source communities are ready to help us.

Advantage of Linux Operating System

14. Privacy

Linux always takes care of user privacy as it never takes much private data from the user. Comparatively, other operating systems ask for the user's private data.

15. Networking

Linux facilitates with powerful support for networking. The client-server systems can be easily set to a Linux system. It provides various command-line tools such as ssh, ip, mail, telnet, and more for connectivity with the other systems and servers. Tasks such as network backup are much faster than others.

16. Compatibility

Linux is compatible with a large number of file formats as it supports almost all file formats.

17. Installation

Linux installation process takes less time than other operating systems such as Windows. Further, its installation process is much easy as it requires less user input. It does not require much more system configuration even it can be easily installed on old machines having less configuration.

Advantage of Linux Operating System

18. Multiple Desktop Support

Linux system provides multiple desktop environment support for its enhanced use. The desktop environment option can be selected during installation. We can select any desktop environment such as **GNOME (GNU Network Object Model Environment)** or **KDE (K Desktop Environment)** as both have their specific environment.

19. Multitasking

It is a multitasking operating system as it can run multiple tasks simultaneously without affecting the system speed.

20. Heavily Documented for beginners

There are many command-line options that provide documentation on commands, libraries, standards such as manual pages and info pages. Also, there are plenty of documents available on the internet in different formats, such as Linux tutorials, Linux documentation project, Serverfault, and more. To help the beginners, several communities are available such as **Ask Ubuntu**, Reddit, and **StackOverflow.**

System Calls

- System Calls in Operating System
- 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.

System Calls

- How are system calls made?
- When a computer software needs to access the operating system's kernel, it makes a system call. The system call uses an API to expose the operating system's services to user programs. It is the only method to access the kernel system. All programs or processes that require resources for execution must use system calls, as they serve as an interface between the operating system and user programs.
- Below are some examples of how a system call varies from a user function.

1. A system call function may create and use kernel processes to execute the asynchronous processing.
2. A system call has greater authority than a standard subroutine. A system call with kernel-mode privilege executes in the kernel protection domain.
3. System calls are not permitted to use shared libraries or any symbols that are not present in the kernel protection domain.
4. The code and data for system calls are stored in global kernel memory.

System Calls

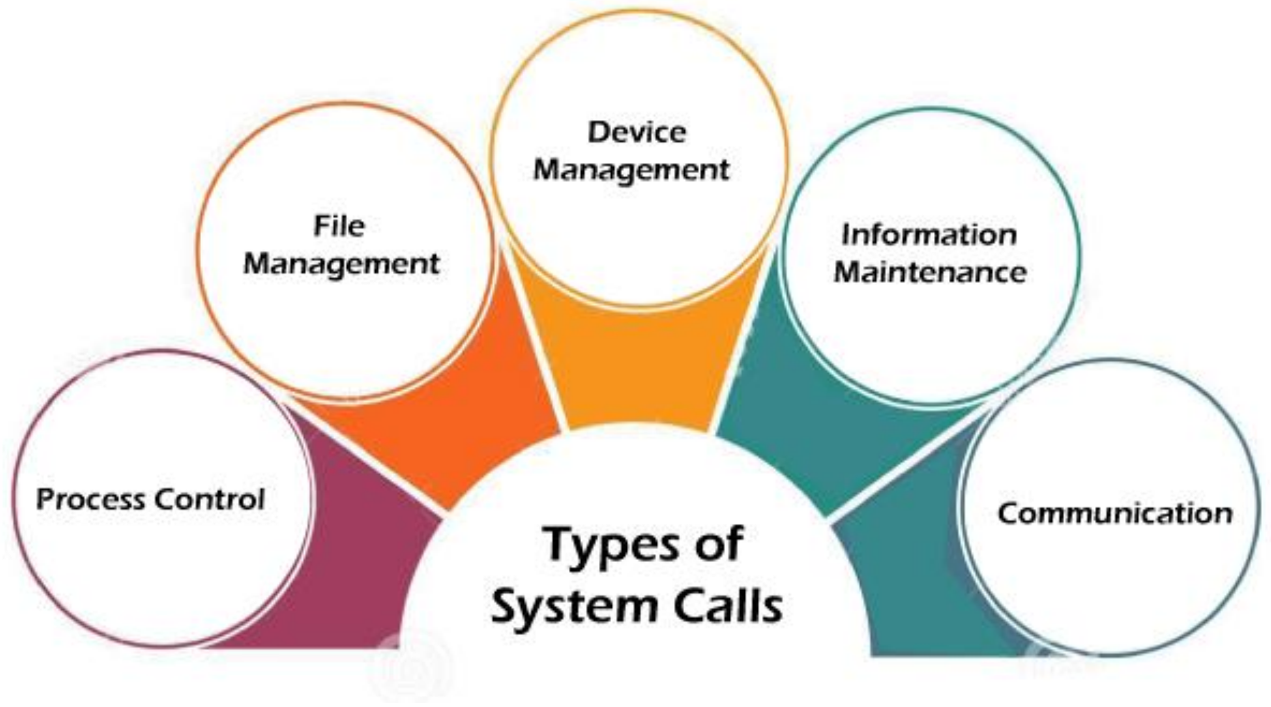
- Why do you need system calls in Operating System?
- There are various situations where you must require system calls in the operating system. Following of the situations are as follows:
- It is must require when a file system wants to create or delete a file.
- Network connections require the system 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 Calls

