Agenda: Day 1

- Introduction to OS - theory
- Introduction to Linux – theory
- Basic Linux commands
- Hands on – Practice Linux commands

Introduction to Operating Systems

Prachi Pandey

System Software Development Group (SSDG)

C-DAC, Bangalore

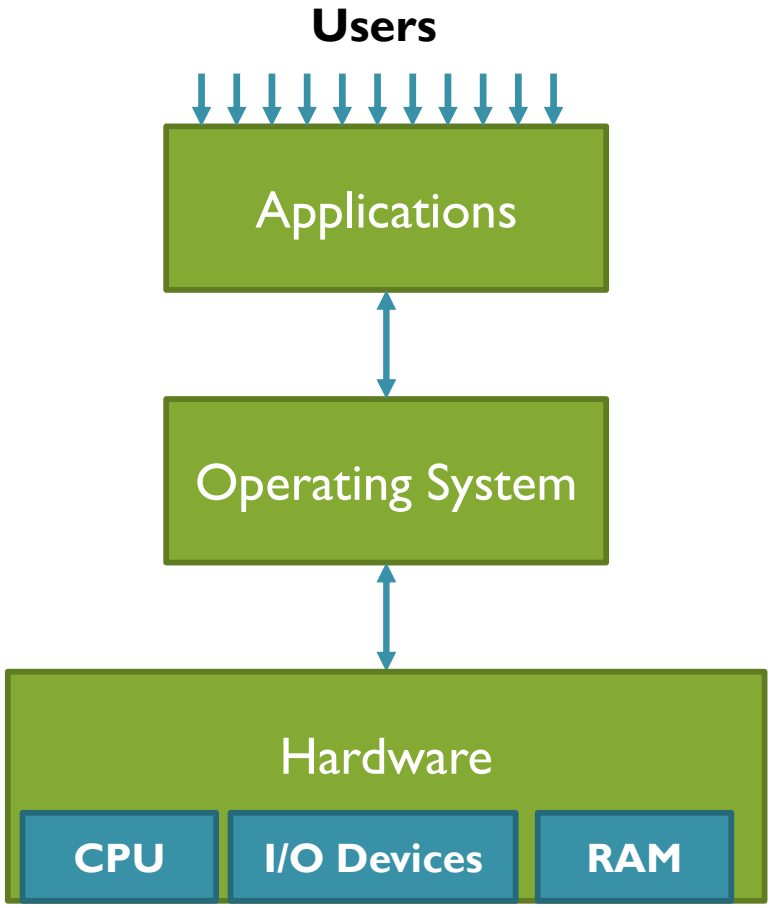
Topics

- About
- Evolution/History of OS
- Types of OS
- Components/Functions of OS
- User and Kernel space

Popular Operating Systems

What is an Operating System

- An Operating System is a program that acts as an interface between a user of a computer and the computer hardware



What if we don't have an OS

- How do you get your program onto the hardware?
- How do you print out the answer?

Once upon a time, users had to Toggle in program in binary and read out the answer from LED's!

Altair 8080

Why do we need an OS?

- Primary Goal – Convenience
- Secondary Goal – Efficiency

Or it could be the other way round!

Operating Systems Goals

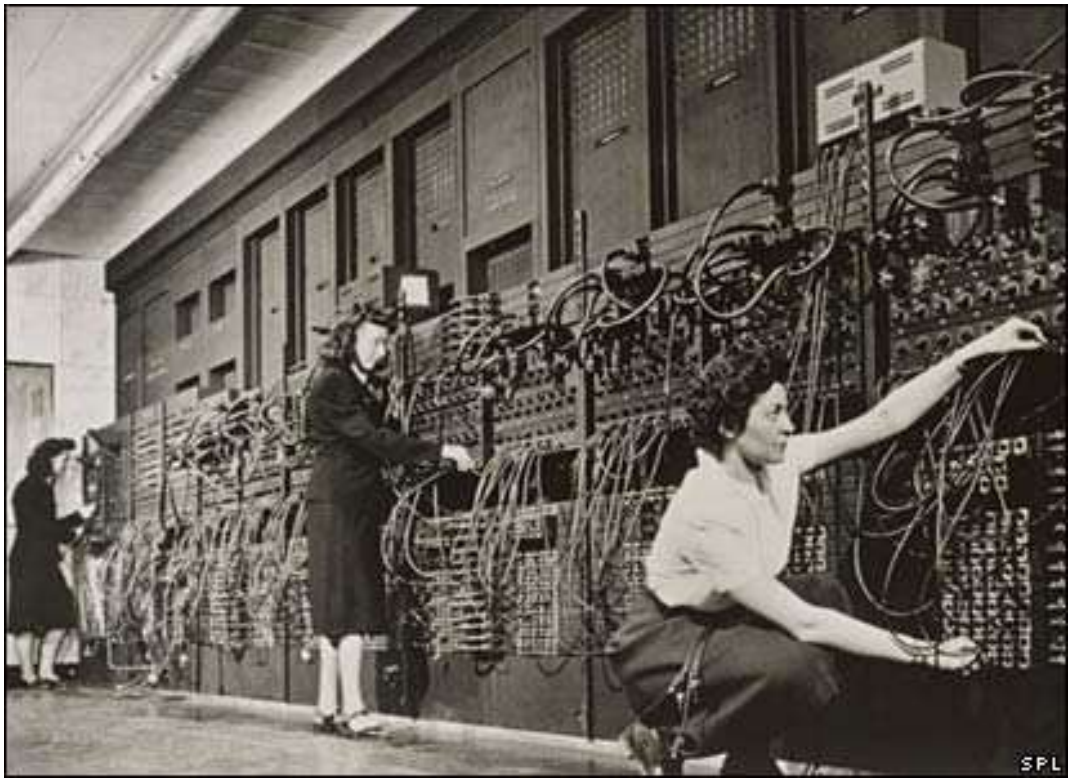
- Make the computer system convenient to use
  - Provide clean, easy-to-use abstractions of physical resources
- Use the computer hardware and resources in an efficient manner
  - Manage protection, isolation, and sharing of resources
- Control/execute user/application programs by providing standard services

Evolution/History of OS

- Bare machine (No Operating System)
- Batch Operating Systems
- Multiprogramming Operating Systems
- Timesharing / Multitasking Operating Systems
- Multiprocessing Operating Systems
- Real Time Operating Systems
- Network OS
- Embedded OS

History

- It all started with computer hardware in about 1940s.



ENIAC 1943

History contd.

- ENIAC (Electronic Numerical Integrator and Computer), at the U.S. Army's Aberdeen Proving Ground in Maryland.
  - built in the 1940s,
  - weighed 30 tons,
  - was eight feet high, three feet deep, and 100 feet long
  - contained over 18,000 vacuum tubes that were cooled by 80 air blowers.

