General Information

- Class time (NC 141)

Mon (8.00-10.00)

Tue (12.00-13.00)

- Lab Tutorial (Mon, 2.00pm-4.00pm) Room 119

General Information

- **Grading Policy**
  - Midsem – 30%
  - Endsem – 50%
  - Internal – 20% (Tutorial, Class tests, Scribe and Attendance)
- http://cse.iitkgp.ac.in/~bivasm/OS2016.html
- **Text Books / References :**

1. Silbersehatz A. and Galvin P., “Operating System Concepts”, Wiley.

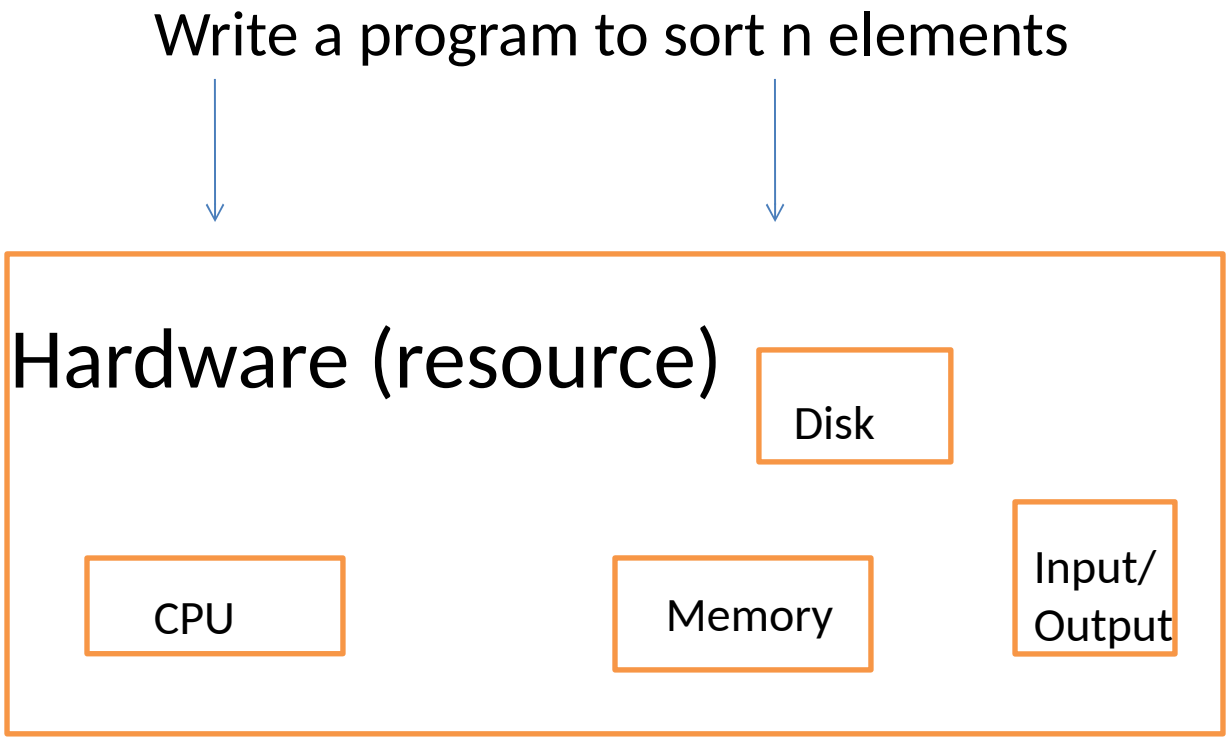
2. Tanenbaum A.S., “Operating System Design & Implementation”, Practice Hall NJ.

3. Stalling, William, “Operating Systems”, Maxwell McMillan International Editions, 1992.

4. Dietel H. N., “An Introduction to Operating Systems”, Addison Wesley.

What is an Operating System?

Why do we need an Operating System?



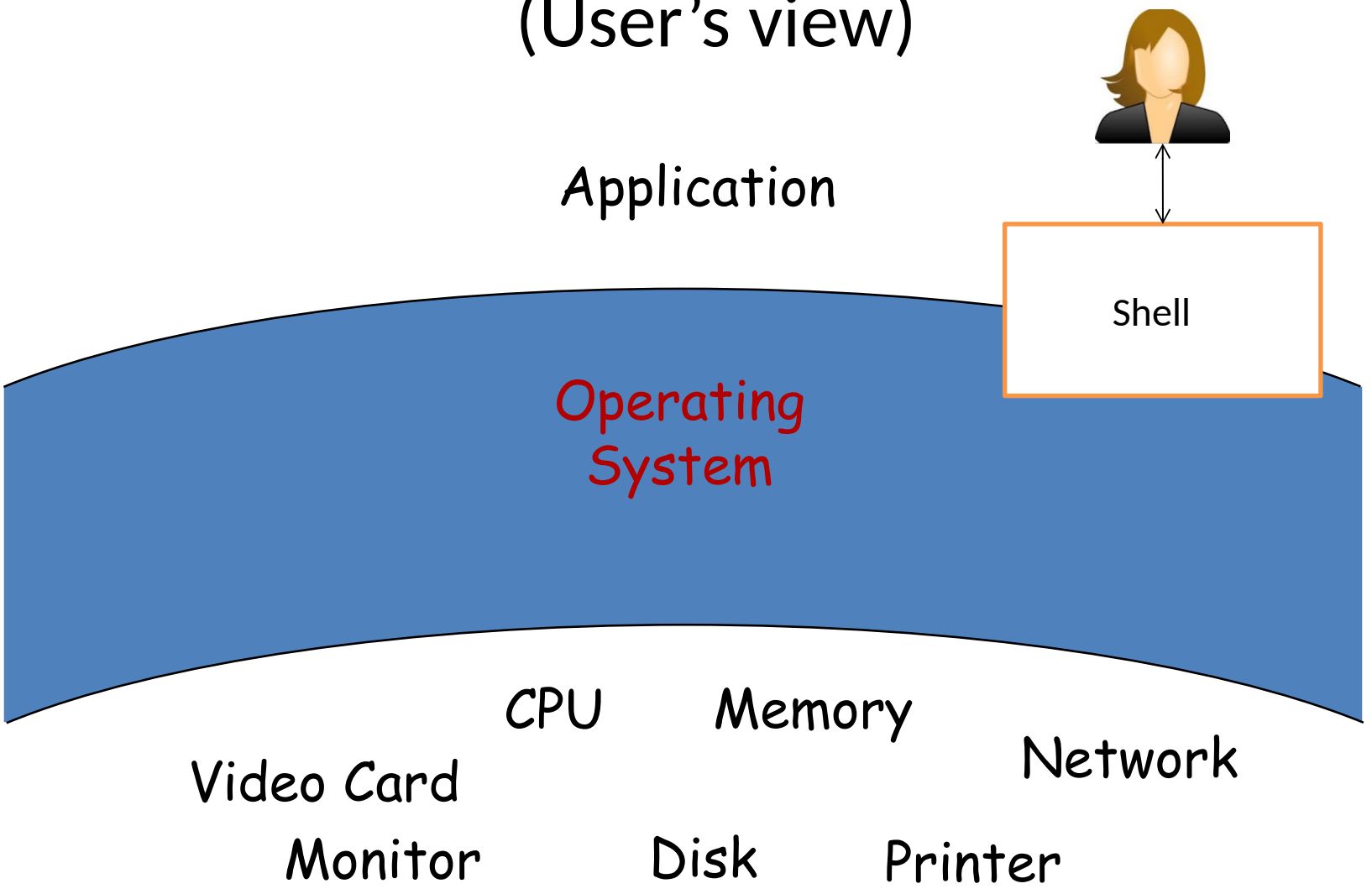
What is an Operating System? (User's view)

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Defines an interface for the user to use services provided by the system
- Creates an environment for the user

What is an Operating System? (User's view)

- Abstract Machine
  - Hides complex details of the underlying hardware
  - Provides common API to applications and services
  - Simplifies application writing
- Command Interpreter
  - Part of a OS that understands and executes commands that are entered interactively by a human being or from a program
  - Shell

What is an Operating System? (User's view)



Why is abstraction important?