- How System Calls Work
- The Applications run in an area of memory known as user space. A system call connects to the operating system's kernel, which executes in kernel space. When an application creates a system call, it must first obtain permission from the kernel. It achieves this using an interrupt request, which pauses the current process and transfers control to the kernel.
- If the request is permitted, the kernel performs the requested action, like creating or deleting a file. As input, the application receives the kernel's output. The application resumes the procedure after the input is received. When the operation is finished, the kernel returns the results to the application and then moves data from kernel space to user space in memory.
- A simple system call may take few nanoseconds to provide the result, like retrieving the system date and time. A more complicated system call, such as connecting to a network device, may take a few seconds. Most operating systems launch a distinct kernel thread for each system call to avoid bottlenecks. Modern operating systems are multi-threaded, which means they can handle various system calls at the same time.

System Calls

- Types of System Calls



System Calls

1. **Process Control**
2. **File Management**
3. **Device Management**
4. **Information Maintenance**
5. **Communication**

System Calls

1. **Process Control**

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

2. File Management

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc

3. Device Management

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

4. Information Maintenance

Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

5. Communication

Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

System Calls

Process	Windows	Unix

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		Fork()
		Exit()
		Wait()

File Manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		Open()
		Read()
		Write()
		Close()

Device Management	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		Ioctl()
		Read()
		Write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		Getpid()
		Alarm()
		Sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		Pipe()
		Shmget()
		Mmap()

Protection	SetFileSecurity()
	InitializeSecurityDescriptor()
	SetSecurityDescriptorgroup()
		Chmod()
		Umask()
		Chown()

System Calls

- open()
- The **open()** system call allows you to access a file on a file system. It allocates resources to the file and provides a handle that the process may refer to. Many processes can open a file at once or by a single process only. It's all based on the file system and structure.
- read()
- It is used to obtain data from a file on the file system. It accepts three arguments in general:
- A file descriptor.
- A buffer to store read data.
- The number of bytes to read from the file.
- The file descriptor of the file to be read could be used to identify it and open it using **open()** before reading.