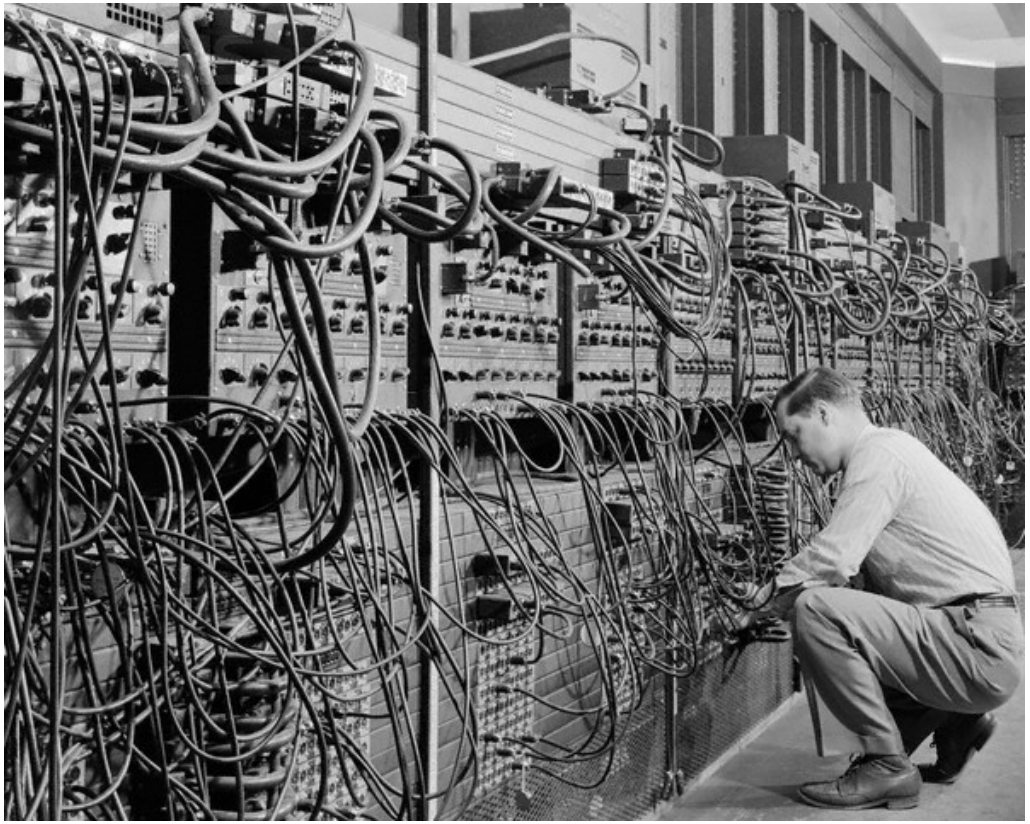
History contd.

- Computers were using vacuum tube technology.

ENIAC's vacuum tubes

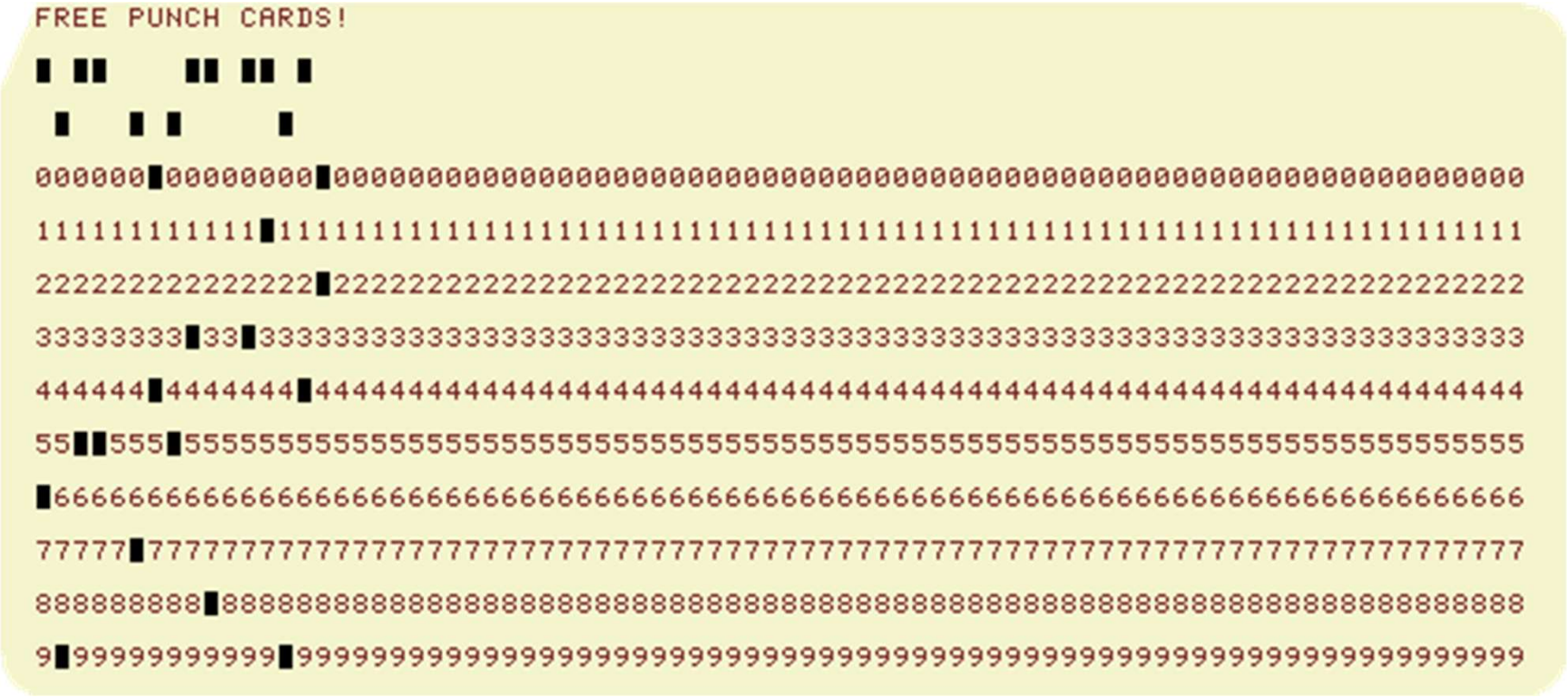
History contd.