- Without OSs and abstract interfaces, application writers must program all device access directly
  - load device command codes into device registers
  - understand physical characteristics of the devices

- Applications suffer!
  - very complicated maintenance and upgrading
  - no portability

What Operating Systems Do (User's view)

Depends on the point of view

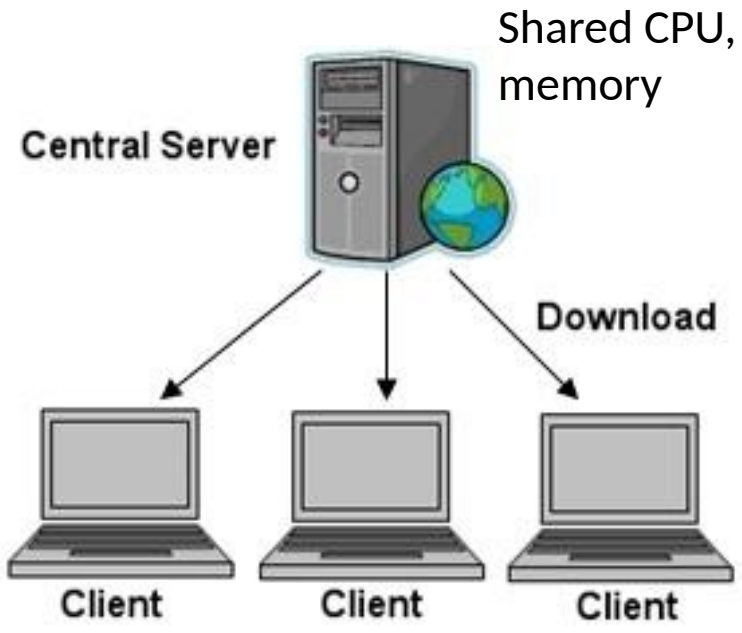
- Single user system
- Users want convenience, **ease of use**
  - Don't care about **resource utilization**

Optimized for single user experience

What Operating Systems Do (User's view)

Depends on the point of view

- Shared computer such as **mainframe** must keep all users happy
- Response time minimum
  - Keep all the users happy



What Operating Systems Do (Systems view)

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

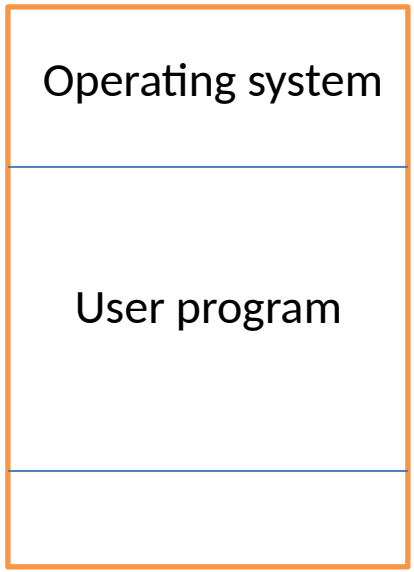
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Concept of Process

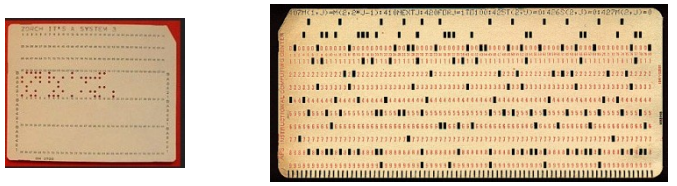
- Process
  - Program loaded in memory and in execution
- Program is a passive entity
- Process is an active entity

Types of Systems

- Batch Systems
  - Multiple jobs, but only one job in memory at one time and executed (till completion) before the next one starts

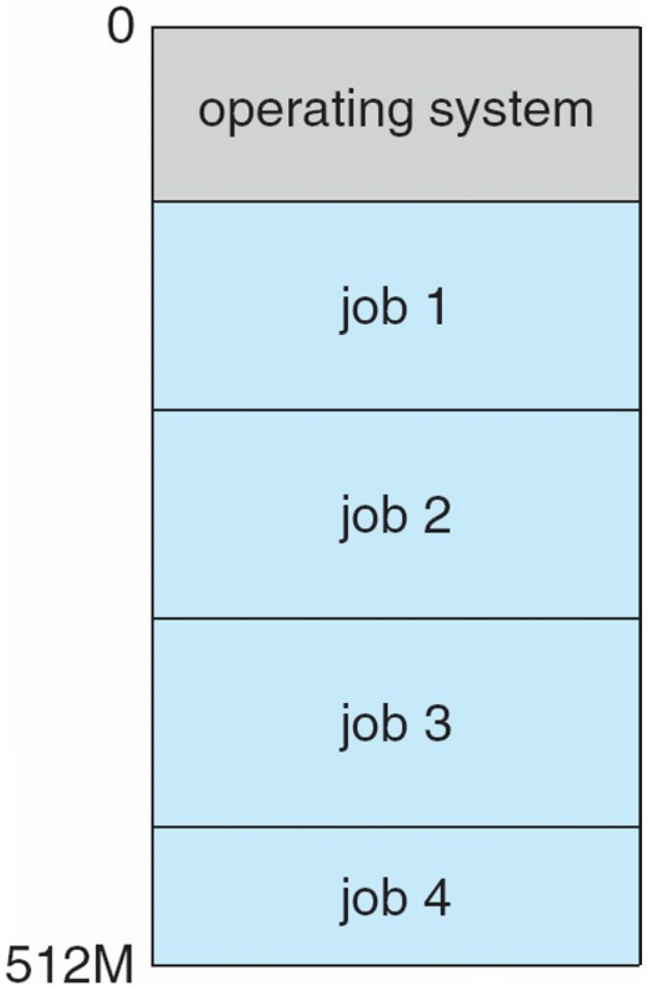


Jobs waiting



Types of Systems

- Multiprogrammed Systems
  - Multiple jobs in memory, CPU is multiplexed between them
  - Single user cannot keep CPU and I/O devices busy at all times
  - When it has to wait (for I/O for example), OS switches to another job
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**



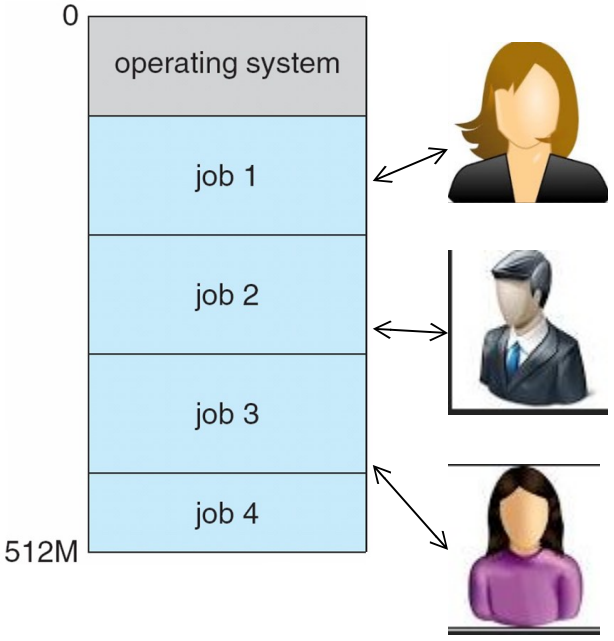
- **Effective resource utilization**
- **Poor user experience**

