Chapter 1: Introduction

(OS Structure, Modes and Services)

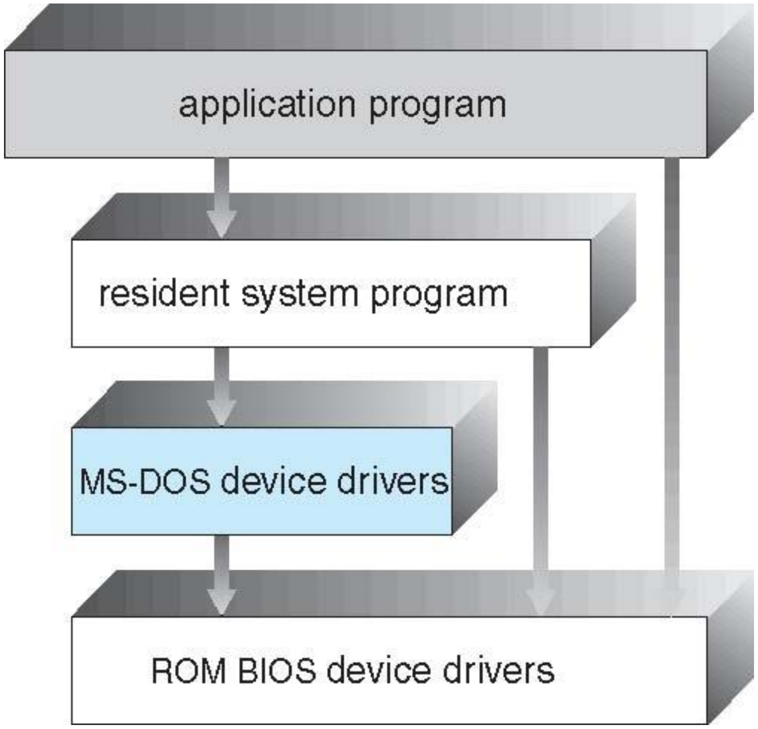
By: Navjot Kaur

Operating System Structure

- Various ways to structure a system

Simple Structure -- MS-DOS

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

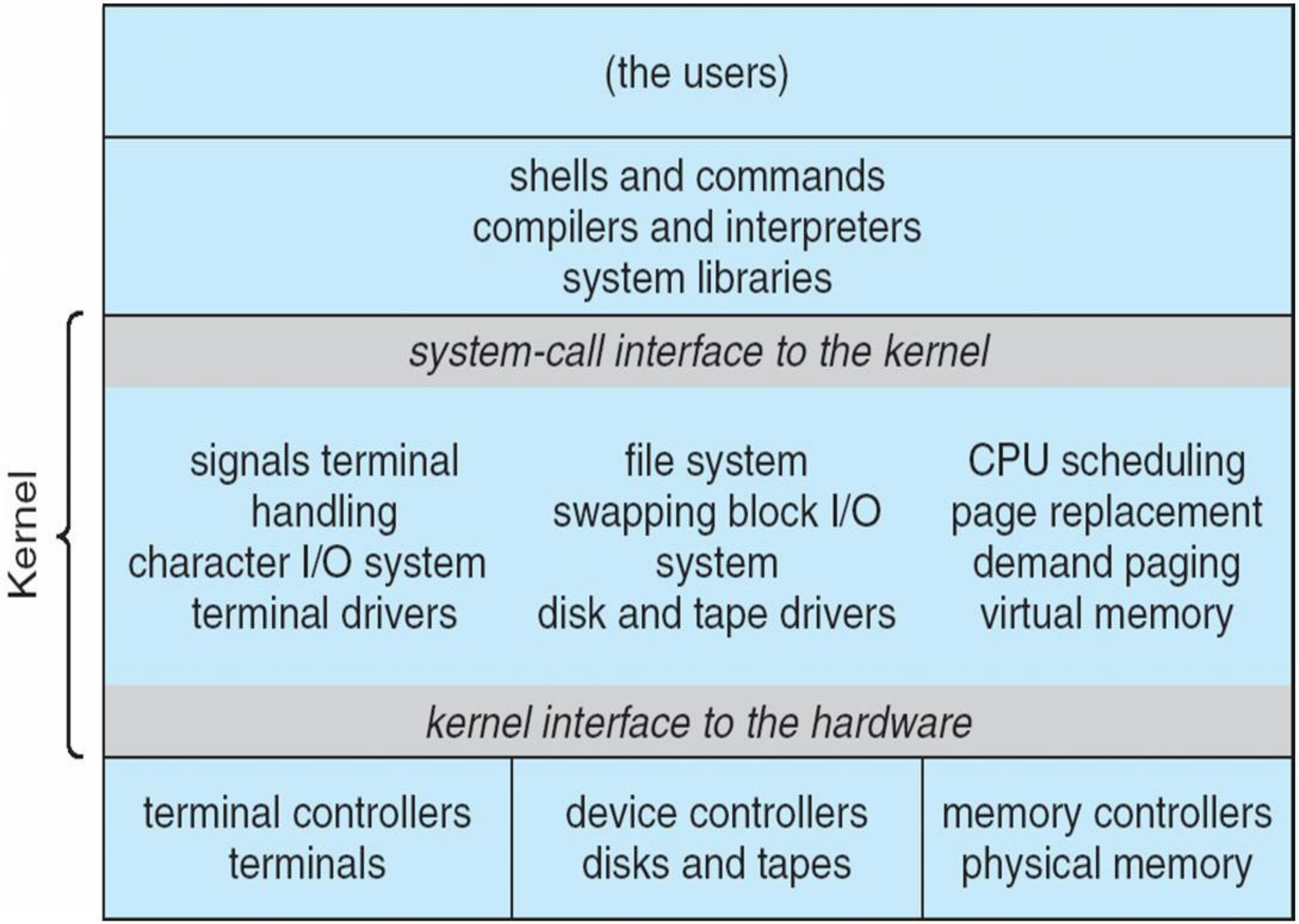


Non Simple Structure -- UNIX

The UNIX OS consists of two parts:

- System programs
- The kernel
  - Consists of everything below the system-call interface and above the physical hardware
  - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

Traditional UNIX System Structure



(the users)

shells and commands
compilers and interpreters
system libraries

system-call interface to the kernel

signals terminal
handling
character I/O system
terminal drivers

file system
swapping block I/O
system
disk and tape drivers

CPU scheduling
page replacement
demand paging
virtual memory

kernel interface to the hardware

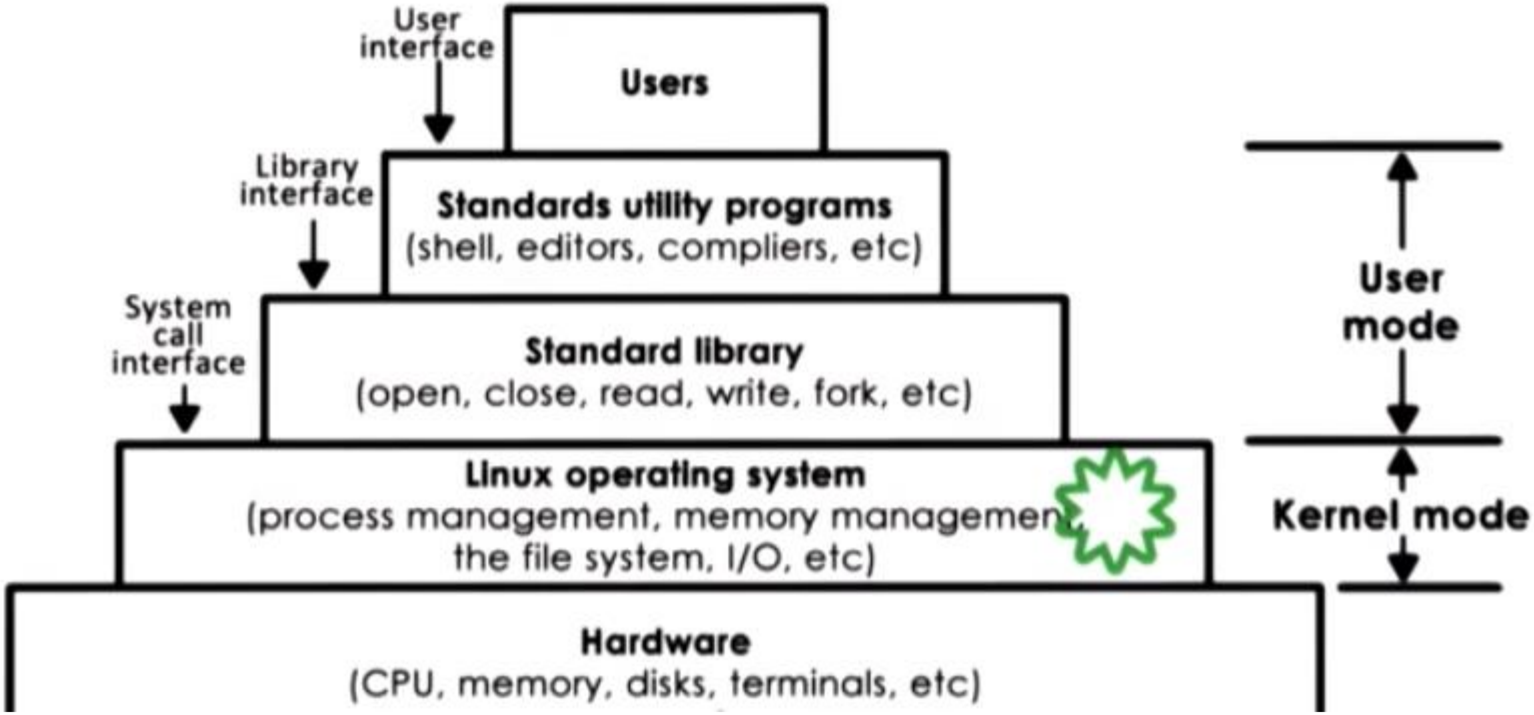