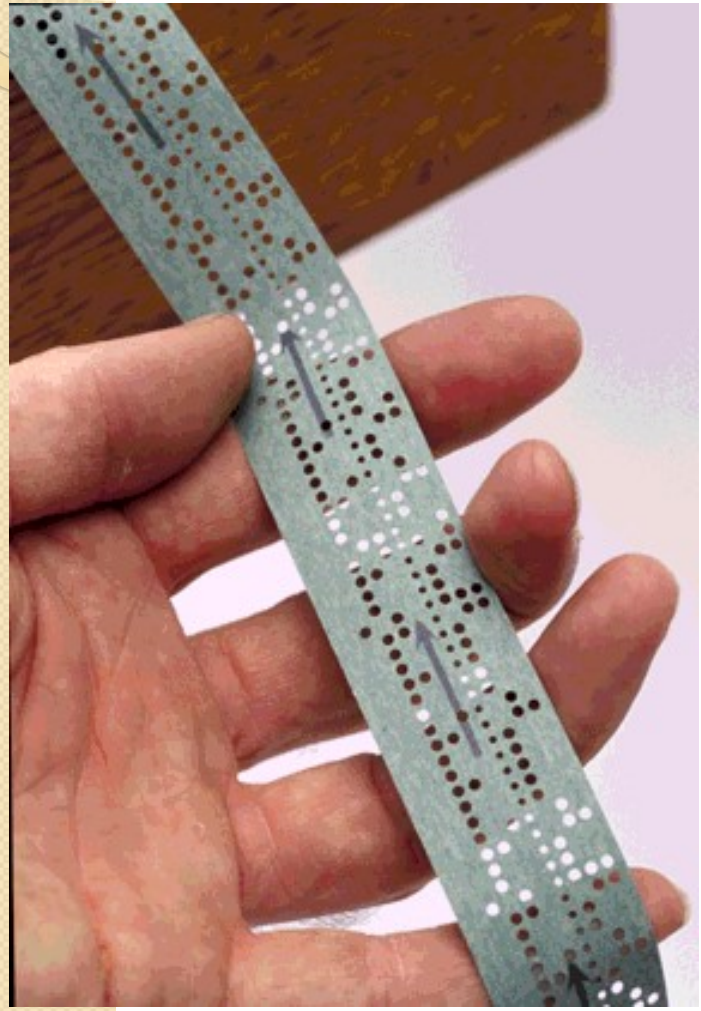
Programs were loaded into memory manually using switches, punched cards, or paper tapes.