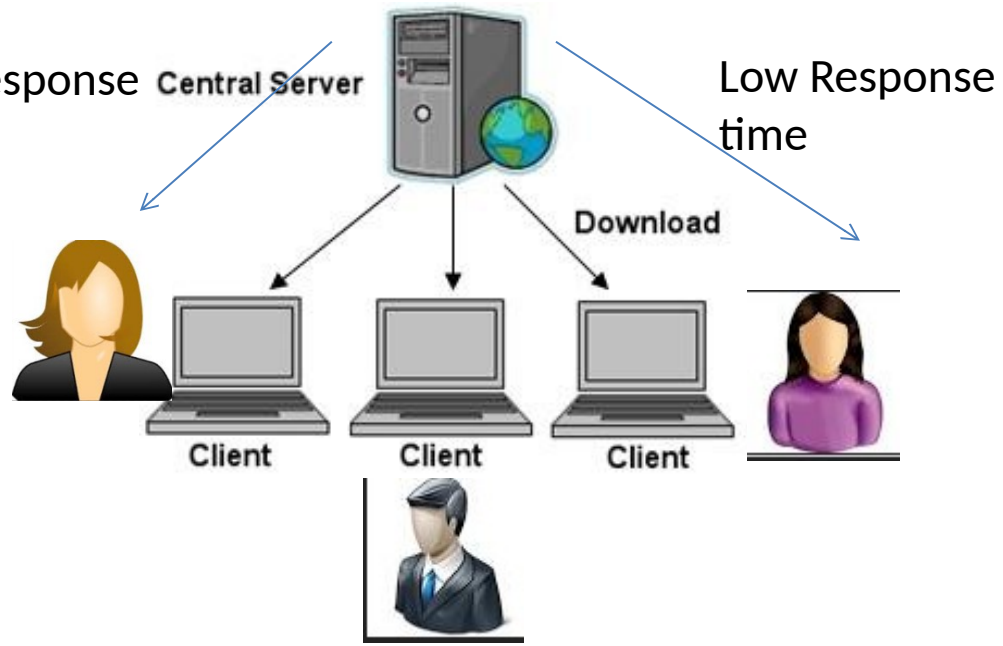
Types of Systems

- Time-sharing Systems **(multitasking)**

logical extension of multiprogramming in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory
  - If several jobs ready to run at the same time ✣ **CPU scheduling**



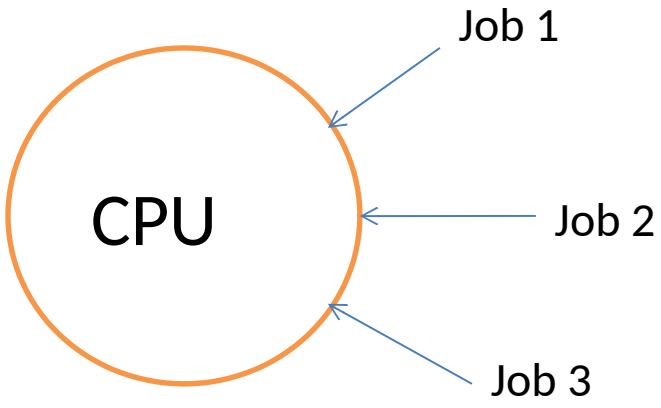


What Operating Systems Do (Systems view)

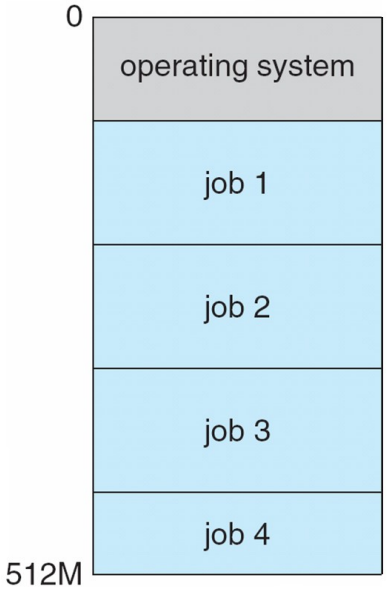
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Manage resources



1. Share the CPU with several users
2. Decide when to allocate CPU to which user (CPU scheduling)
3. Ensure fair user experience



Memory

1. Share memory with several different users
2. Should not overlap
3. Ensure protection

What Operating Systems Do (Systems view)

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

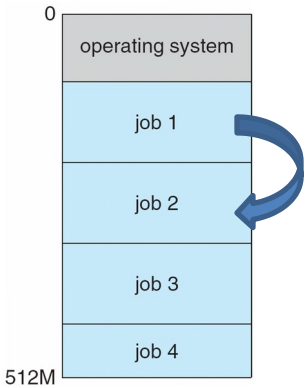
Control program (Protection)

- Multiple jobs are sharing the common resource
  - With sharing, many processes could be adversely affected by a bug in one program
  - Make sure that error in one program could cause problems only for that program
  - A job gets stuck in an infinite loop
    - Prevent correct operations of other jobs

J2, J3 waiting

CPU (J1)