terminal controllers
terminals

device controllers
disks and tapes

memory controllers
physical memory

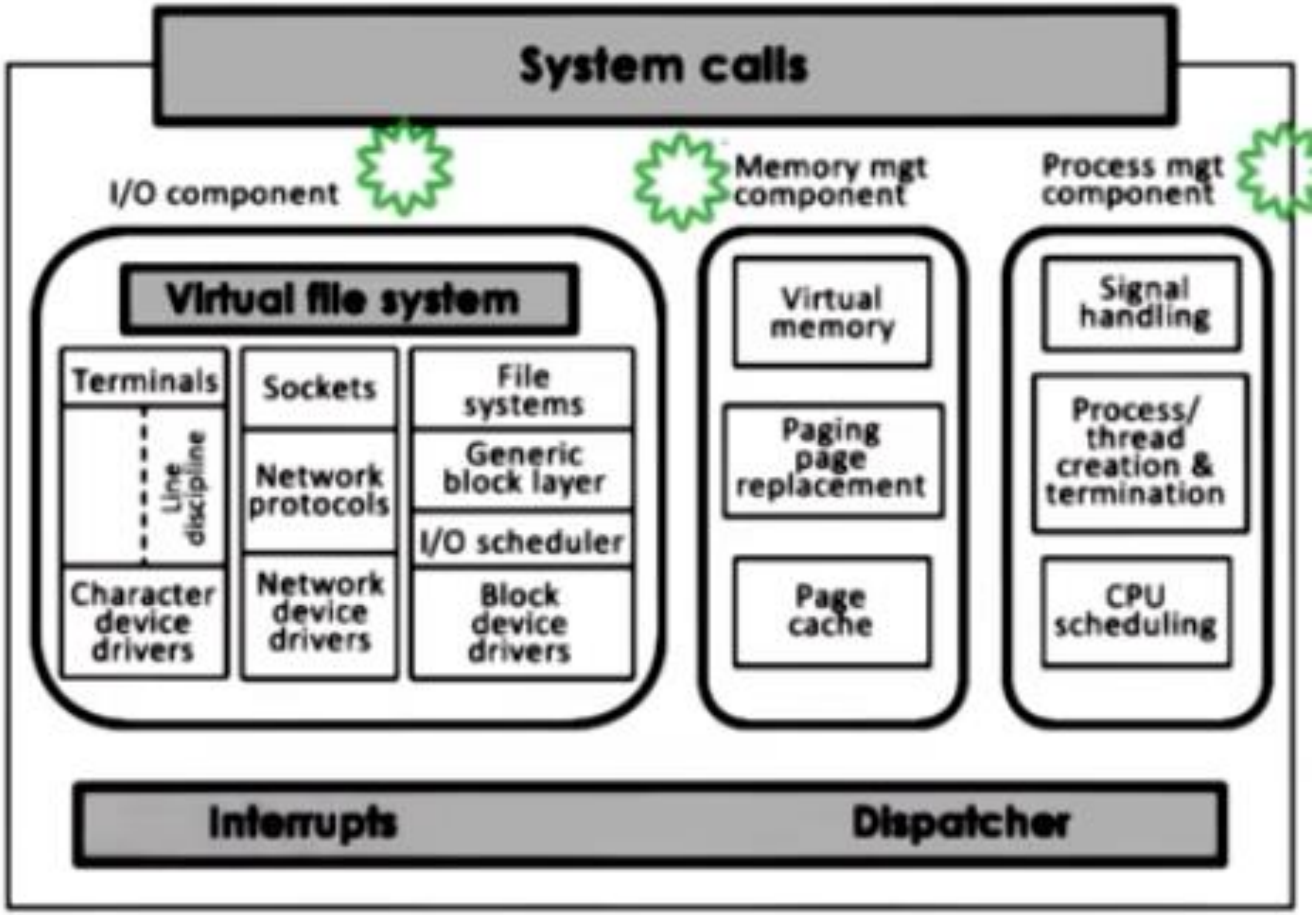
Traditional LINUX System Structure

Linux Architecture



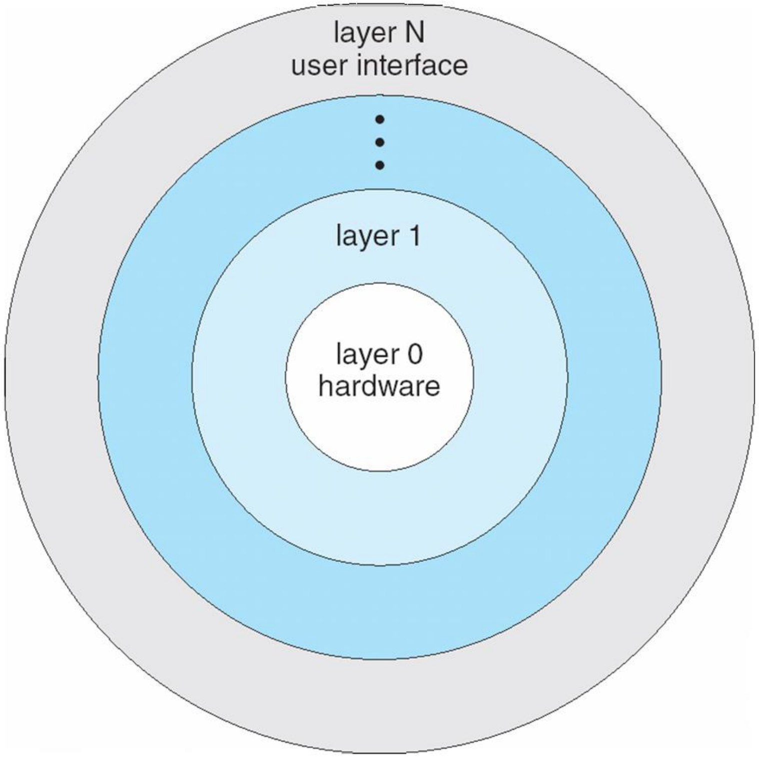
User interface
Users
Library interface
Standards utility programs
(shell, editors, compliers, etc)
System call interface
Standard library
(open, close, read, write, fork, etc)
Linux operating system
(process management, memory management, the file system, I/O, etc)
Hardware
(CPU, memory, disks, terminals, etc)
User mode
Kernel mode

