SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS

(20CS0507)

COURSE OBJECTIVES:

1. Explain main components of an OS & their functions.

2. Describe the process management and scheduling.

3. Discuss various issues in Inter Process Communication (IPC) and the role of OS in IPC.

4. Illustrate the concepts and implementation of Memory management policies and virtual memory

5. Explain working of an OS as a resource manager, file system manager, process manager, memory manager and I/O manager and methods used to implement the different parts of OS.

COURSE OUTCOMES (CO's):

On successful completion of the course, the student will be able to

1. Describe the important computer system resources and the role of operating system in their management policies and algorithms.
2. Understand the process management policies and scheduling of processes by CPU.
3. Analyze the requirement for process synchronization and coordination handled by operating system.
4. Describe and analyze the memory management and its allocation policies. Technologies
5. Categorize the storage management policies with respect to different storage management technologies
6. Study the need for special purpose operating system with the advent of new emerging technologies

Topics to be Covered:

Operating Systems Overview:

What is an operating system-History of operating systems Operating system functions- Operating systems Operations- Types of Operating Systems and Computing Environments.

System Structures:

Operating System Services-User and Operating-System Interface systems calls- Types of System Calls- system programs- Structure of an OS - Layered Monolithic- Microkernel Operating Systems and Concept of Virtual Machine.

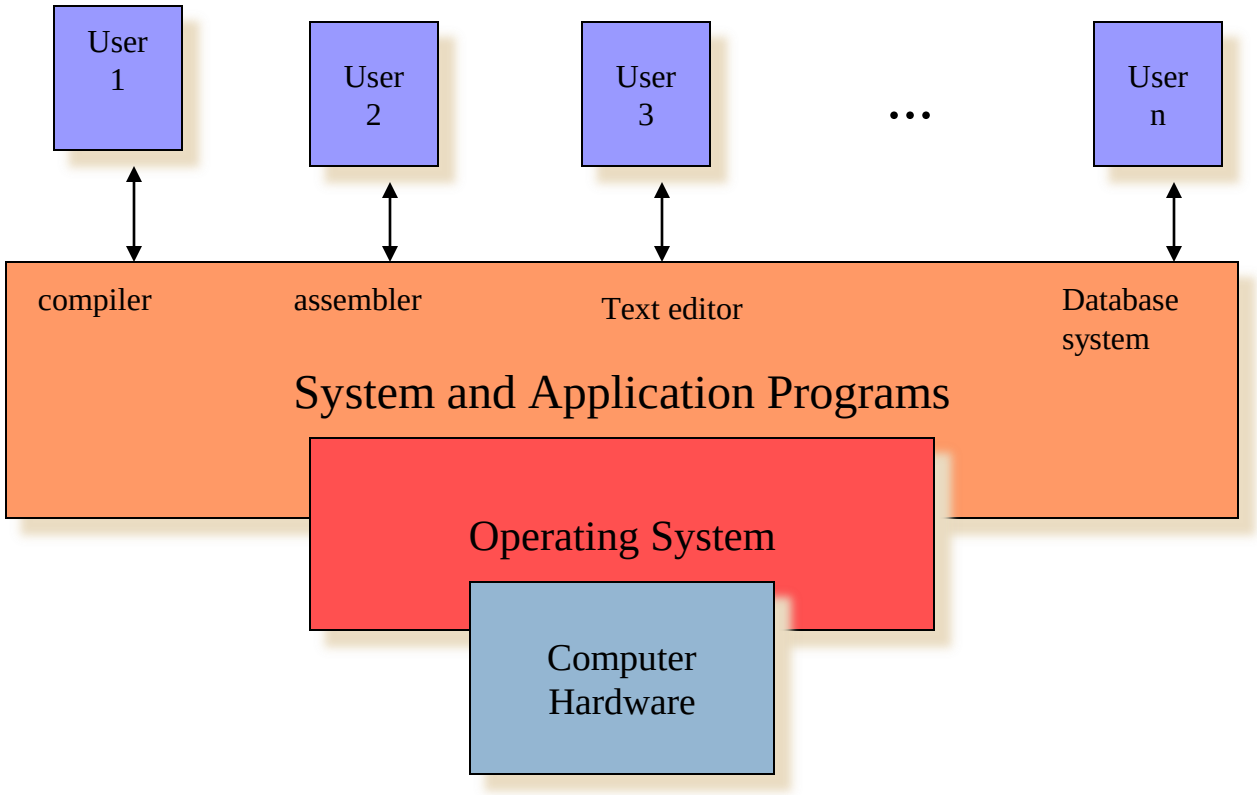
What is an Operating System?

- An OS is a program that acts an intermediary between the user of a computer and computer hardware.
- Major cost of general purpose computing is software.
  - OS simplifies and manages the complexity of running application programs efficiently.

Computer System Components

- Hardware
  - Provides basic computing resources (CPU, memory, I/O devices).
- Operating System
  - Controls and coordinates the use of hardware among application programs.
- Application Programs
  - Solve computing problems of users (compilers, database systems, video games, business programs such as banking software).
- Users
  - People, machines, other computers

Abstract View of System



Operating System Views

- Resource allocator
  - to allocate resources (software and hardware) of the computer system and manage them efficiently.
- Control program
  - Controls execution of user programs and operation of I/O devices.
- Kernel
  - The program that executes forever (everything else is an application with respect to the kernel).

Operating system roles

- **Referee**
  - Resource allocation among users, applications
  - Isolation of different users, applications from each other
  - Communication between users, applications
- **Illusionist**
  - Each application appears to have the entire machine to itself
  - Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport
- **Glue**
  - Libraries, user interface widgets, …
  - Reduces cost of developing software

Example: file systems

- Referee
  - Prevent users from accessing each other's files without permission
- Illusionist
  - Files can grow (nearly) arbitrarily large
  - Files persist even when the machine crashes in the middle of a save
- Glue
  - Named directories, printf, …

Goals of an Operating System

- Simplify the execution of user programs and make solving user problems easier.
- Use computer hardware efficiently.
  - Allow sharing of hardware and software resources.
- Make application software portable and versatile.
- Provide isolation, security and protection among user programs.
- Improve overall system reliability
    - error confinement, fault tolerance, reconfiguration.