  - One erroneous program might modify another program, even operating system

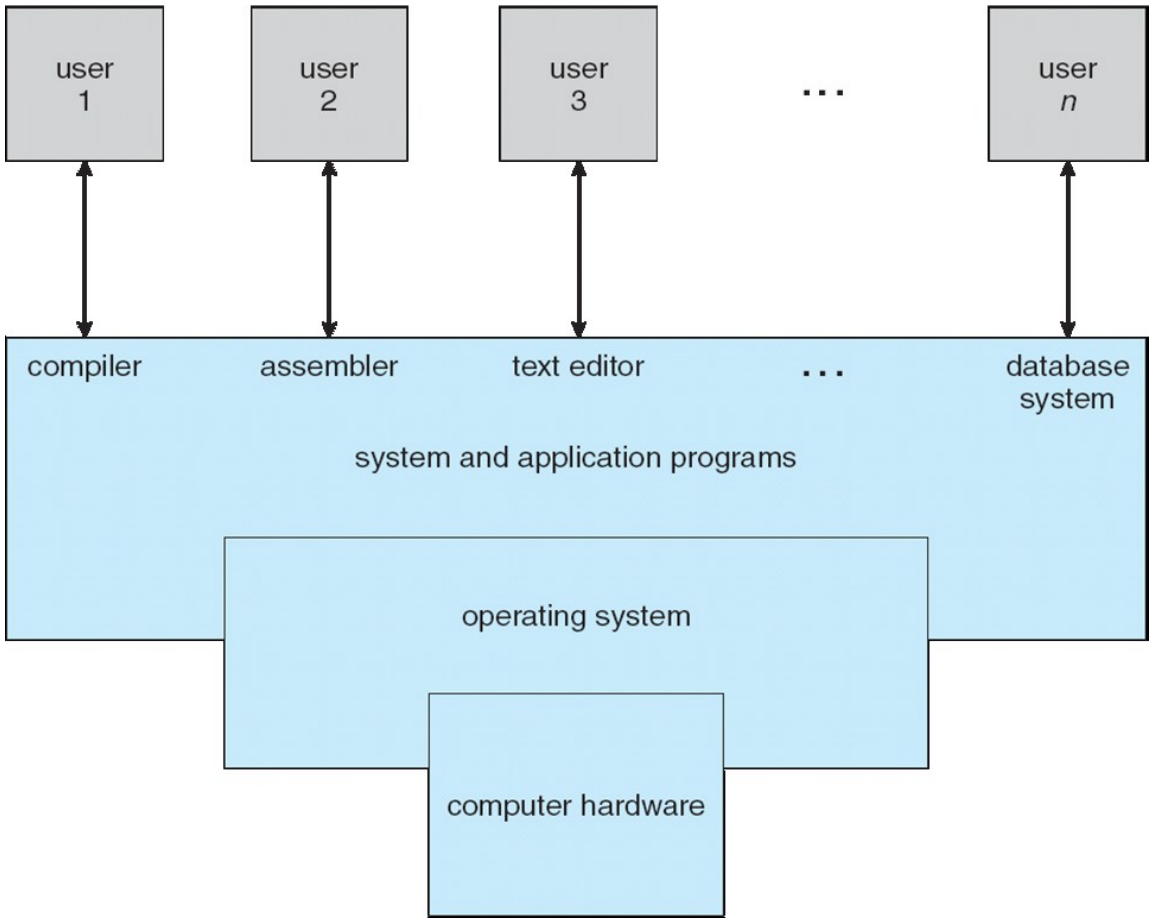


Incorrect program cannot cause other programs to execute incorrectly

Role of Operating system

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

Four Components of a Computer System

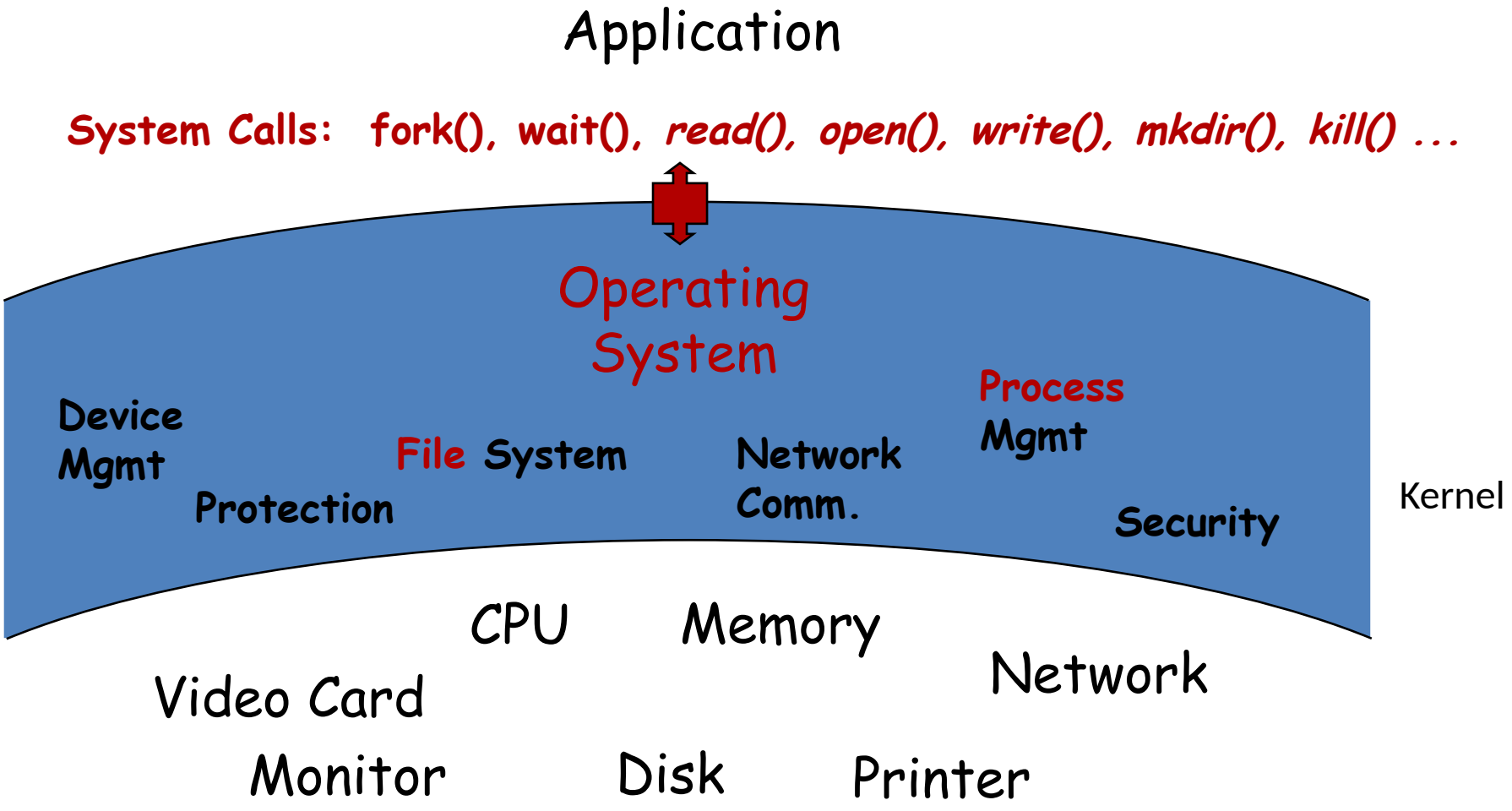


Execution of OS

- Interrupt driven
- Until an interrupt comes, OS remains Idle
- Interrupt/trap
  - Possibility 1---- error
  - Possibility 2
    - User program invokes OS code by generating Interrupt, system call
    - To perform some task reserved for OS
    - Accessing I/O devices (read, write files)

ISR

Providing abstraction via system calls



Execution of OS

- Interrupt driven
- Until an interrupt comes, OS remains Idle
- User program invokes OS code by generating Interrupt, system call
  - To perform some task reserved for OS
  - Accessing I/O devices (read, write files)
- Any difference in execution between user and OS program?

Operating-System Operations

- Must distinguish between the use level code and OS code
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - System call changes mode to kernel, return from call resets it to user