Kernel has many inbuilt modules



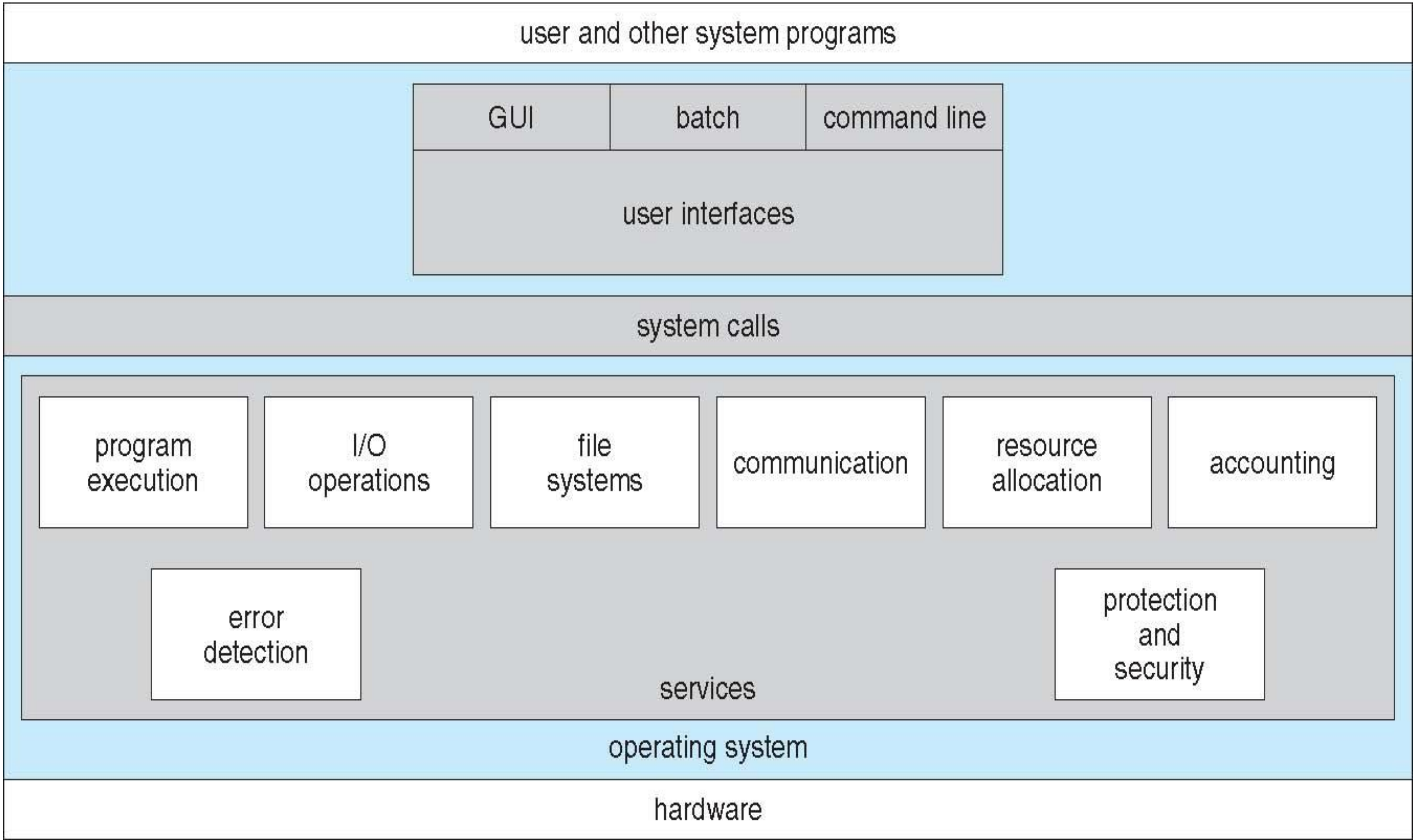
Layered Approach

- The operating system 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



Operating System Services

- An operating system provides an **environment for the programs to run.**
- It provides certain services to programs



Operating System Services

- Operating-system services provides functions that are helpful to the user:
  - **User interface** - Almost all operating systems have a user interface (UI)
    - Varies between Command-Line (CLI), Graphics User Interface (GUI).
  - **Program execution** - The system must be able **to load a program into memory and to run that program**, end execution, either normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device.
  - **File-system manipulation** - read and write files and directories, create and delete them, search them, list file Information, permission management.

Operating System Services

- **Communications** – Processes may exchange information, on the same computer or between computers over a network
  - **Communications may be via shared memory or through message passing** (packets moved by the OS)

- **Error detection – OS needs to be constantly aware of possible errors**
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error**, OS should take the appropriate action** to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system.

- **Resource allocation –** OS must ensure allocation of resources to all programs running.
  - **Many types of resources** - such as **CPU cycle time, main memory, and file storage**, **I/O devices**

Operating System Services

- **Accounting -** To keep track of which users use how much and what kinds of **computer resources.**

- **Protection and security -** 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
  - **Protection** involves **ensuring that all access to system resources is controlled**
  - **Security** of the system from outsiders requires **user authentication**, extends to defending external I/O devices from invalid access attempts
  - If a system is to be protected and secure, precautions must be instituted throughout it.
  - **A chain is only as strong as its weakest link.**

Kernel

- A kernel is a central component of an operating system.
- It acts as an interface between the user applications and the hardware.
- The sole aim of the kernel is to manage the communication between the software (user level applications) and the hardware (CPU, disk memory etc).
- The main tasks of the kernel are :
  - Process management
  - Device management
  - Memory management
  - Interrupt handling
  - I/O communication
  - File system...etc..

Kernel

- A kernel is the lowest level of software that interfaces with the hardware in your computer.

- It is responsible for interfacing all applications that are running in “user mode” down to the physical hardware, and allowing processes, to get information from each other using inter-process communication (IPC).

Kernel Types

kernels fall into one of three types:

- Monolithic
- Microkernel
- Hybrid

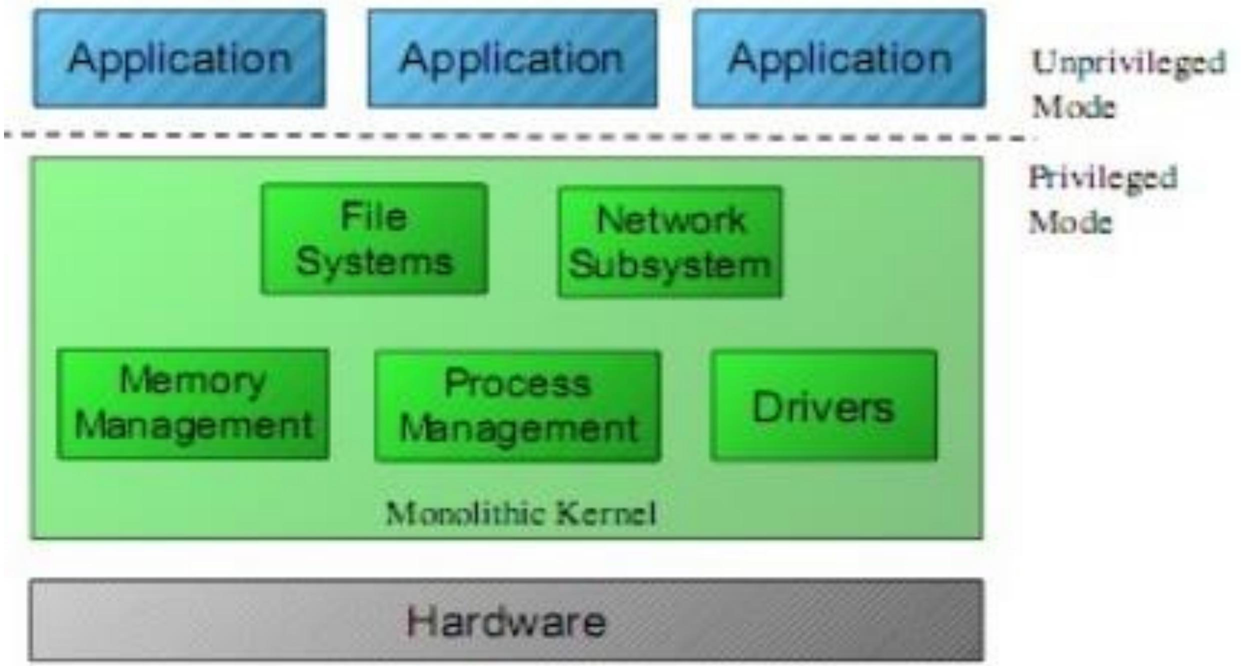
Monolithic Kernel

- A **monolithic kernel** is an operating system architecture where the entire operating system (which includes the device drivers, file system, and the application IPC etc.) is working in kernel space, in supervisor mode.

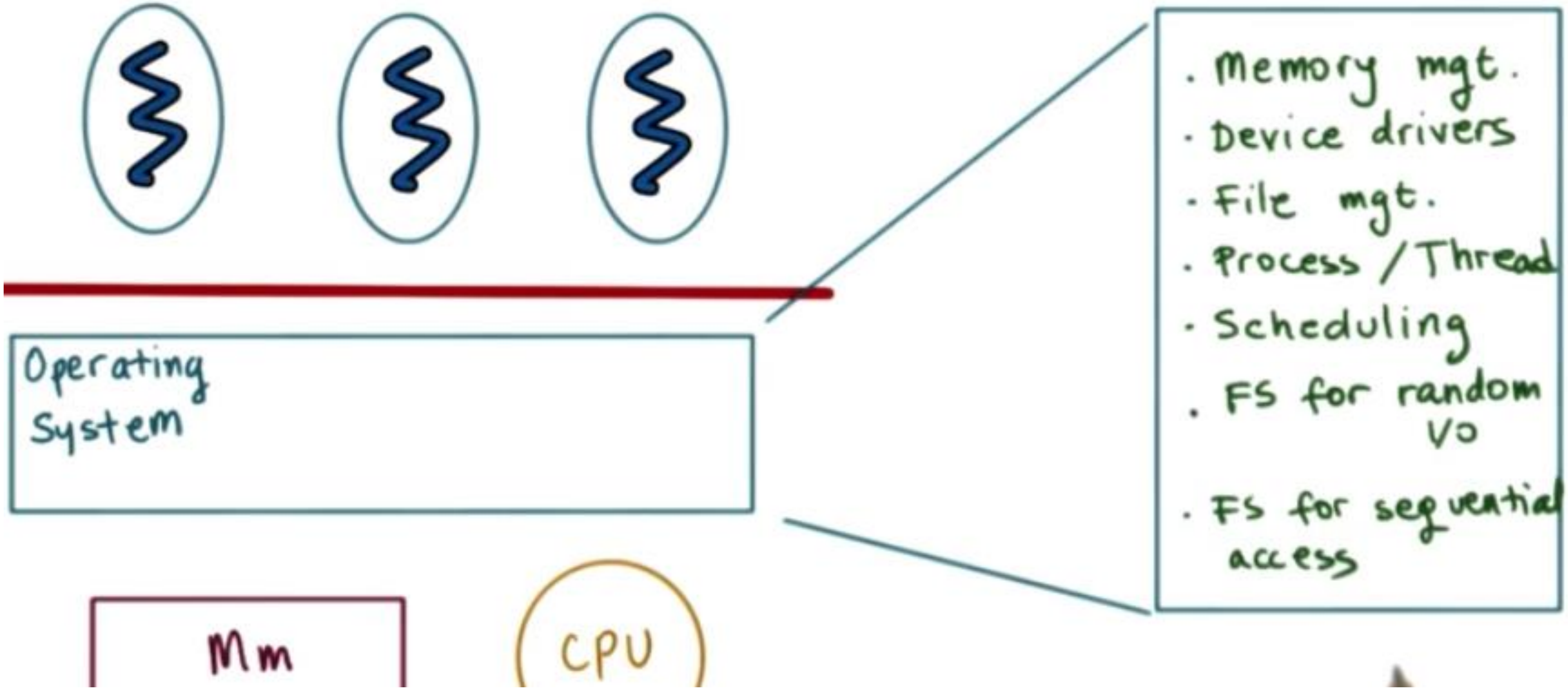
- Monolithic kernels are able to dynamically load (and unload) executable modules at runtime.

- Examples of operating systems that use a monolithic kernel are - Linux, Solaris, OS-9, DOS, Microsoft Windows (95,98,Me) etc.