Why should I study Operating Systems?

- Need to understand interaction between the hardware and applications
  - New applications, new hardware..
  - Inherent aspect of society today
- Need to understand basic principles in the design of computer systems
  - efficient resource management, security, flexibility
- Increasing need for specialized operating systems
  - e.g. embedded operating systems for devices - cell phones, sensors and controllers
  - real-time operating systems - aircraft control, multimedia services

History of Operating System

- The first true digital computer was designed by the English mathematician Charles Babbage (1792-1871). Although Babbage spent most of his life and fortune trying to build his “analytical engine.” he never got it working properly because it was purely mechanical, and the technology of his day could not produce the required wheels, gears, and cogs to the high precision that he needed.

History of Operating System

Generations

- The First Generation (1945-55) Vacuum Tubes and Plug boards
- The Second Generation (1955-65) Transistors and Batch Systems
- The Third Generation (1965-1980) ICs and Multiprogramming
- The Fourth Generation (1980-2010) Personal Computers
- The Fifth Generation (2010 to present) – Artificial Intelligence

Operating System Functions

Functions of Operating Systems:

- Process Management
- Memory Management
- Device or I/O Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Operating Systems Operations

Operating system contains two modes of operations:

- **User mode**
- **Kernel mode**

- **User mode:** it can execute the programs without accessing memory, hardware's etc., i.e when the computer system is executing on behalf of the user applications then the system is in user mode.it represent with mode bit '1'

Operating Systems Operations

Kernel mode:

- it will execute the programs with direct accessing of memory, hardware's etc., it represent with mode bit '0'. It is also called supervisor mode or system mode or privileged mode.
- When a user application requests a service from the operating system where resources available(via system call), it must transition from user to kernel mode to fulfill the request.

Types of Operating System

Types of Operating System

- 1. Batch operating system
- 2. Real-time
- 3. Multi-user vs. Single-user
- 4. Multi-tasking vs. Single-tasking
- 5. Single-processor Systems
- 6. Multi-processor Systems
- 7. Distributed:
- 8. Embedded

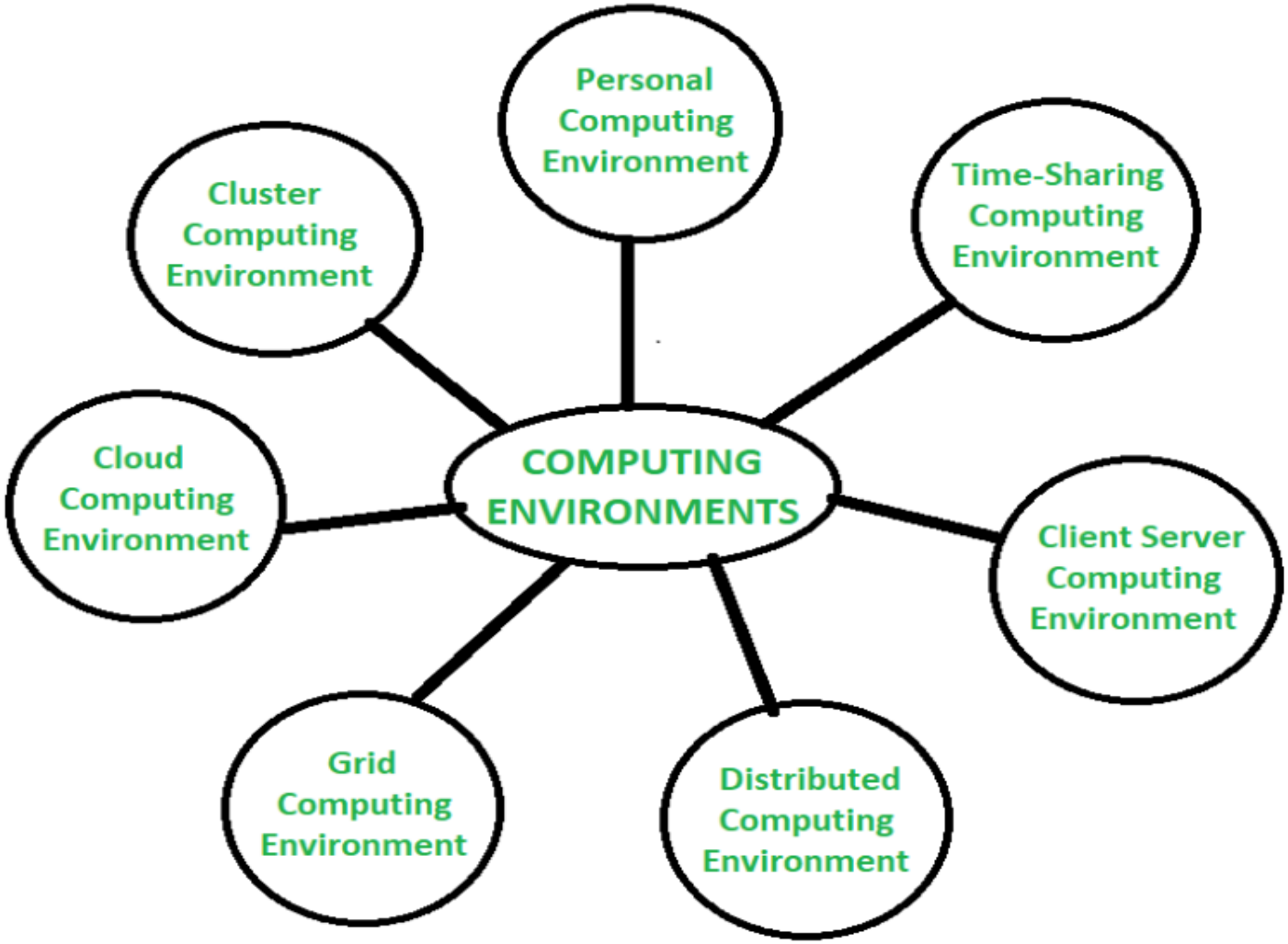
Computing Environments

When a problem is solved by the computer, during that time computer uses many devices, arranged in different ways and which work together to solve problems. This constitutes a computing environment where various number of computer devices arranged in different ways to solve different types of problems in different ways.