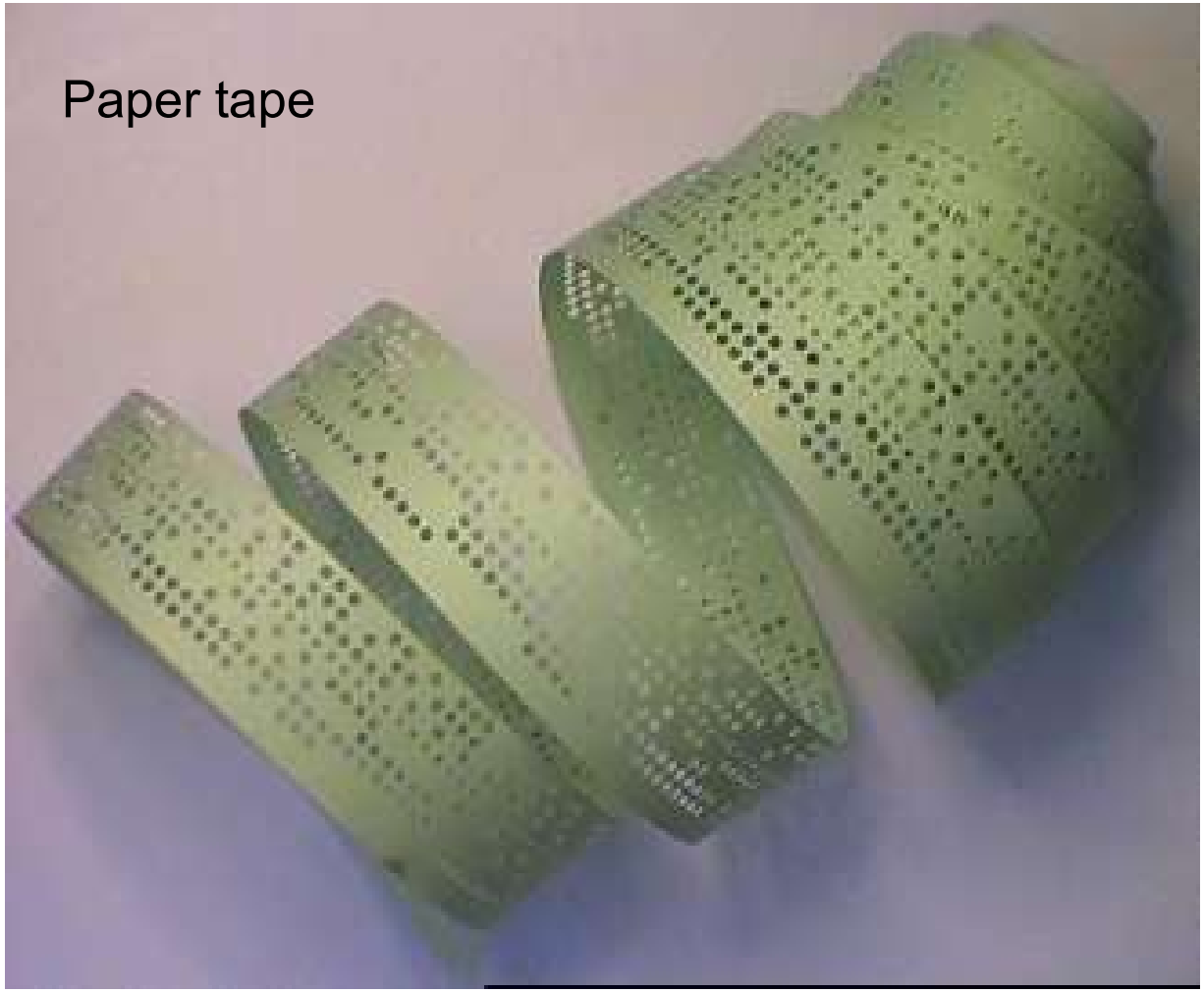


ENIAC : coding by cable connections

Punch Card





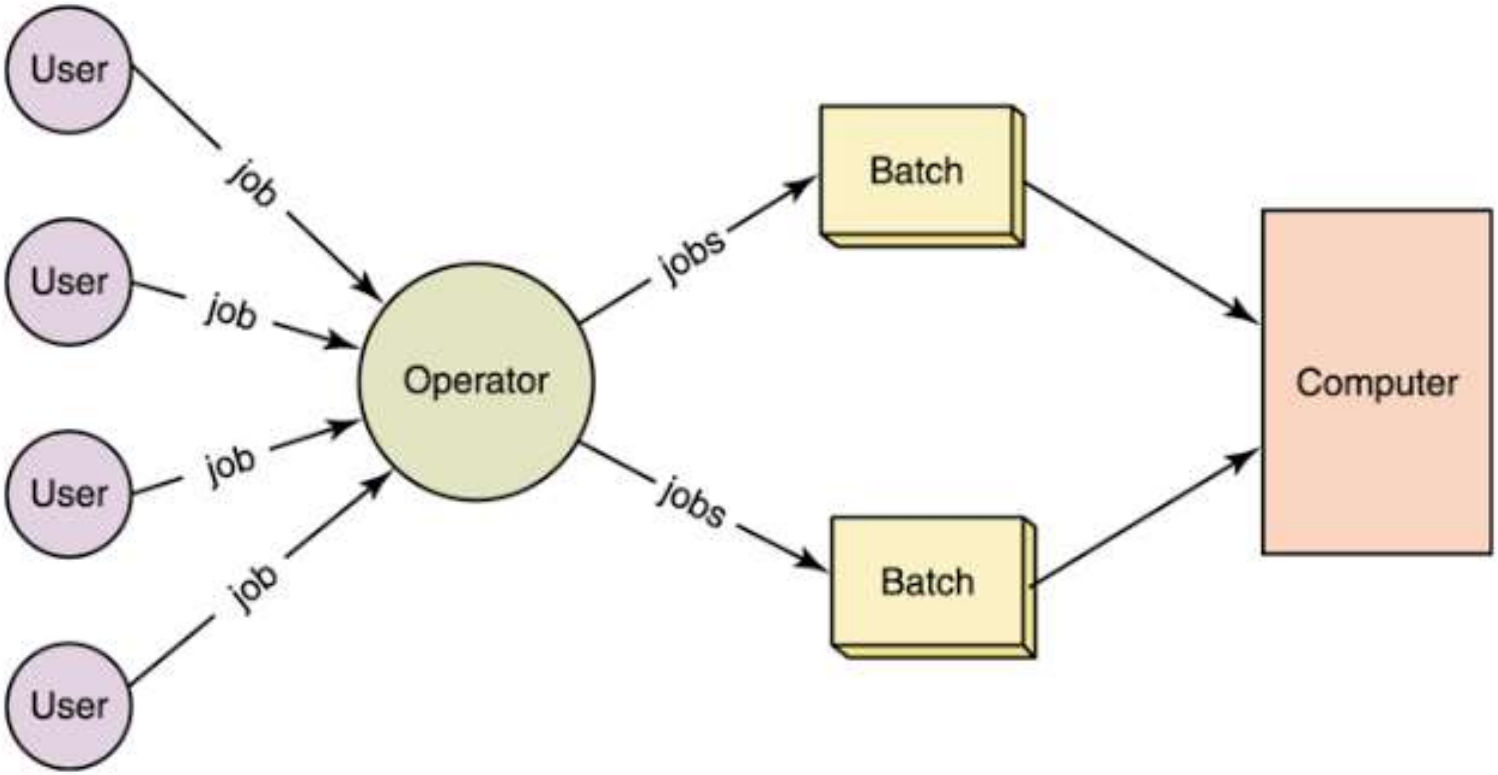

Paper tape

Types of OS

- Batch Operating Systems
- Multiprogramming Operating Systems
- Timesharing / Multitasking Operating Systems
- Multiprocessing Operating Systems
- Real Time Operating Systems
- Network OS
- Embedded OS

Batch Operating Systems

- Users did not interact with computer directly.
- Each user submitted his/her jobs to the operator who in turn fed the programs to the computer.
- User had to come back later to collect the output.



Batch Operating Systems

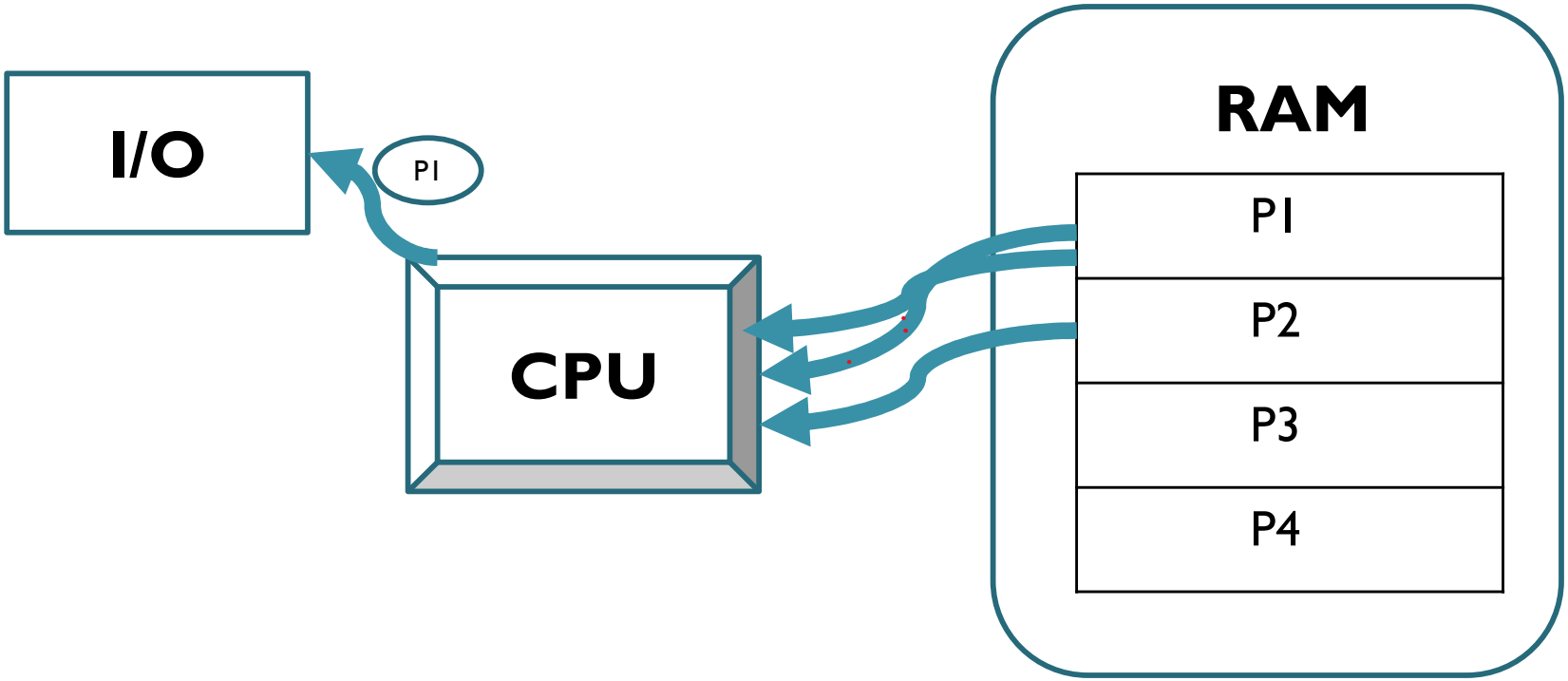
Disadvantages

- Lack of interaction between the user and job
- CPU is often idle
- Difficult to provide the desired priority
- Job Starvation

Multiprogramming OS

- Multiprogramming means sharing of resources between more than one processes.
- CPU time is not wasted, because, while one process moves on some I/O work, the OS picks another process to execute till the current one passes to I/O operation.
- Goal of multiprogramming is to efficiently utilize all of the computing resources.