Monolithic Kernel



Monolithic Kernel



- Memory mgt.
- Device drivers
- File mgt.
- Process / Thread
- Scheduling
- FS for random I/O
- FS for sequential access

Monolithic Kernel

Pros:

- More direct access to hardware for programs
- Easier for processes to communicate between each other
- If your device is supported, it should work with no additional installations
- Processes react faster because there isn’t a queue for processor time.

Cons:

- Large install footprint
- Large memory is needed
- Less secure because everything runs in supervisor mode

MicroKernel

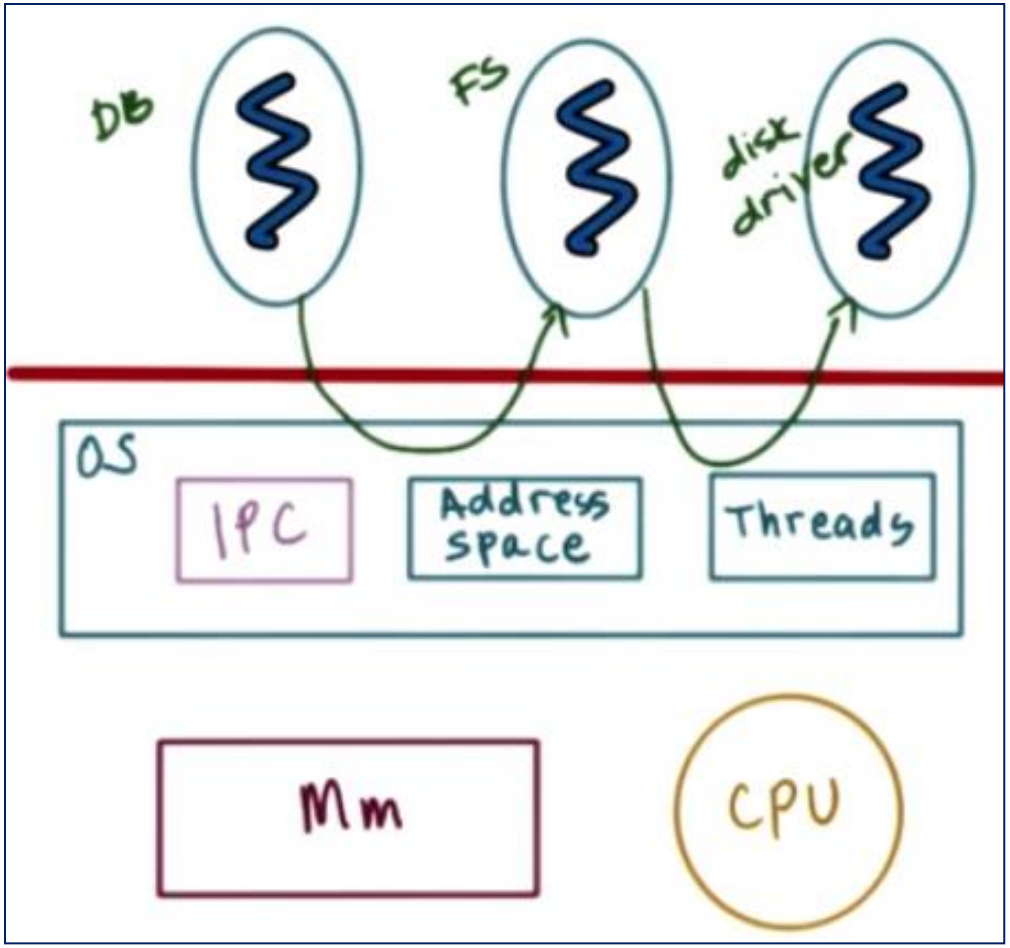
- In a Microkernel architecture, the core functionality is isolated from system services and device drivers.

- This architecture allows some basic services like device driver management, file system etc. to run in user space.

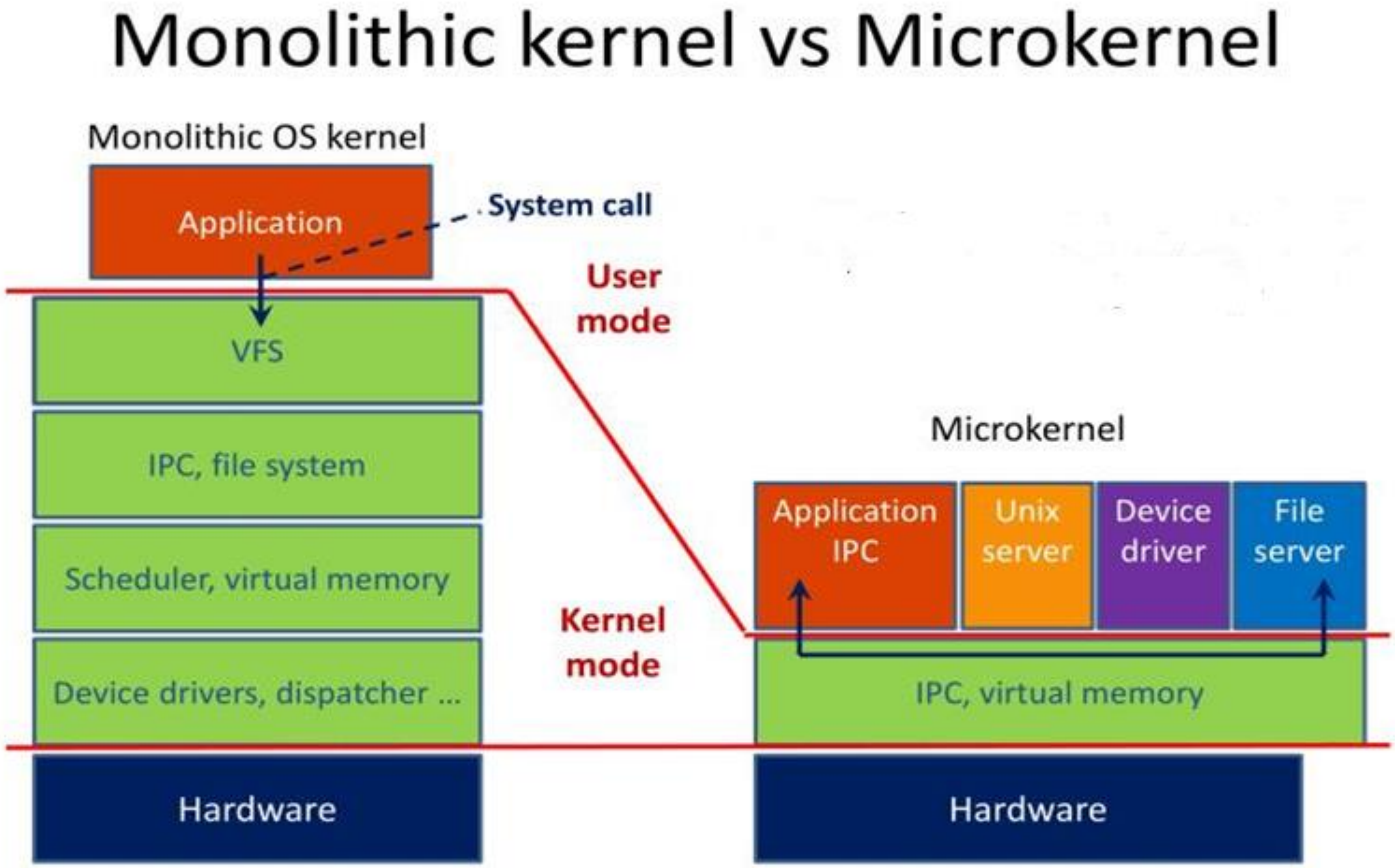
- This reduces the kernel code size and also increases the security and stability of OS as we have minimum code running in kernel.