System boot

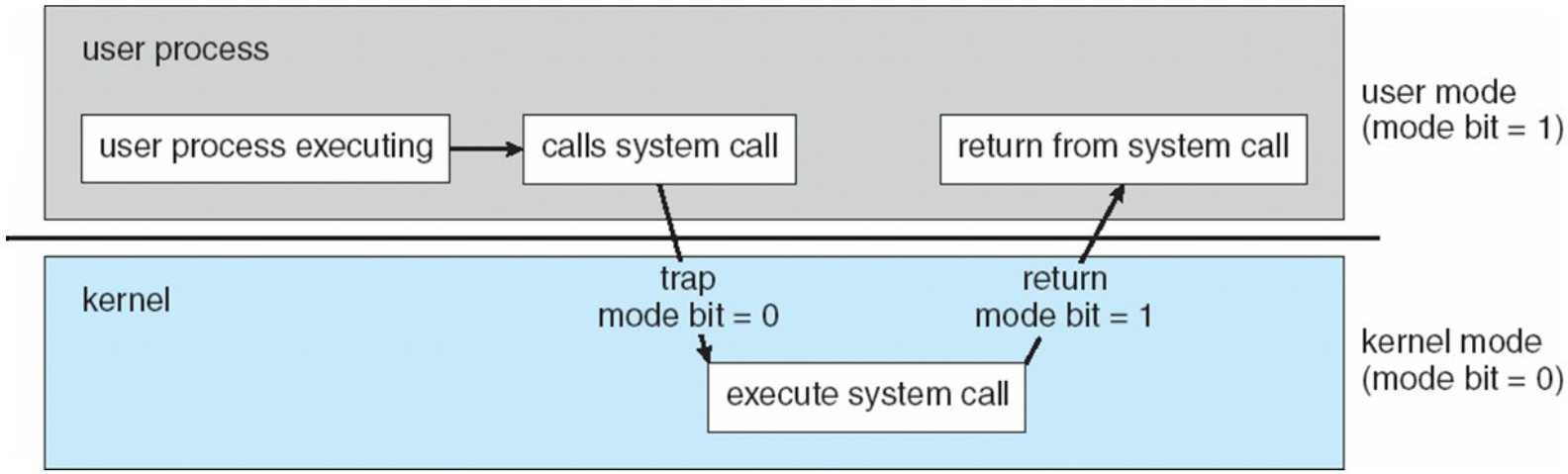
Hardware starts kernel mode

Load Operating system

Start user application

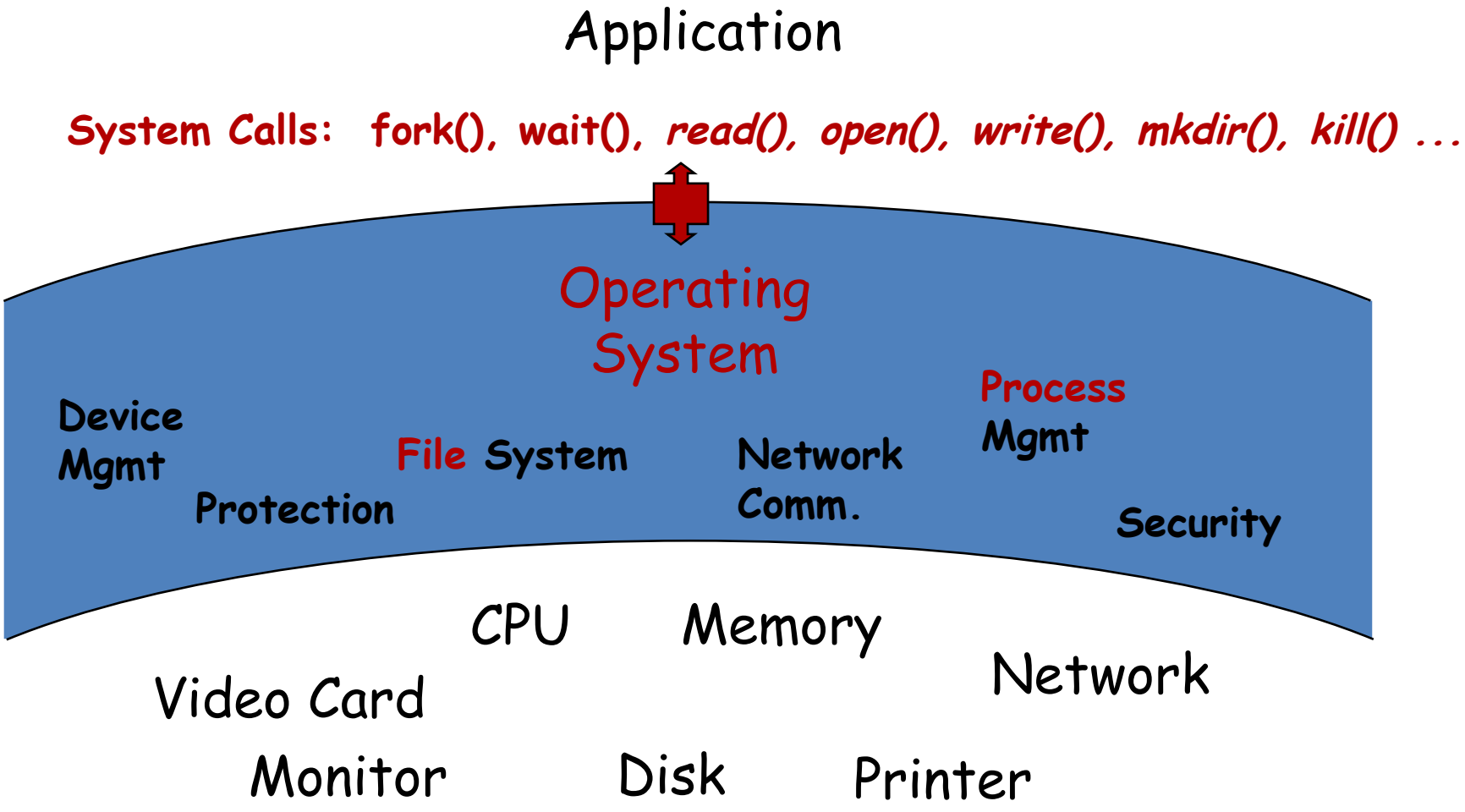
Switch to User mode

Whenever Trap or interrupt occurs, hardware switches to user to kernel mode



Some instructions designated as **privileged**, only executable in kernel mode

Providing abstraction via system calls



What Operating Systems Do (Systems view)

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use

- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

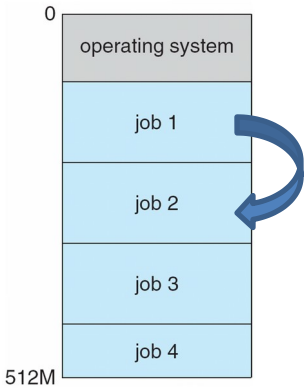
Control program (Protection)

- Multiple jobs are sharing the common resource
  - With sharing, many processes could be adversely affected by a bug in one program
  - Make sure that error in one program could cause problems only for that program
  - A job gets stuck in an infinite loop
    - Prevent correct operations of other jobs

J2, J3 waiting

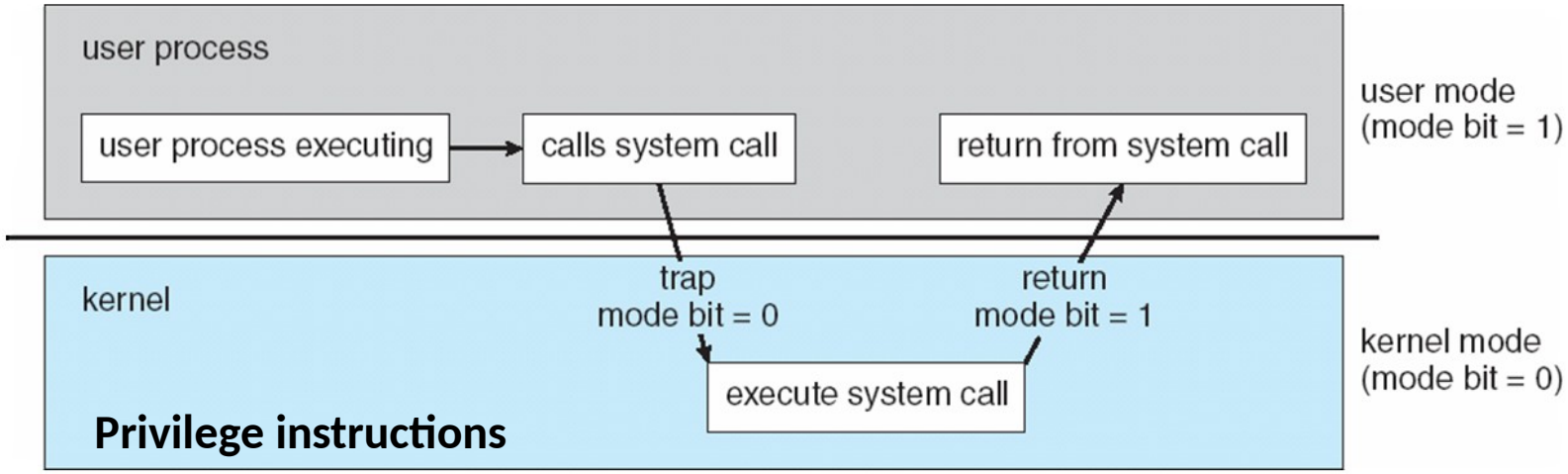
CPU (J1)