- **Types of Computing Environments:**

The are the various types of computing environments. They are :

Computing Environments



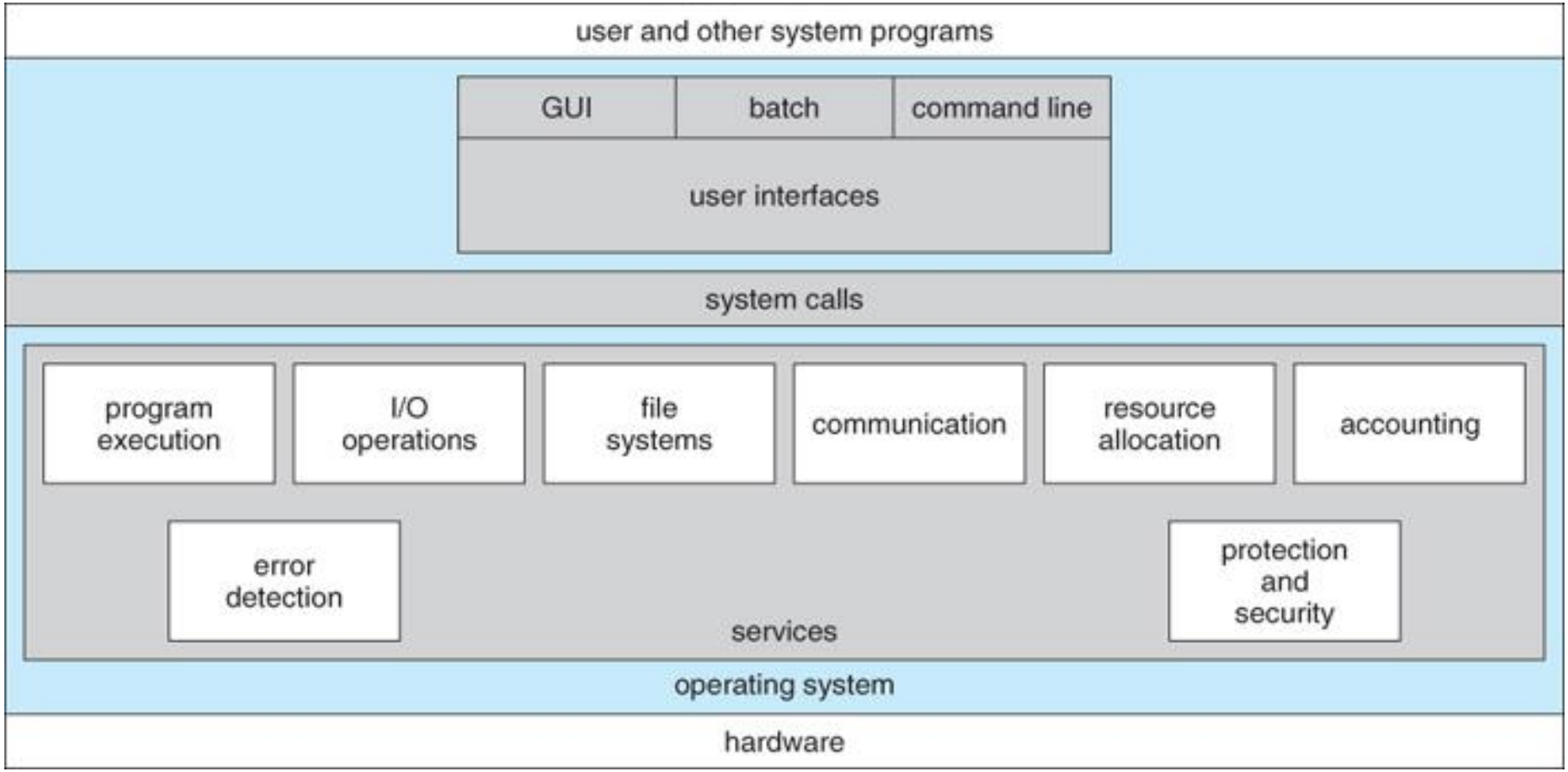
Operating System provides services:

An Operating System provides services to programs and to the users of programs. It provides an environment to execute programs. It provides services to the user to execute the programs in a convenient manner.

Following are a few common services provided by an operating system −

1. Program execution
2. I/O operations
3. File System manipulation
4. Communication
5. Error Detection
6. Resource Allocation
7. Protection

Operating System provides services:



Operating-System Interface

Command Interpreter

- Gets and processes the next user request, and launches the requested programs.
- In some systems the CI may be incorporated directly into the kernel.
- More commonly the CI is a separate program that launches once the user logs in or otherwise accesses the system.