- Examples of operating systems that use a microkernel are - QNX, Integrity, PikeOS, Symbian, L4Linux, Singularity, K42, Mac OS X, HURD, Minix, and Coyotos.

MicroKernel



Types of Kernel



Monolithic kernel vs Microkernel

MicroKernel

Pros

- Portability
- Small install footprint
- Small memory
- Security

Cons

- Hardware is more abstracted through drivers
- Hardware may react slower because drivers are in user mode
- Processes have to wait in a queue to get information
- Processes can’t get access to other processes without waiting

HybridKernel

- Hybrid kernels have the ability to pick and choose what they want to run in user mode and what they want to run in supervisor mode.
- Device drivers and file system I/O will be run in user mode while IPC and server calls will be kept in the supervisor mode.
- This require more work of the hardware manufacturer because all of the driver responsibility is up to them.

Hybrid Kernel

Pros

- Developer can pick and choose what runs in user mode and what runs in supervisor mode
- Smaller install than monolithic kernel
- More flexible than other models

Cons

- Processes have to wait in a queue to get information
- Processes can’t get access to other processes without waiting
- Device drivers need to be managed by user

Interrupts

- An interrupt is a signal from a device attached to a computer or from a program within the computer that causes the main program that operates the computer (the operating system) to stop and figure out what to do next.

- Interrupts can be of following type:

  - Generated by Hardware (Hardware Interrupt)

  - Generated by Software (Software Interrupt)

Q3. In which type of operating system users do not interact directly with the computer system?

a) Multiprogramming operating systems

b) Multiprocessing operating systems

c) Batch operating systems

d) Distributed operating systems

Ans 3) c

Q4. What is the objective of multiprogramming operating systems?

a) Maximize CPU utilization

b) Switch the CPU among processes

c) Achieve multitasking

d) None of the above

Ans4. a)

Q5. Who signalled for the occurrence of an event either from the hardware or the software?

a) Bootstrap program

b) Interrupt

c) Disk Controller

d) CPU

Ans5: b

Q6. In which type of I/O interrupts the control return to the user program after the completion of I/O operation?