  - One erroneous program might modify another program, even operating system



Incorrect program cannot cause other programs to execute incorrectly

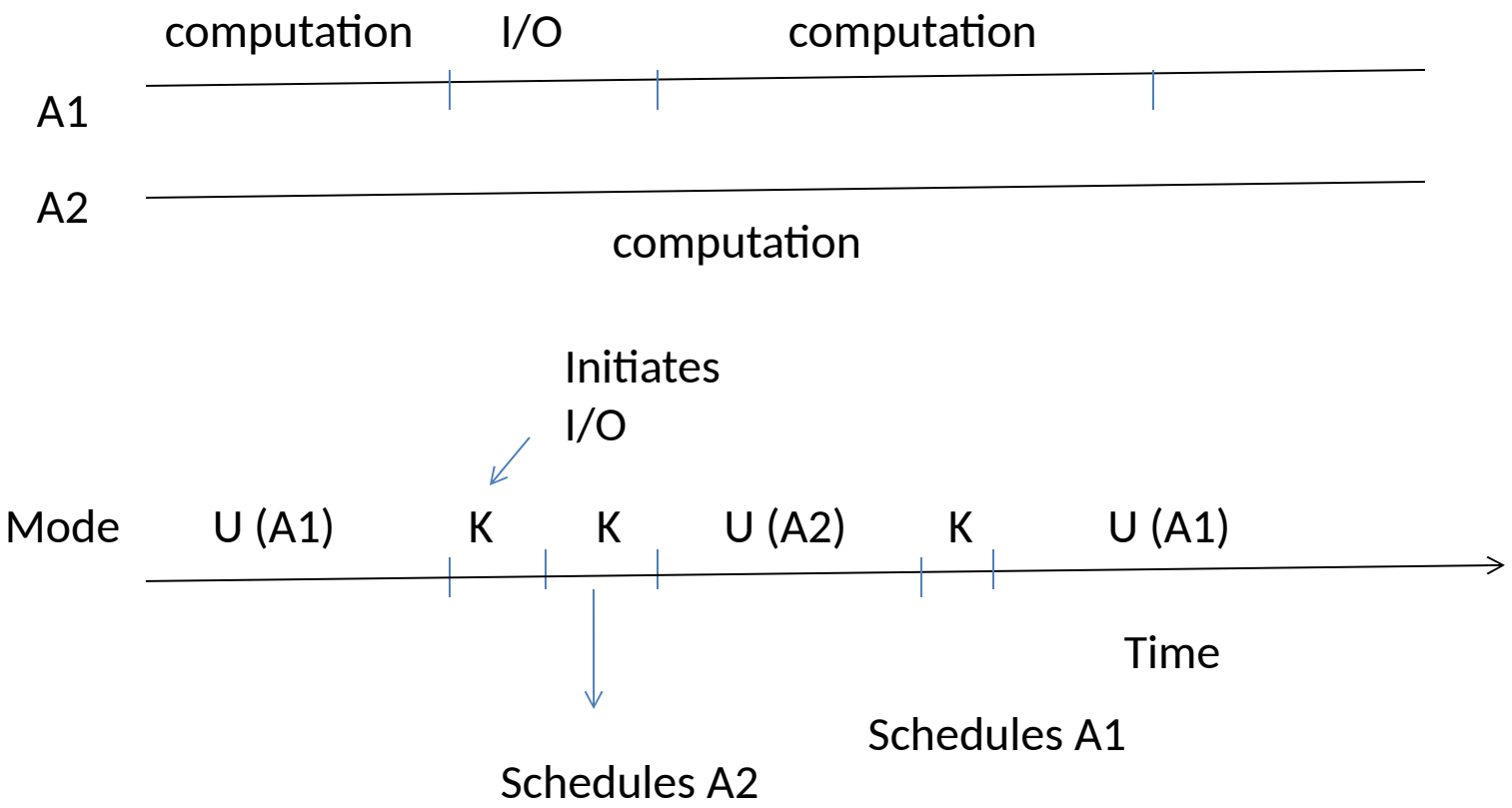
Dual-mode operation allows OS to protect itself and other system components



- Software error creates **exception** or **trap**
  - Division by zero, request for operating system service, setting timer
- Restricts user process from executing **privilege instruction**

- E.g. Segmentation fault!

Mode change


computation
I/O
computation
A1
A2
computation
Initiates
I/O
Mode
U (A1)
K
K
U (A2)
K
U (A1)
Time
Schedules A1
Schedules A2

What is the difference between

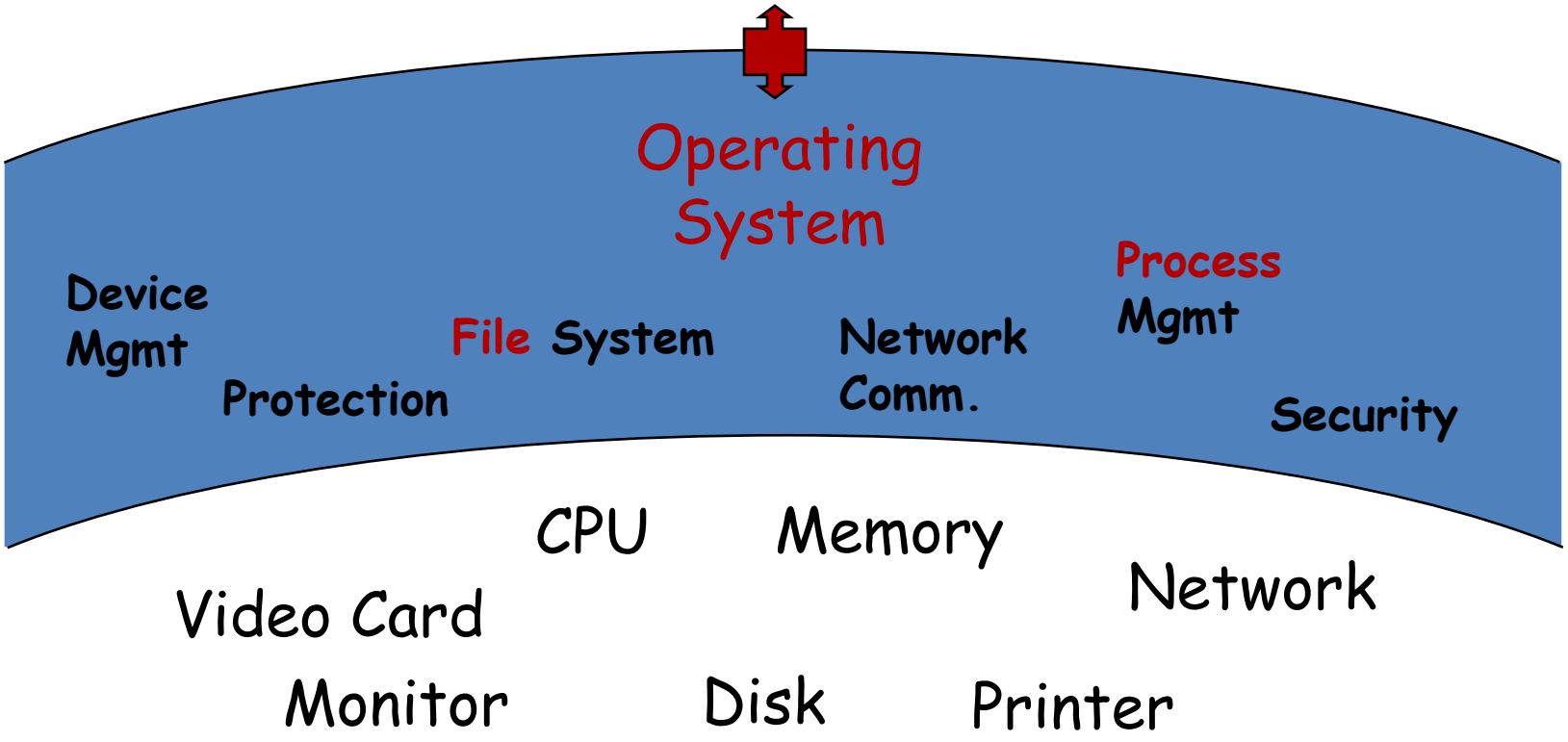
Privileged Instruction and System call?