Multi Programming



Time Sharing Operating System

Time sharing or multitasking is a logical extension of multiprogramming. It is same as multiprogramming with preemption.

- OS switches between user's programs very quickly, generally in round-robin fashion.
- Switching between users is very fast.
- Goal is to give the illusion that each user has his/her own machine.
- Response time is a priority.

Multiprocessing Operating System

- Has multiple processors sharing memory and peripheral devices.
- With this configuration, they have greater computing power and higher reliability.
- Multiprocessor systems are classified into two as tightly-coupled (shared memory) and loosely-coupled (distributed).
- In the tightly-coupled one, each processor is assigned a specific duty but processors work in close association, sharing the same memory.
- In the loosely coupled one, each processor has its own memory and copy of the OS.

Real Time Operating Systems

- A real-time operating system is an operating system that guarantees to process events or data within a stipulated time.

- Two kinds – Hard and soft RTOS

❑ Hard real time

- Strictly follow deadlines, no delay acceptable
- Failure if response time too long.
- Missile launch, air bags in car

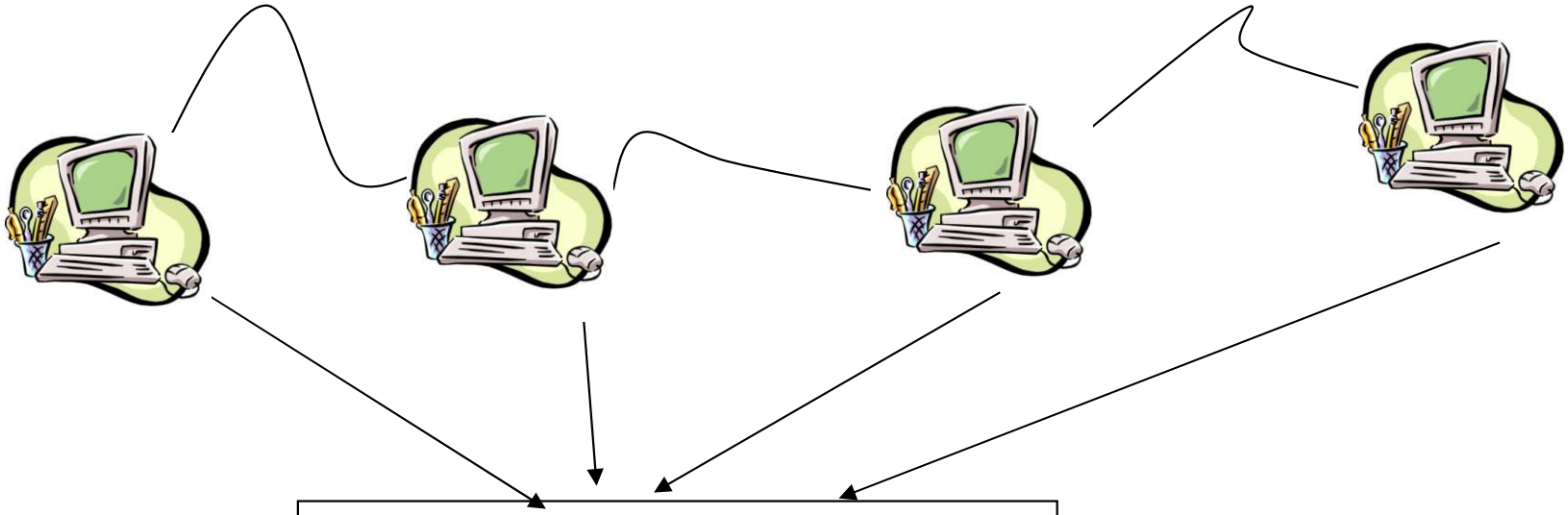
❑Soft

- Less accurate if response time is too long.
- Useful in applications such as multimedia, virtual reality.

Network Operating Systems

- Use of the networks required OSs appropriate for them.
- In **network systems**, each process runs in its own machine but the OS have access to other machines.
- In networks, users are aware of the fact that s/he is working in a network and when information is exchanged.
- The user explicitly handles the transfer of information.
- By this way, file sharing, messaging, etc. became possible

Network OS



Each is a computer having its own CPU, RAM, etc. An OS supporting networks is installed on them.

Embedded Operating System

- Embedded system is a combination of hardware and software designed to do a specific function
- Specifically configured for a certain hardware
- Work on a fixed functionality e.g. microwave, AC, etc. which cannot be changed

Mobile Operating System

- Software that allows smartphones, tablet PCs and other devices to run applications and programs.

- Two of the most widely adopted mobile OS are
  - iPhone's OS, iOS,
  - Google's open source OS, Android

Operating System Components

- Resource Management
- Process Management
- Memory Management
- Storage/Filesystem Management
- Security

Resource Management

- Manages and protects multiple computer resources: CPU, Internal/External memory, files, Applications, Communication channels, etc…
- Handles and allocates resources to multiple users or multiple programs running at the same time and space (e.g., processor time, memory, I/O devices).
- Decides between conflicting requests for efficient and fair resource use (e.g., maximize throughput, minimize response time).
- Load balancing

Process Management

- A process is a program in execution
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
- Process needs resources to accomplish its task
- Process termination requires reclaim of any reusable resources
- Concurrency is achieved by multiplexing the CPUs among the processes / threads