a) Synchronous I/O interrupts

b) Asynchronous I/O interrupts

c) System Call

d) Hardware

Ans 6) a

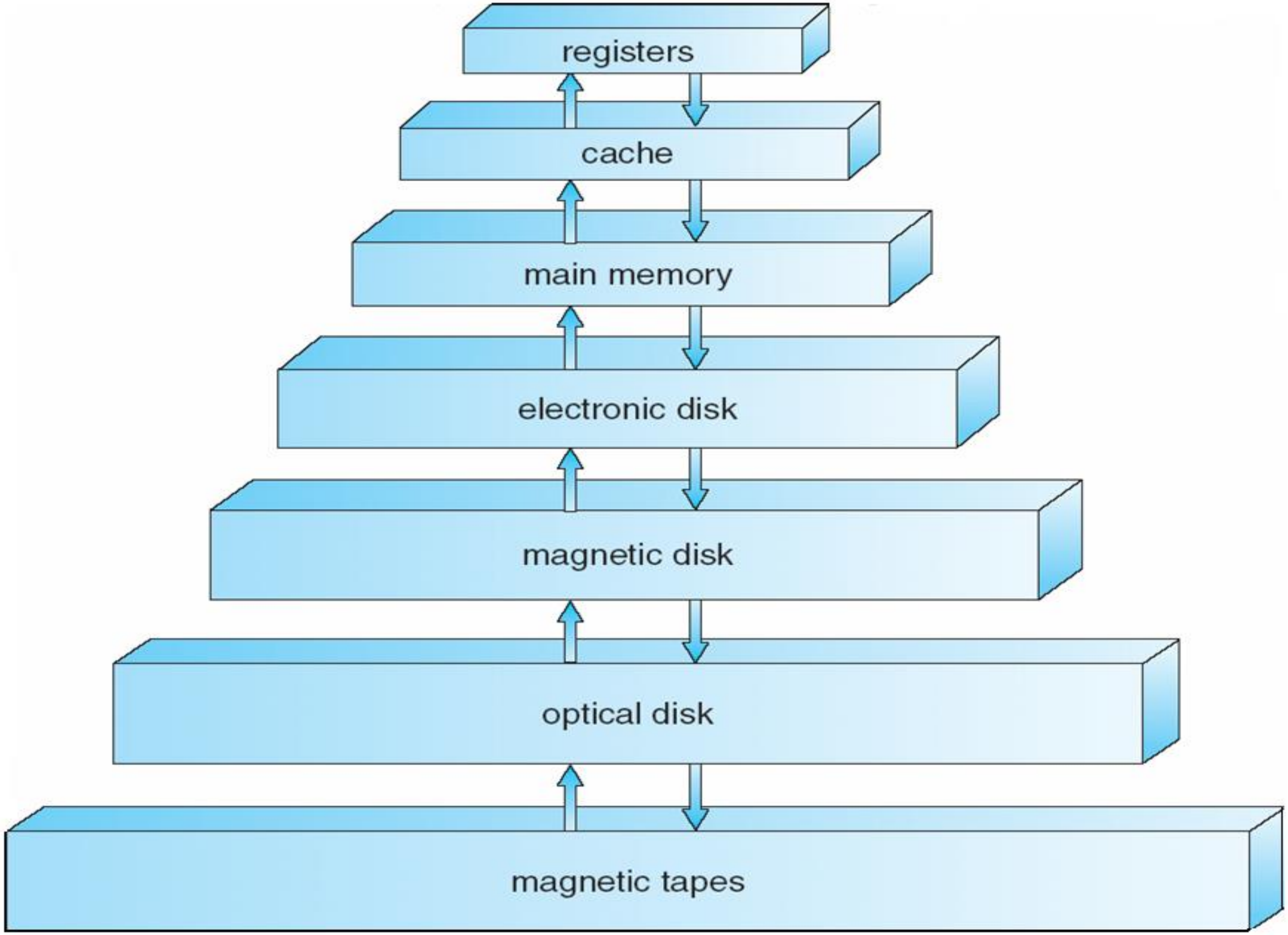
Q7: The device-status table contains

a) each I/O device type
b) each I/O device state
c) each I/O device address
d) all of the above

(d)

Ans 7 (d)

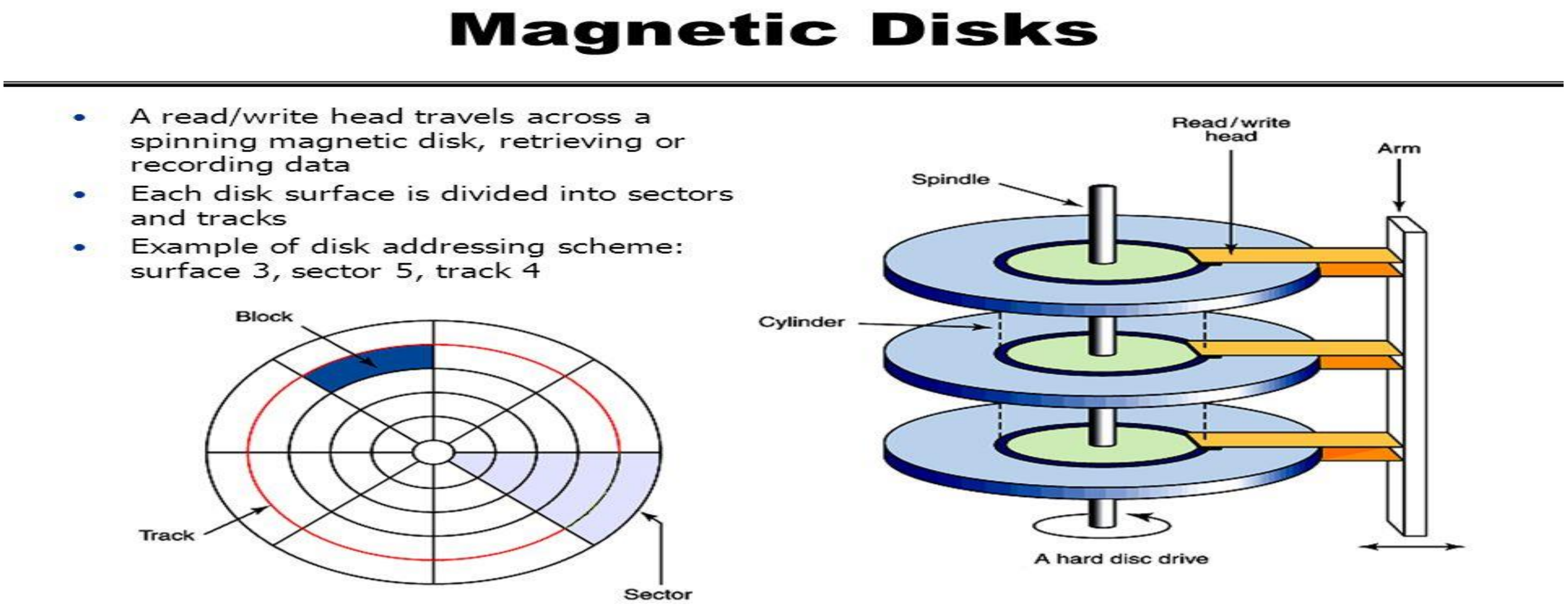
Storage Structure and Hierarchy



Magnetic Tape

Optical Tape

Main Memory (RAM)



Magnetic Disks

- A read/write head travels across a spinning magnetic disk, retrieving or recording data
- Each disk surface is divided into sectors and tracks
- Example of disk addressing scheme: surface 3, sector 5, track 4