System calls

Win 32
POSIX
JVM

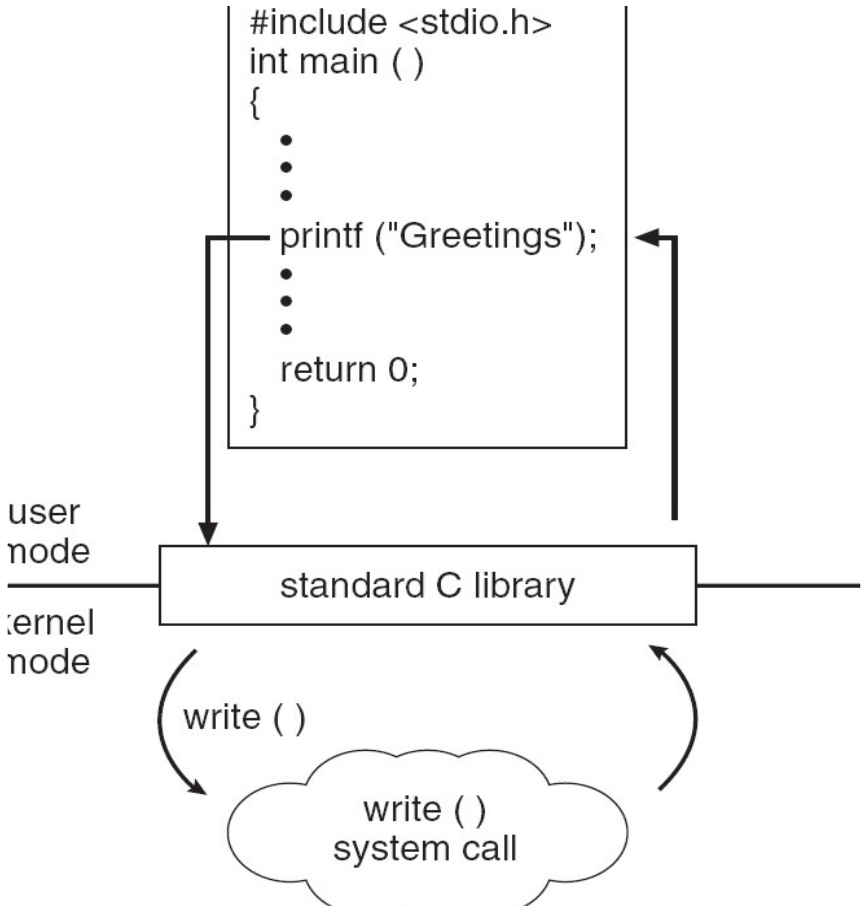
Application

System Calls: *fork(), wait(), read(), open(), write(), mkdir(), kill() ...*



Standard C Library Example

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



Resources Managed by OS

- Physical
  - CPU, Memory, Disk, I/O Devices like keyboard, monitor, printer
- Logical
  - Process, File, …

Hence we have

1. Process management
2. Memory management
3. File management
4. I/O management

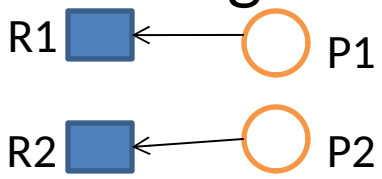
Process Management

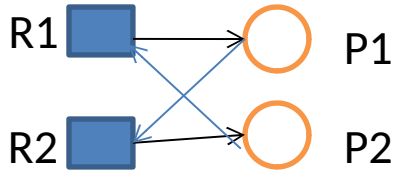
- A process is a program in execution. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
  - CPU time
- Representation of process
  - Process has one **program counter** specifying location of next instruction to execute
  - Data structure (stores information of a process)
- Many processes may be associated with the same program
- Typically system has many processes
  - some user processes,
  - some operating system processes
- Life cycle of a process
  - States
  - Arrival, Computation, I/O, I/O completion, termination

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

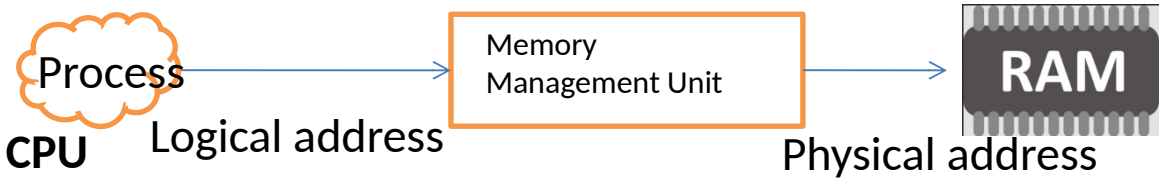
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Process scheduling
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling





Memory Management

- All instructions and data in memory in order to execute
  - Translate the logical address to physical address