Operating-System Interface

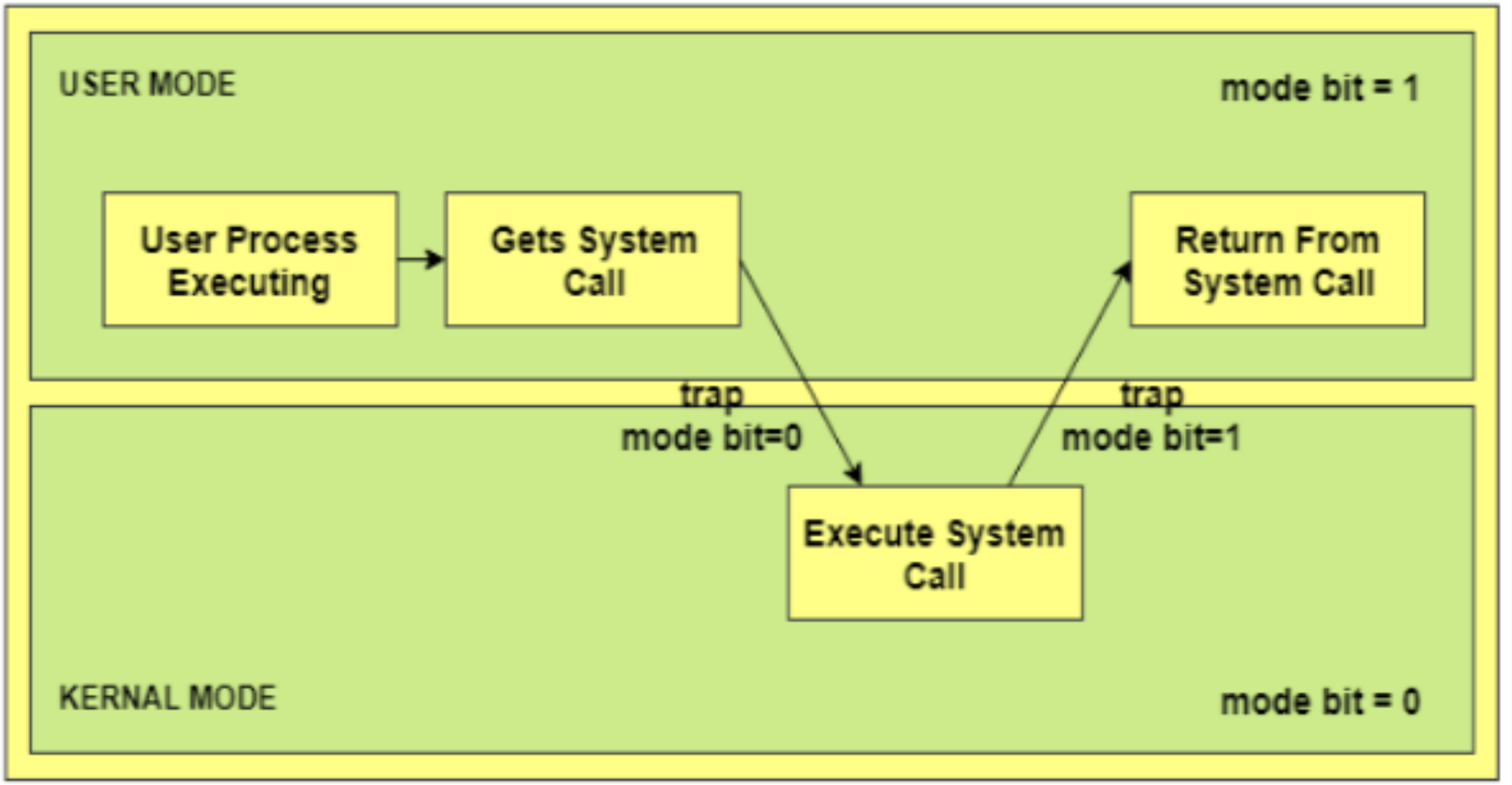
Graphical User Interface, GUI

- Generally implemented as a desktop metaphor, with file folders, trash cans, and resource icons.
- Icons represent some item on the system, and respond accordingly when the icon is activated.
- First developed in the early 1970's at Xerox PARC research facility.
- Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)

System Call:

- **System calls** provide a mechanism for user or application programs to call upon the services of the operating system.
- As we know that for performing any Operation as user must have to specify the Operation which he wants to Operate on the Computer. We can say that For Performing any Operation a user must have to Request for a Service from the System. For Making any Request, a user will prepare a Special call which is also known as the **System Call**.

System Call:


USER MODE
mode bit = 1
User Process Executing
Gets System Call
Return From System Call
trap
mode bit=0
trap
mode bit=1
Execute System Call
KERNAL MODE
mode bit = 0

System Call:

Example System Call Sequence

Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally

Types of System Calls:

- Process control
  - end, abort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
- File management
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes
- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get process, file, or device attributes
  - set process, file, or device attributes
- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices

Figure 2.8 Types of system calls.

System programs:

System programs provide a convenient environment for program development and execution. Some of them are simply user interfaces to system calls; others are considerably more complex.

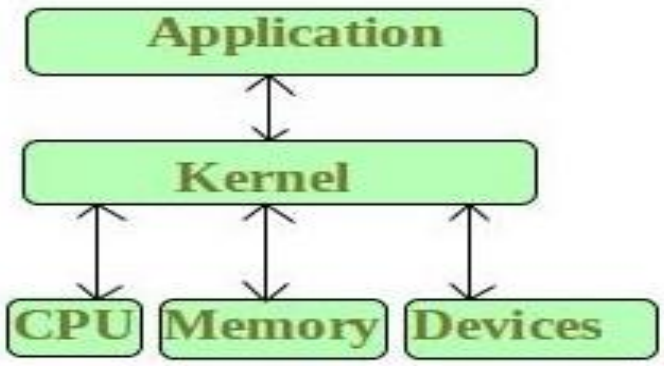
They can be divided into these categories:

- File management
- Status information
- File modification
- Programming-language support.
- Program loading and execution
- Communications