Process Management Activities

Complete activity required for managing a process throughout its lifecycle

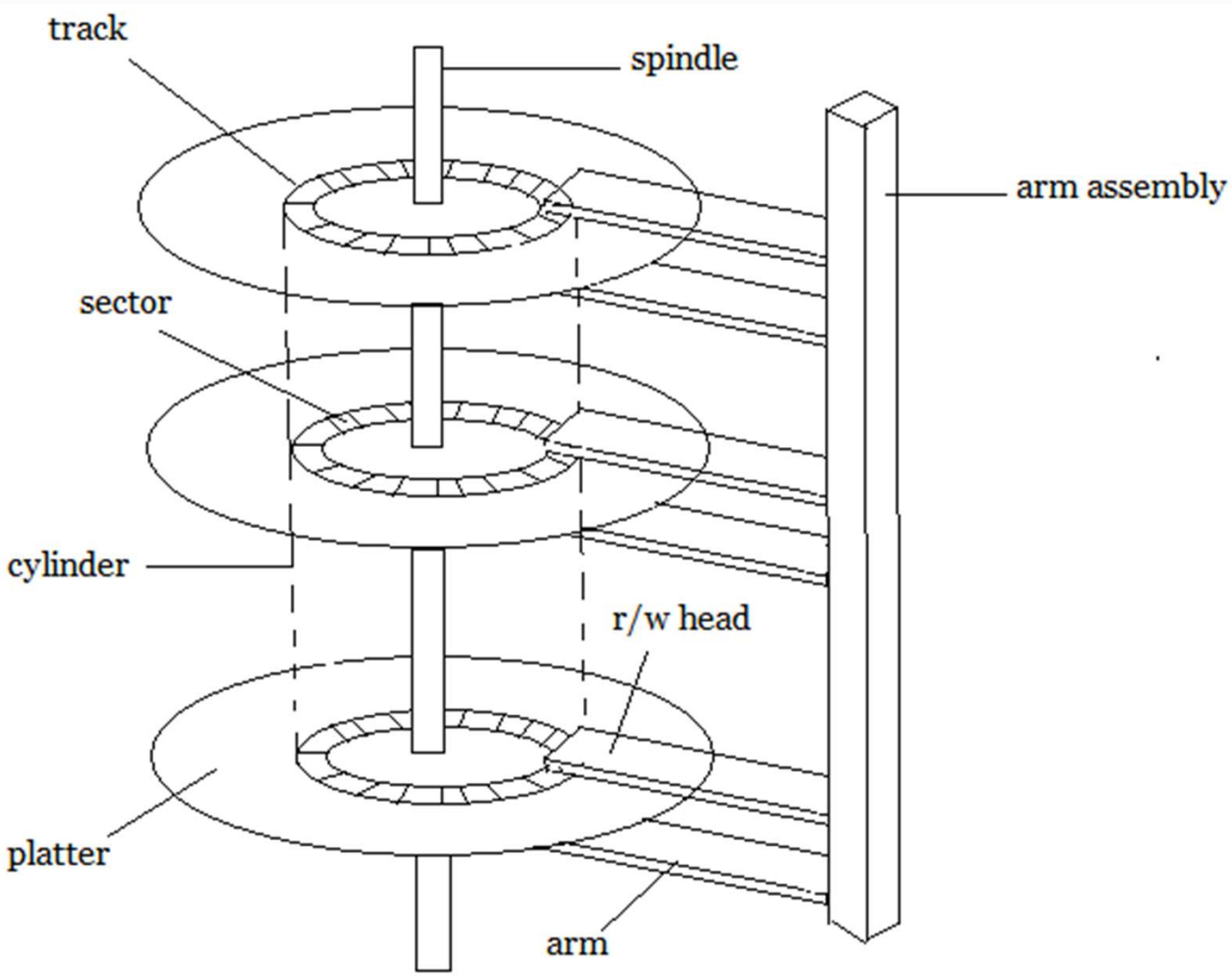
- Allocate resources to processes
- Enable processes to share and exchange information
- Protect the resources of each process from other processes
- Enable synchronization among processes.
- Maintain a data structure for each process
- Execution and Control of processes

Memory Management

- To execute a program all (or part) of the instructions must be in main memory
- All (or part) of the data that is needed by the program must be in memory
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities:
  - Allocating and deallocating memory space as needed
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory

Mass - Storage 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
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection



Structure of a magnetic disk

Protection and Security

- **Protection** – mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service

Protecting processes from each other

Problem:

How to run multiple applications in such a way that they are protected from one another

Goal:

- Keep User Programs from Crashing each other
- Keep User Programs from Crashing OS

Keep Parts of OS from crashing other parts?

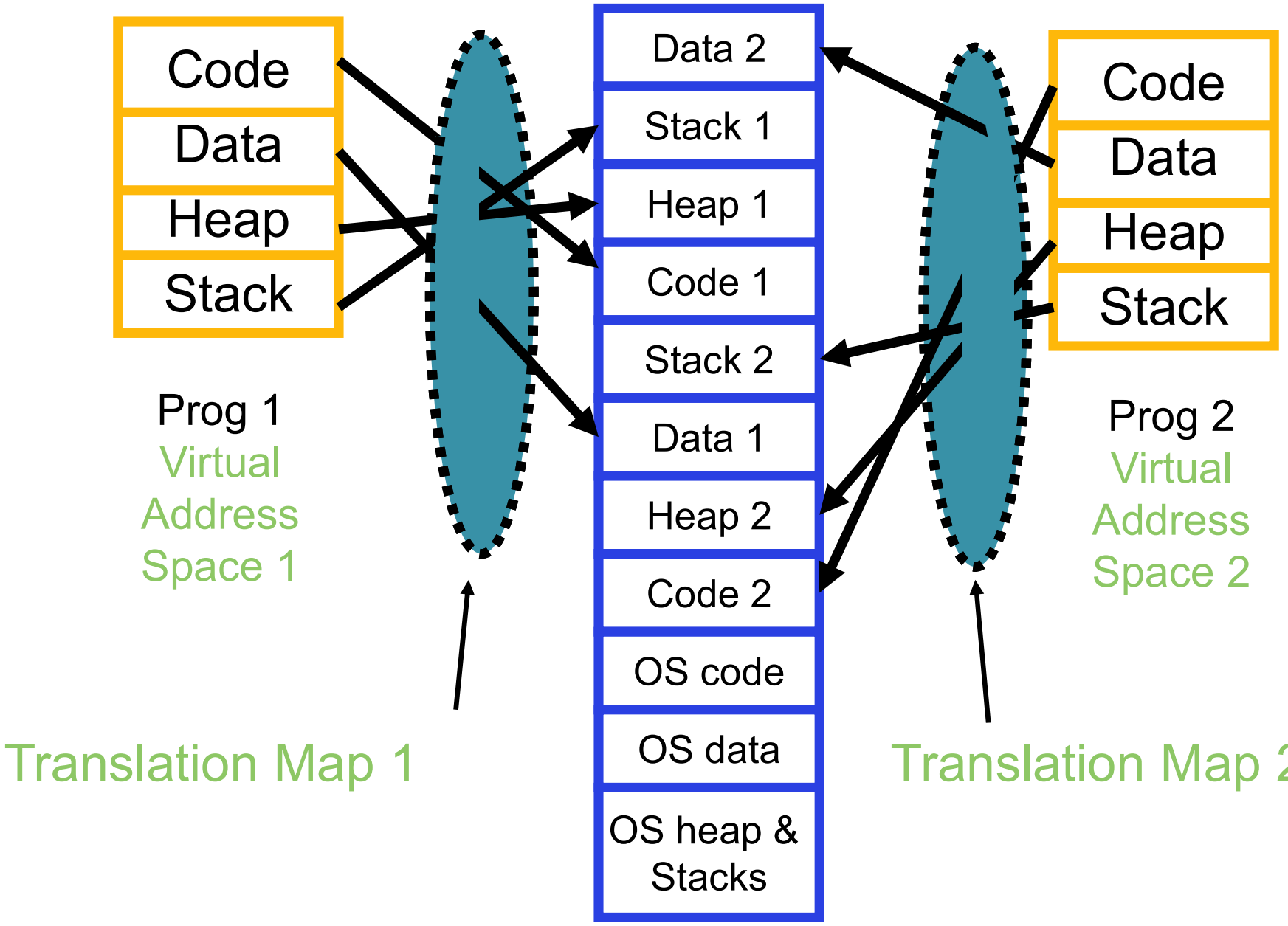
Simple Policy:

Programs are not allowed to read/write memory of other Programs or of Operating System

Mechanisms:

- Address Translation
- Dual Mode Operation

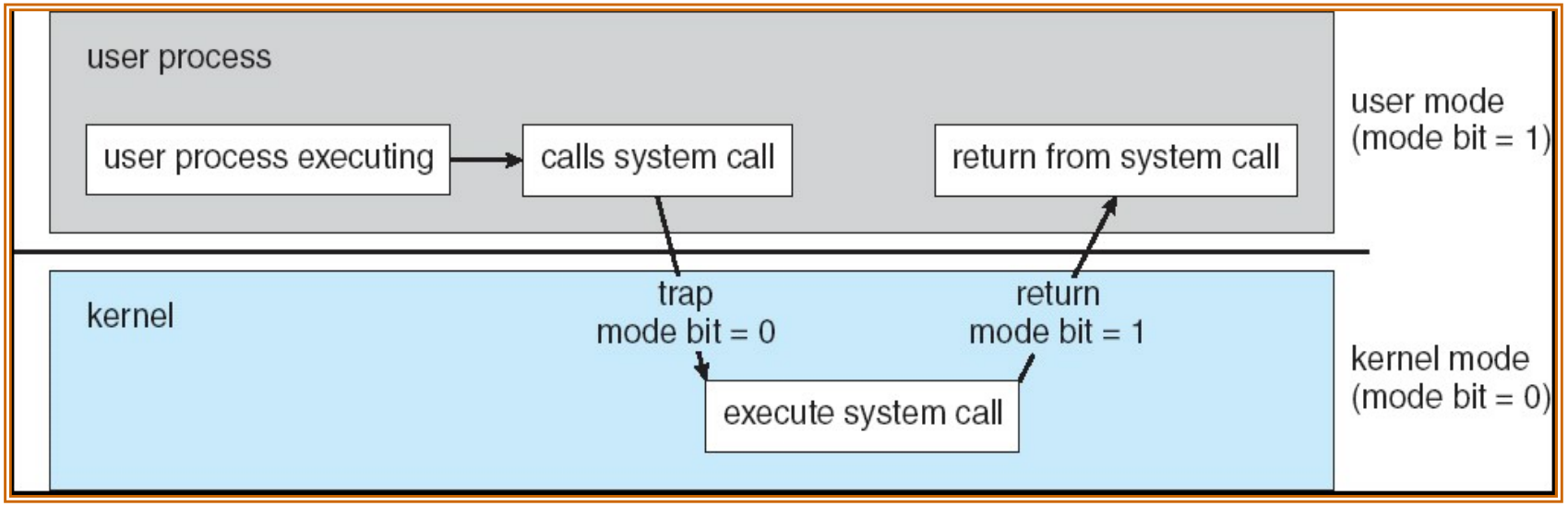
Example of Address Translation



Dual Mode Operation

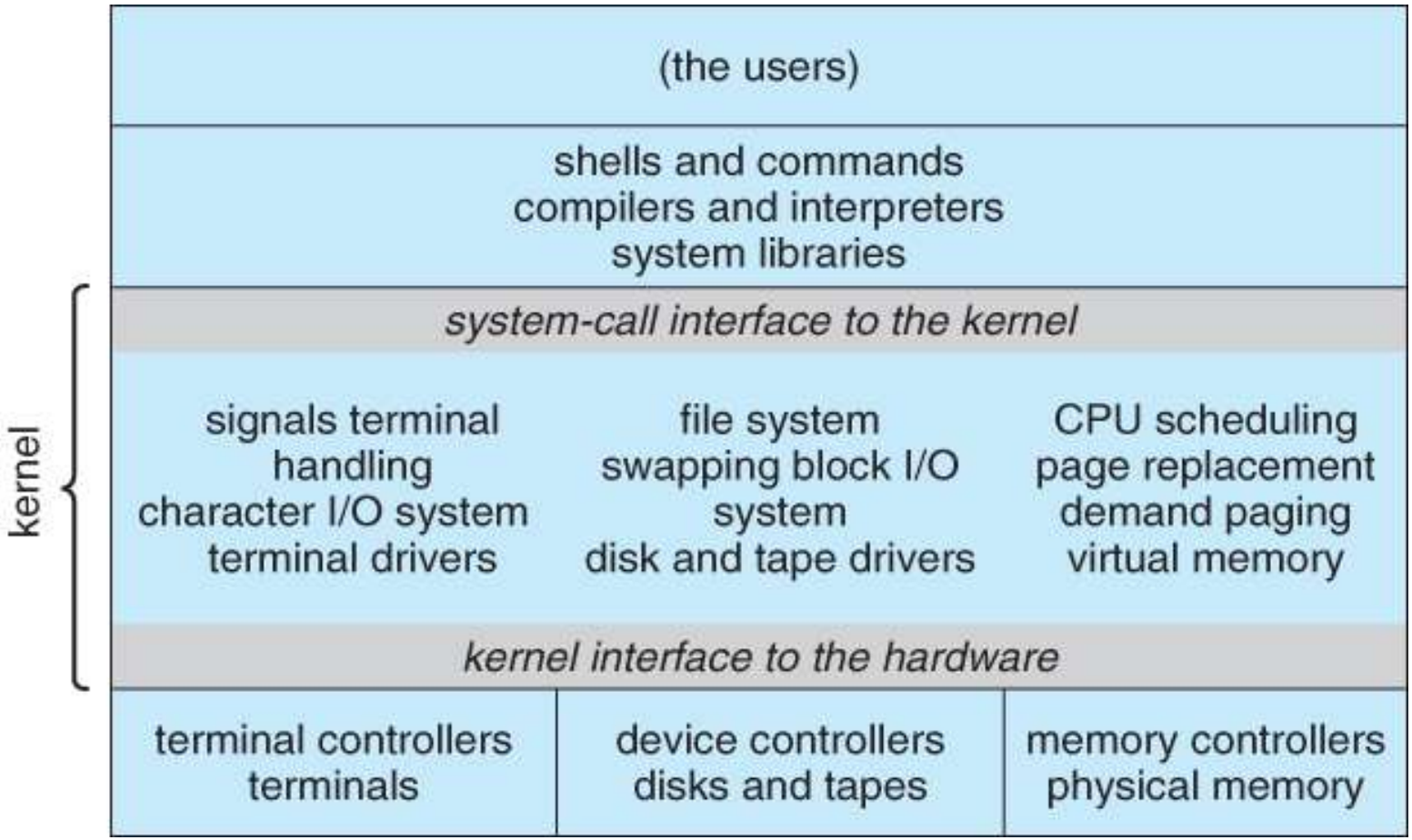
- Two modes:
  - "Kernel" mode (or "supervisor" or "protected")
  - "User" mode: Normal programs executed
- Some instructions/ops prohibited in user mode:
  - Example: cannot modify page tables in user mode