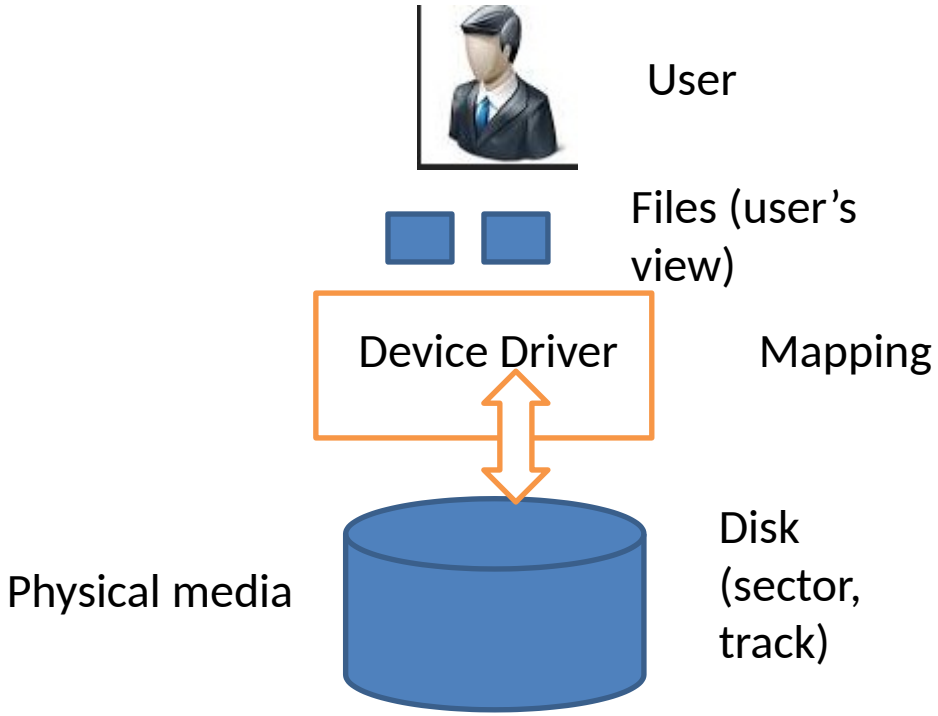


- Process terminates => MMU declares that the memory space is available
- Multiprogramming: Memory management manages several processes in memory
  - Optimizing CPU utilization and computer response to users
- Ensure memory protection
  - Track illegal address

- Memory management activities
  - Keeping track of which parts of memory are currently being used and by which process
  - Allocating and deallocating memory space as needed
- Introduces Virtual memory
  - If the process size is bigger than the RAM size
- Hardware support

File Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **File**
  - File => Collection of related information defined by the creator
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)



File Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

- OS implements the abstract concept of file by managing mass storage media (disk etc) and devices that control them

- Files usually organized into directories
- Access control on most systems to determine who can access what
- File-System management
  - Creating and deleting files and directories
  - Primitives to manipulate files and dirs
  - Mapping files onto secondary storage

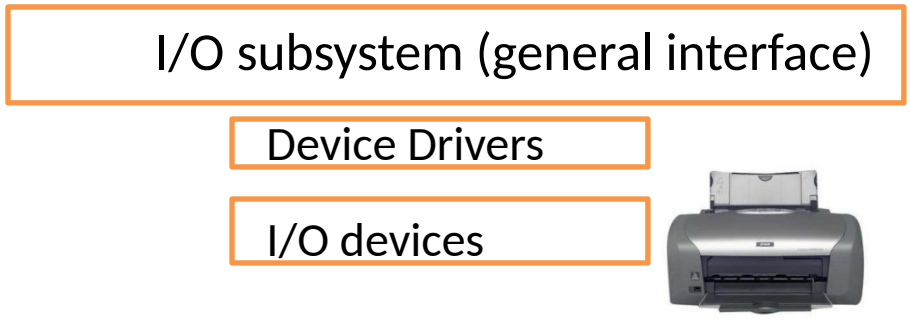
Disk Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
  - Most of the programs are stored on disk
- Proper management is of central importance
- Entire speed of computer operation depends on disk subsystem and its algorithms
- OS activities
  - Storage allocation (logical blocks)
  - Free-space management
  - Disk scheduling

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user

- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance)
  - General device-driver interface
  - Drivers for specific hardware devices



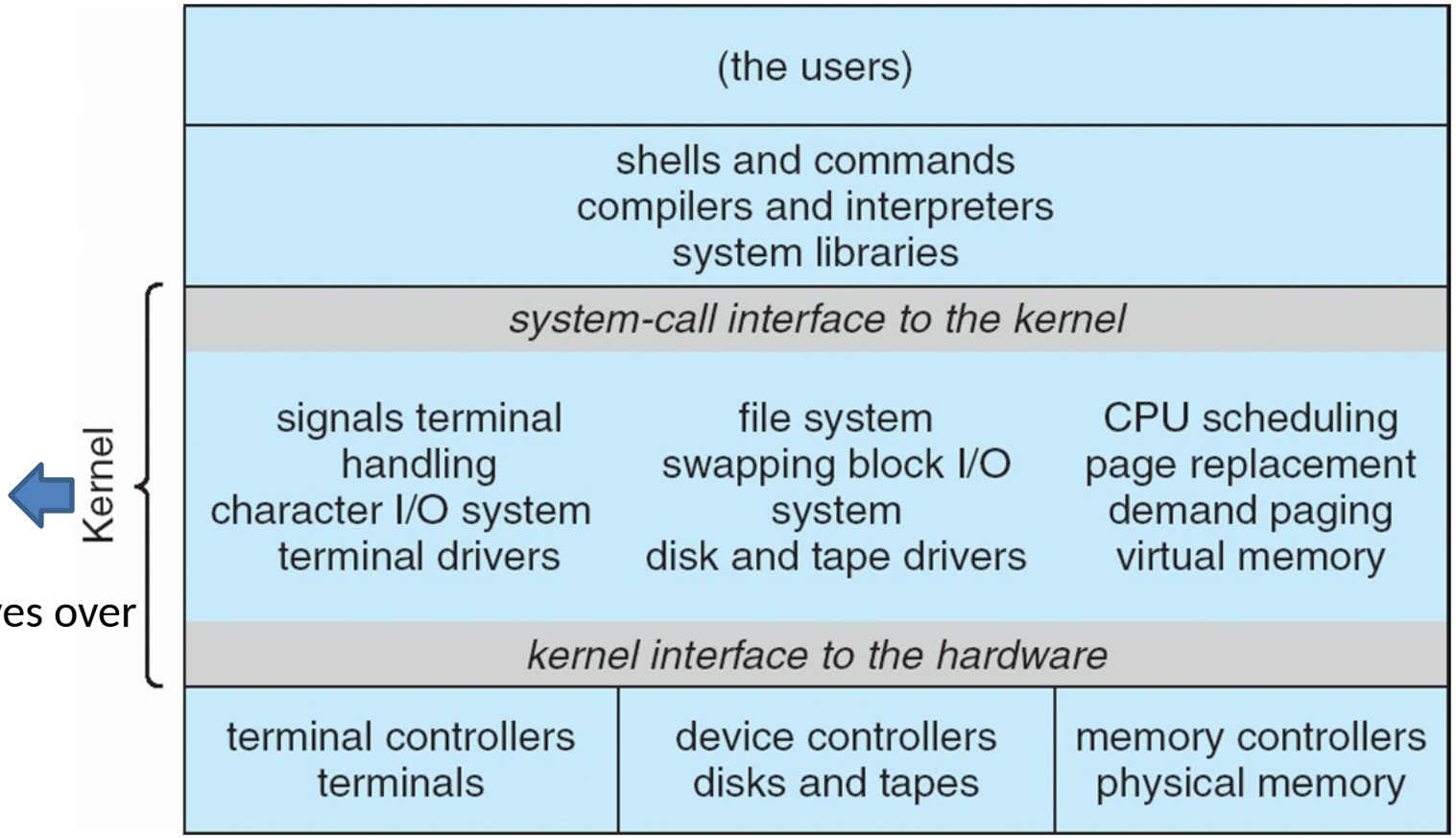
OS design and structure

- Large complex system
  - Designed carefully
    - if it is to function properly
    - Modified easily
- Common approach
  - Partition the tasks into small components/modules
  - Each module must accomplish some specified task

UNIX

- UNIX – consists of two separable parts
  - Systems 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



Evolves over time

MS-DOS

- Application programs can directly access I/O routines
- Makes the system vulnerable
- No mode bit
  - Limitations in hardware
  - Intel 8088