Operating System Structure

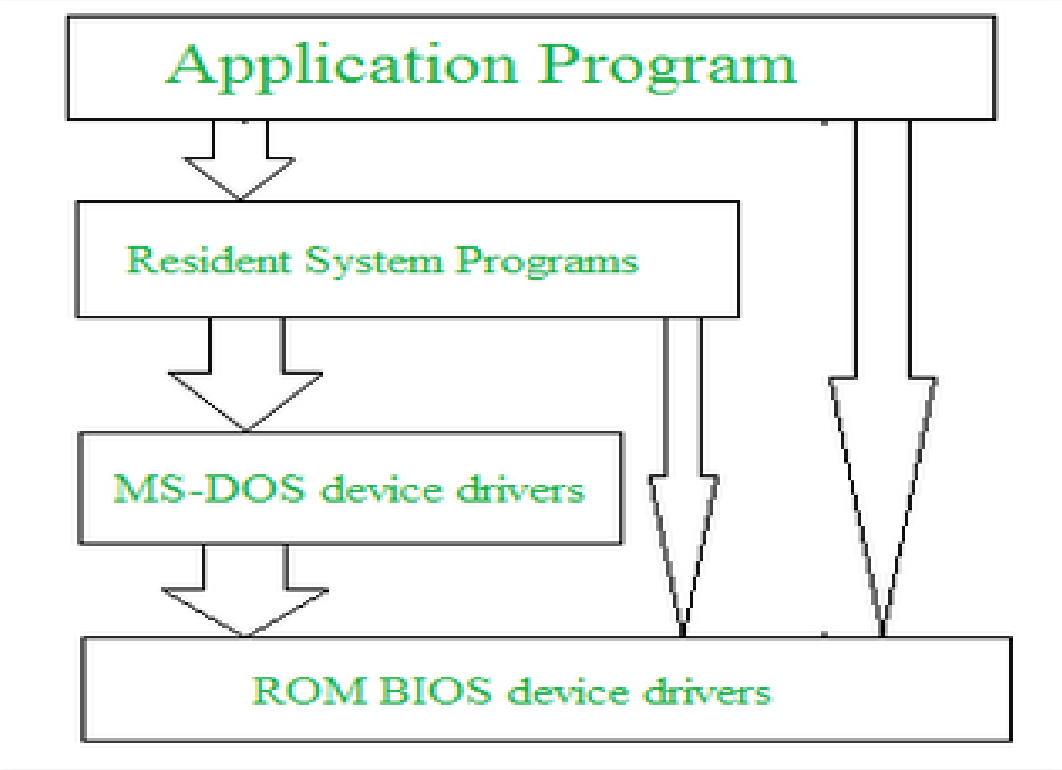
Kernel

- A kernel is an important part of an OS that manages system resources. It also acts as a bridge between the software and hardware of the computer.