    Attempt to modify ⇒ Exception generated
- Transitions from user mode to kernel mode:
  - System Calls, Interrupts, Other exceptions



The OS Kernel

- The kernel is the central module of an operating system (OS).
- It is the part of the operating system that loads first, and it remains in main memory.
- Consists of everything below the system-call interface and above the physical hardware



User space vs Kernel space

- System memory can be divided into two distinct regions: kernel space and user space.

- Kernel space is where the kernel (i.e., the core of the operating system) executes (i.e., runs) and provides its services.

- User space is that set of memory locations in which user processes (i.e., everything other than the kernel) run.

Kernel Mode

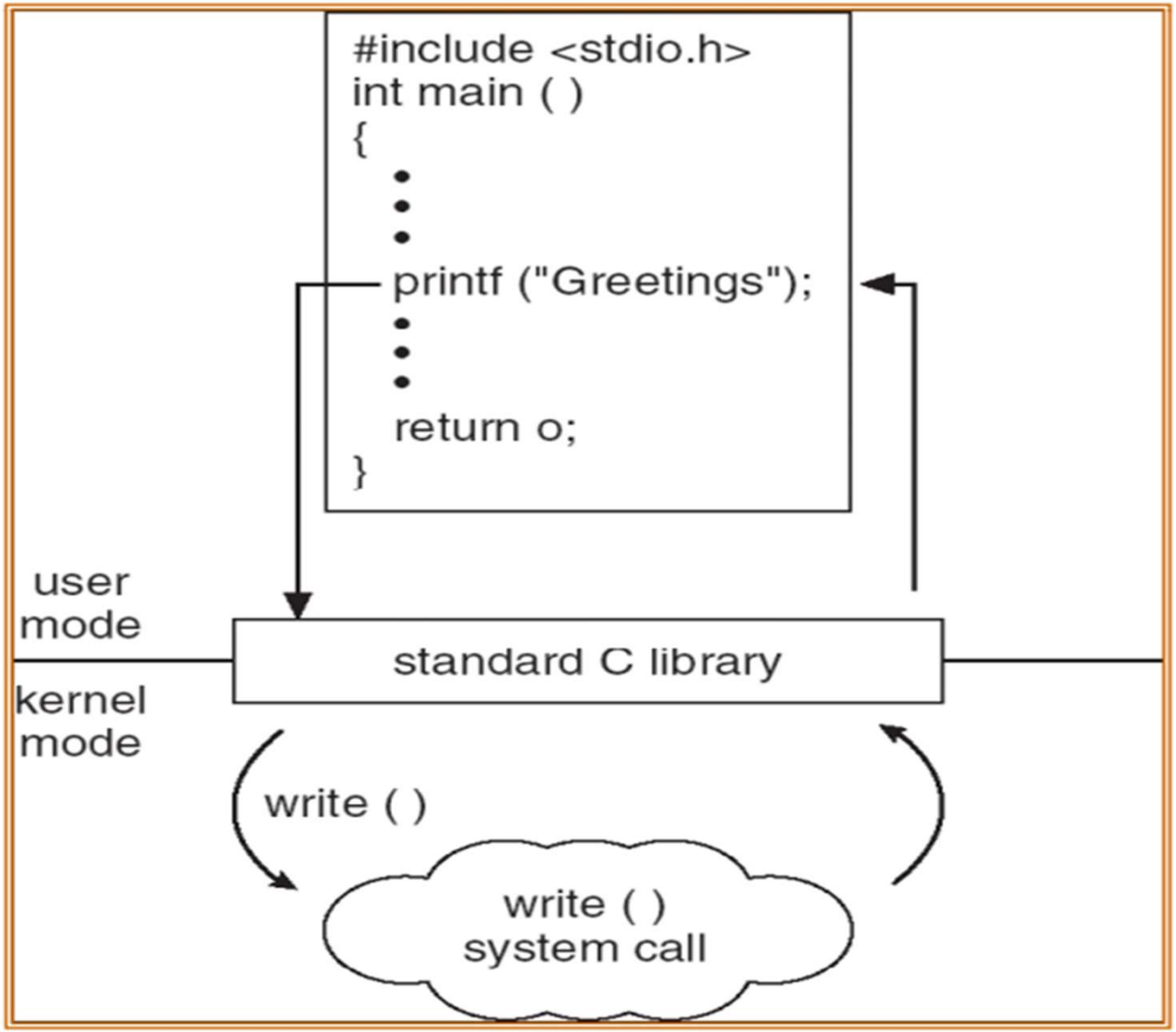
- Also called privileged mode, or as system mode
- Used mainly for Restriction/ Protection from unauthorized user application program
- When the CPU is in kernel mode, it is assumed to be executing trusted software, and thus it can execute any instructions and reference any memory addresses (i.e., locations in memory).

User Mode

- It is a non-privileged mode.
- It is non-privileged in that it is forbidden for processes in this mode to access those portions of memory (i.e., RAM) that have been allocated to the kernel or to other programs.
- User mode is the normal mode of operating for programs.
- Code running in user mode must delegate to system APIs to access hardware or memory.
- Most of the code running on your computer will execute in user mode.

System Call

Example



EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

Any Questions?

Contact:

prachip@cdac.in

Thank You!