System Calls

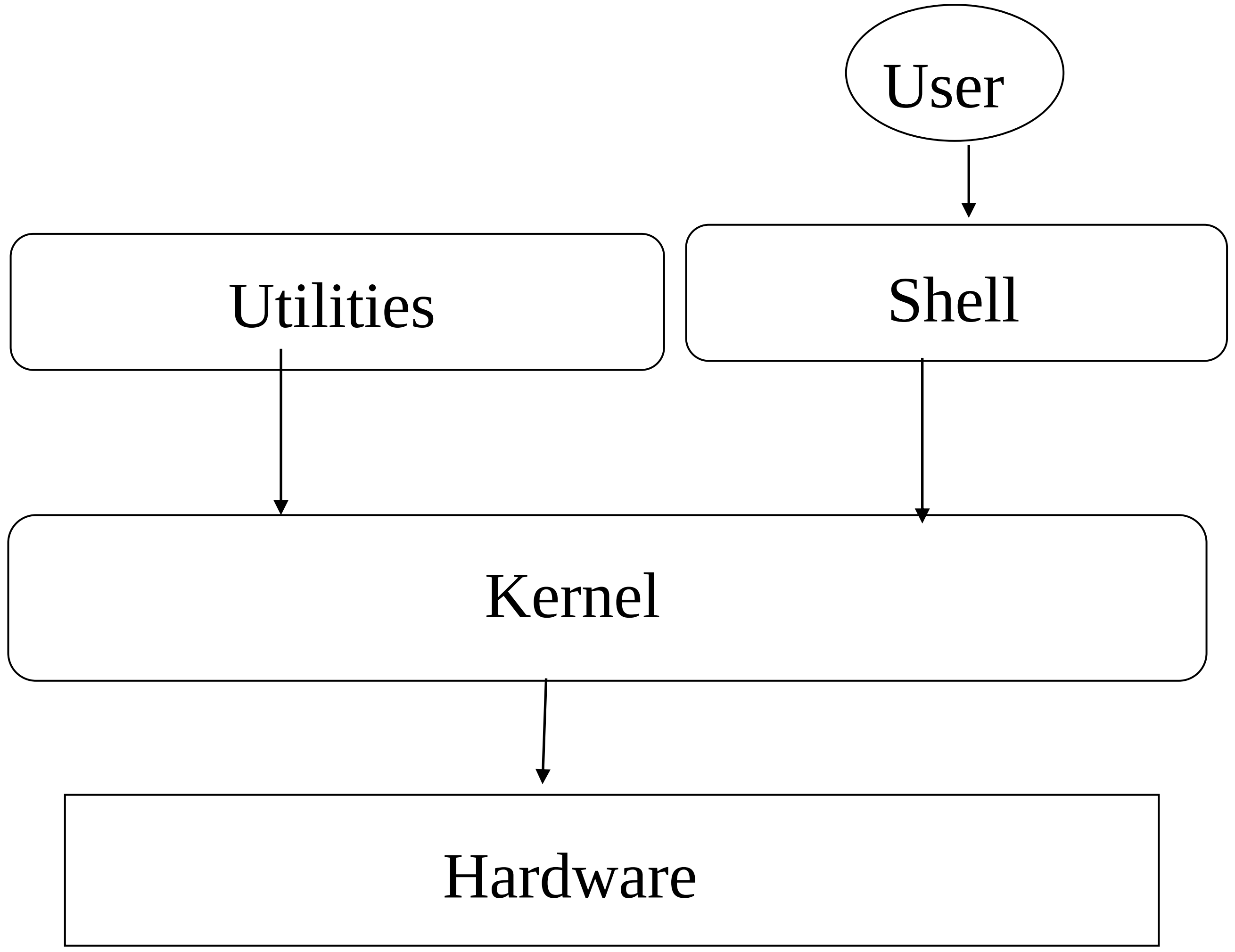
- wait()
- In some systems, a process may have to wait for another process to complete its execution before proceeding. When a parent process makes a child process, the parent process execution is suspended until the child process is finished. The **wait()** system call is used to suspend the parent process. Once the child process has completed its execution, control is returned to the parent process.
- write()
- It is used to write data from a user buffer to a device like a file. This system call is one way for a program to generate data. It takes three arguments in general:
- A file descriptor.
- A pointer to the buffer in which data is saved.
- The number of bytes to be written from the buffer.

System Calls

- fork() : Processes generate clones of themselves using the **fork()** system call. It is one of the most common ways to create processes in operating systems. When a parent process spawns a child process, execution of the parent process is interrupted until the child process completes. Once the child process has completed its execution, control is returned to the parent process.
- close() : It is used to end file system access. When this system call is invoked, it signifies that the program no longer requires the file, and the buffers are flushed, the file information is altered, and the file resources are de-allocated as a result.
- exec() : When an executable file replaces an earlier executable file in an already executing process, this system function is invoked. As a new process is not built, the old process identification stays, but the new process replaces data, stack, data, head, etc.
- exit() : The **exit()** is a system call that is used to end program execution. This call indicates that the thread execution is complete, which is especially useful in multi-threaded environments. The operating system reclaims resources spent by the process following the use of the **exit()** system function.

UNIX Shell and Utilities

- ➢The shell used to be in the kernel but now is a utility outside of it.
- ➢Easy to change/debug.
- ➢Many of them (sh, bsh, csh, ksh, tcsh, wsh, bash)
- ➢Possible to switch between them (chsh)



A very simplified Shell

```
#define TRUE 1

while (TRUE) {                                    /* repeat forever */
    type_prompt( );                               /* display prompt on the screen */
    read_command(command, parameters);            /* read input from terminal */

    if (fork( ) != 0) {                           /* fork off child process */
        /* Parent code. */
        waitpid(−1, &status, 0);                  /* wait for child to exit */
    } else {
        /* Child code. */
        execve(command, parameters, 0);           /* execute command */
    }
}
```