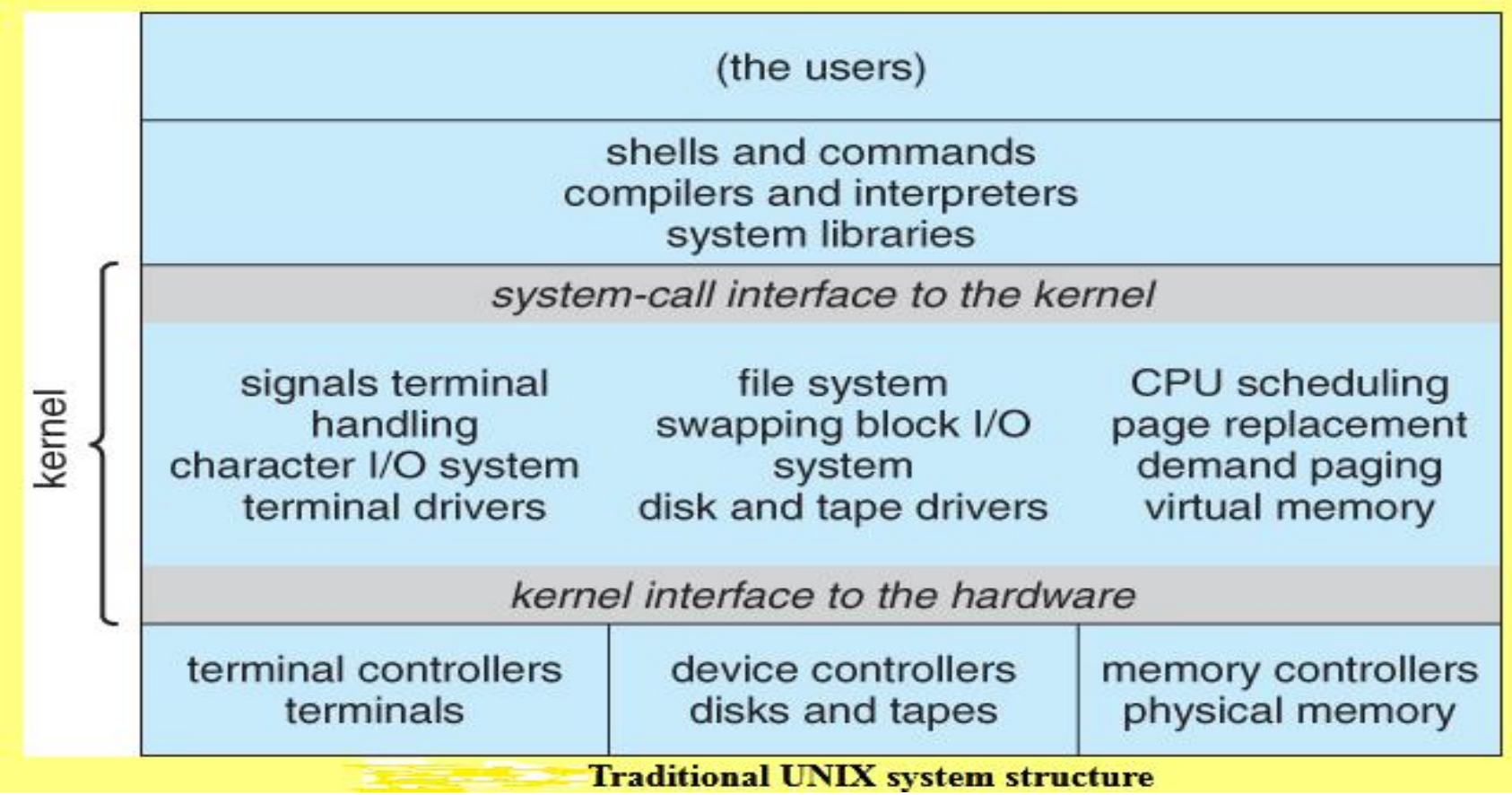
Operating System Structure

- **Simple structure :**



Operating System Structure

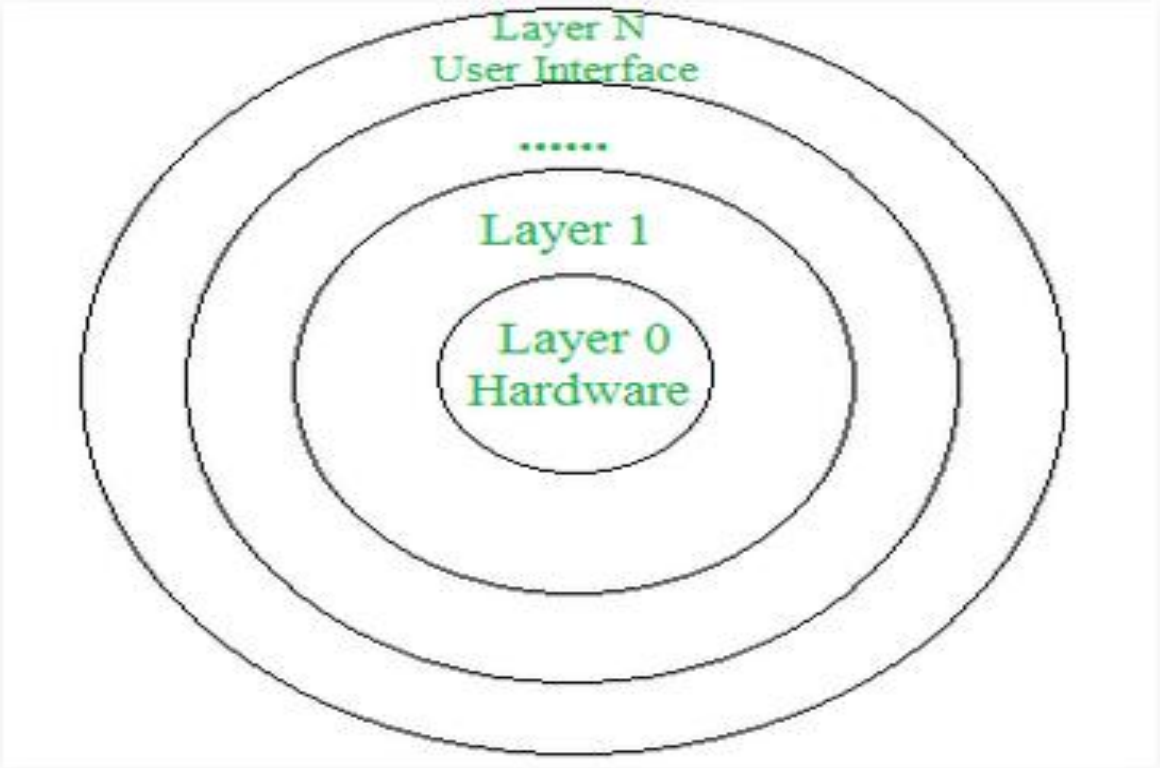
- **Monolithic kernel:**



Traditional UNIX system structure

Operating System Structure

- **Layered structure:**



Personal Computing Systems

Hardware – *cheap* ; Human – *expensive*

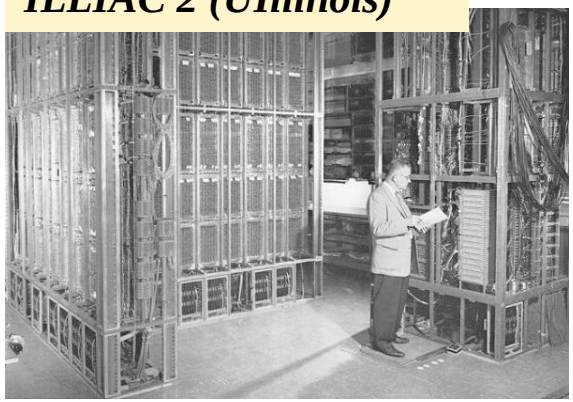
- Single user systems, portable.
- I/O devices - keyboards, mice, display screens, small printers.
- Laptops and palmtops, Smart cards, Wireless devices.
- Single user systems may not need advanced CPU utilization or protection features.
- Advantages:
  - user convenience, responsiveness, ubiquitous

Parallel Systems

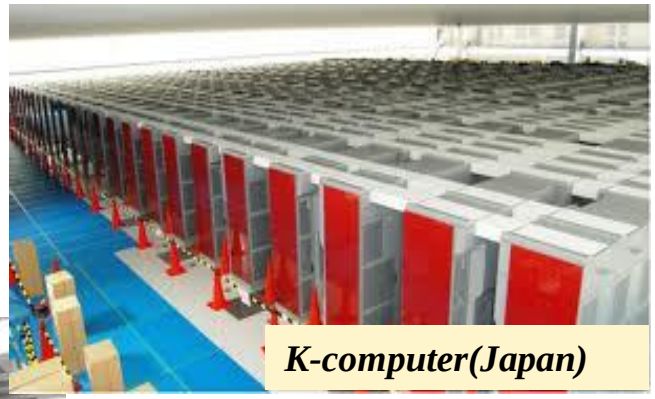
- Multiprocessor systems with more than one CPU in close communication.
- Improved Throughput, economical, increased reliability.
- Kinds:
  - Vector and pipelined
  - Symmetric and asymmetric multiprocessing
  - Distributed memory vs. shared memory
- Programming models:
  - Tightly coupled vs. loosely coupled ,message-based vs. shared variable

Parallel Computing Systems

ILLIAC 2 (UIllinois)



Climate modeling, earthquake simulations, genome analysis, protein folding, nuclear fusion research, …..



K-computer(Japan)

Tianhe-1(China)

Connection Machine (MIT)

IBM Blue Gene

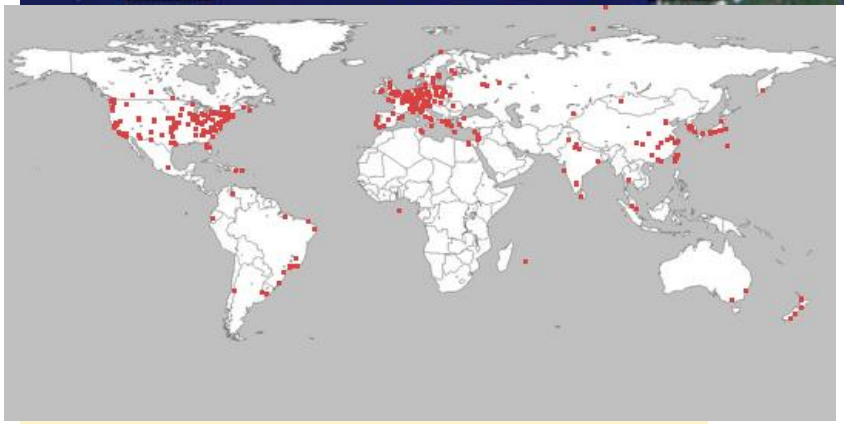
Distributed Systems

Hardware – ***very cheap*** ; **Human** – ***very expensive***

- Distribute computation among many processors.
- Loosely coupled -
  - no shared memory, various communication lines
- client/server architectures
- Advantages:
  - resource sharing
  - computation speed-up
  - reliability
  - communication - e.g. email
- Applications - digital libraries, digital multimedia

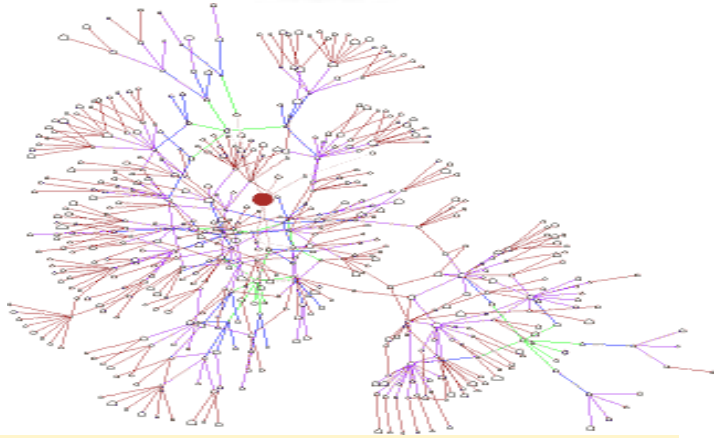
Distributed Computing Systems

Globus Grid Computing Toolkit



PlanetLab

Cloud Computing Offerings



Gnutella P2P Network

Real-time systems

- Correct system function depends on timeliness
- Feedback/control loops
- Sensors and actuators
- Hard real-time systems -
  - Failure if response time too long.
  - Secondary storage is limited
- Soft real-time systems -
  - Less accurate if response time is too long.
  - Useful in applications such as multimedia, virtual reality.

A personal computer today

interaction

- Super AMOLED display
- Capacitive touchscreen (multitouch)
- Audio (speaker, microphone)
- Vibration
- S pen

- 4G LTE
- NFC
- WiFi
- Bluetooth
- Infrared

- 64 GB internal storage (extended by microSD)
- Adreno 330 GPU
- Hexagon DSP
- Multimedia processor

- 13 MP front camera
- 2 MP back camera
- Accelerometer
- Gyroscope
- Proximity sensor
- Compass
- Barometer
- Temperature sensor
- Humidity sensor
- Gesture Sensor
- GPS

A personal computer today

- Super AMOLED display
- Capacitive touchscreen (multitouch)
- Audio (speaker, microphone)
- Vibration
- S pen

- 4G LTE
- NFC
- WiFi
- Bluetooth
- Infrared

- 64 GB internal storage (extended by microSD)
- Adreno 330 GPU
- Hexagon DSP
- Multimedia processor

- 13 MP front camera
- 2 MP back camera
- Accelerometer
- Gyroscope
- Proximity sensor
- Compass
- Barometer
- Temperature sensor
- Humidity sensor
- Gesture Sensor
- GPS

sensing

A personal computer today

- Super AMOLED display
- Capacitive touchscreen (multitouch)
- Audio (speaker, microphone)
- Vibration
- S pen

- 4G LTE
- NFC
- WiFi
- Bluetooth
- Infrared

connectivity

- 64 GB internal storage (extended by microSD)
- Adreno 330 GPU
- Hexagon DSP
- Multimedia processor

- 13 MP front camera
- 2 MP back camera
- Accelerometer
- Gyroscope
- Proximity sensor
- Compass
- Barometer
- Temperature sensor
- Humidity sensor
- Gesture Sensor
- GPS

A personal computer today

- Super AMOLED display
- Capacitive touchscreen (multitouch)
- Audio (speaker, microphone)
- Vibration
- S pen

- 4G LTE
- NFC
- WiFi
- Bluetooth
- Infrared

- 64 GB internal storage (extended by microSD)
- Adreno 330 GPU
- Hexagon DSP *acceleration*
- Multimedia processor

- 13 MP front camera
- 2 MP back camera
- Accelerometer
- Gyroscope
- Proximity sensor
- Compass
- Barometer
- Temperature sensor
- Humidity sensor
- Gesture Sensor
- GPS

Operating systems are everywhere

Operating systems are everywhere

Info-tainment!!

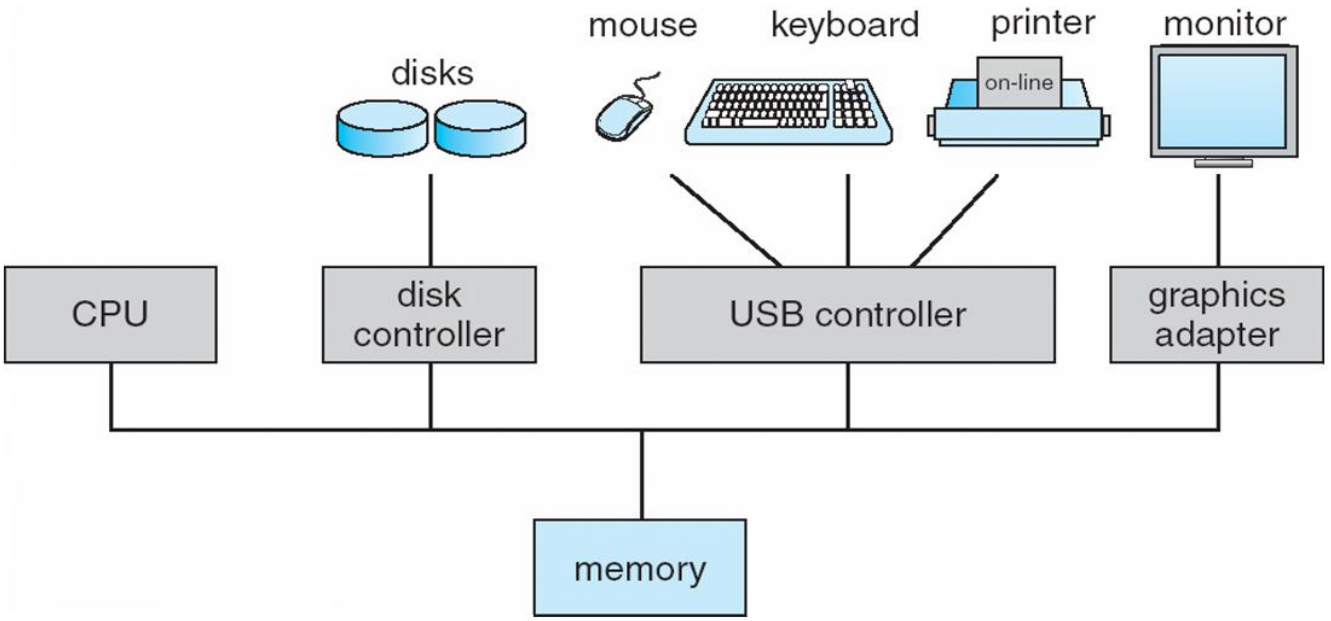
Summary of lecture

- What is an operating system?
- Early Operating Systems
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time-sharing Systems
- Personal Computer Systems
- Parallel and Distributed Systems
- Real-time Systems

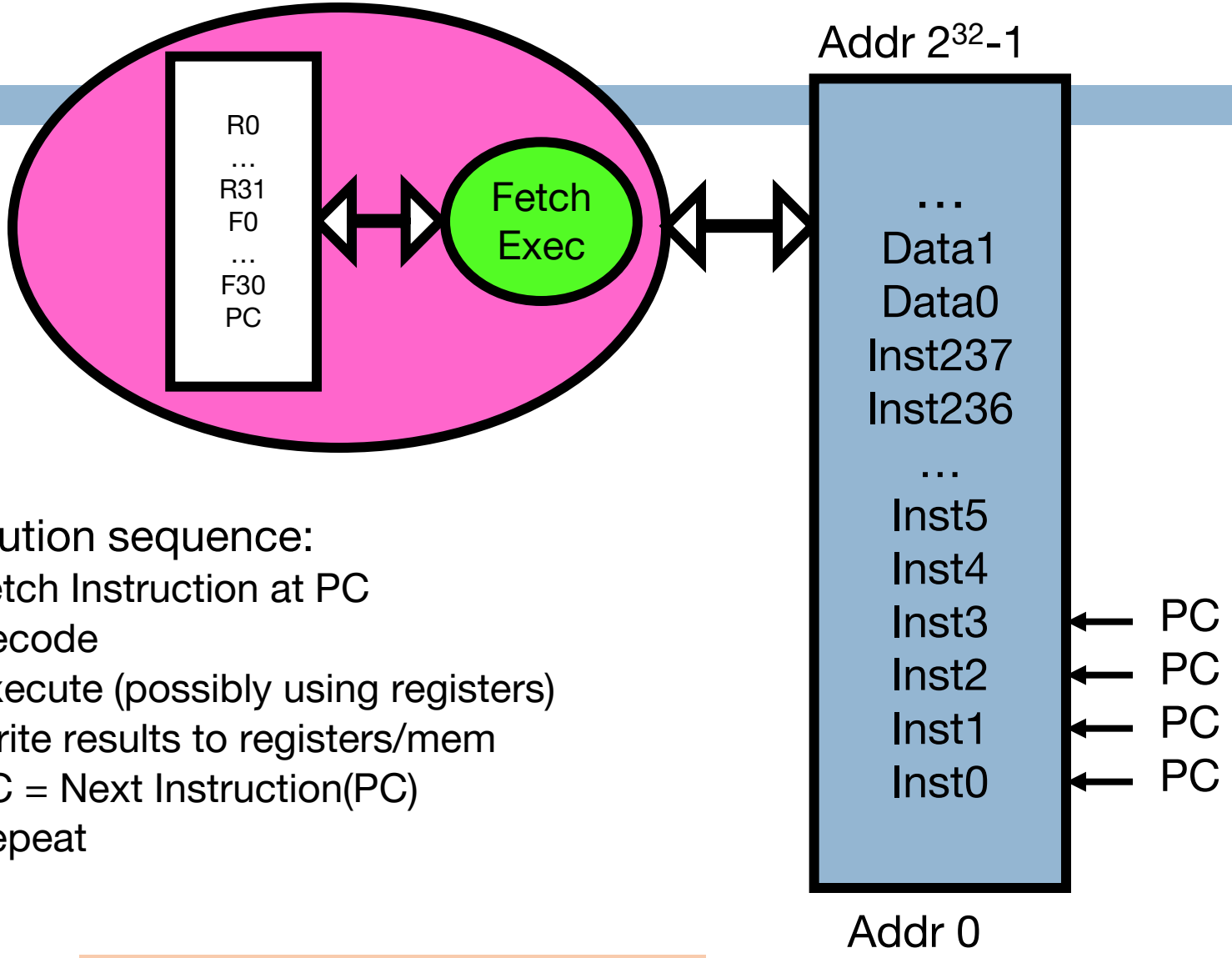
Computer System & OS Structures

- Computer System Organization
- Operational Flow and hardware protection
- System call and OS services
- Storage architecture
- OS organization
- OS tasks
- Virtual Machines

Computer System **Organization**



CPU execution

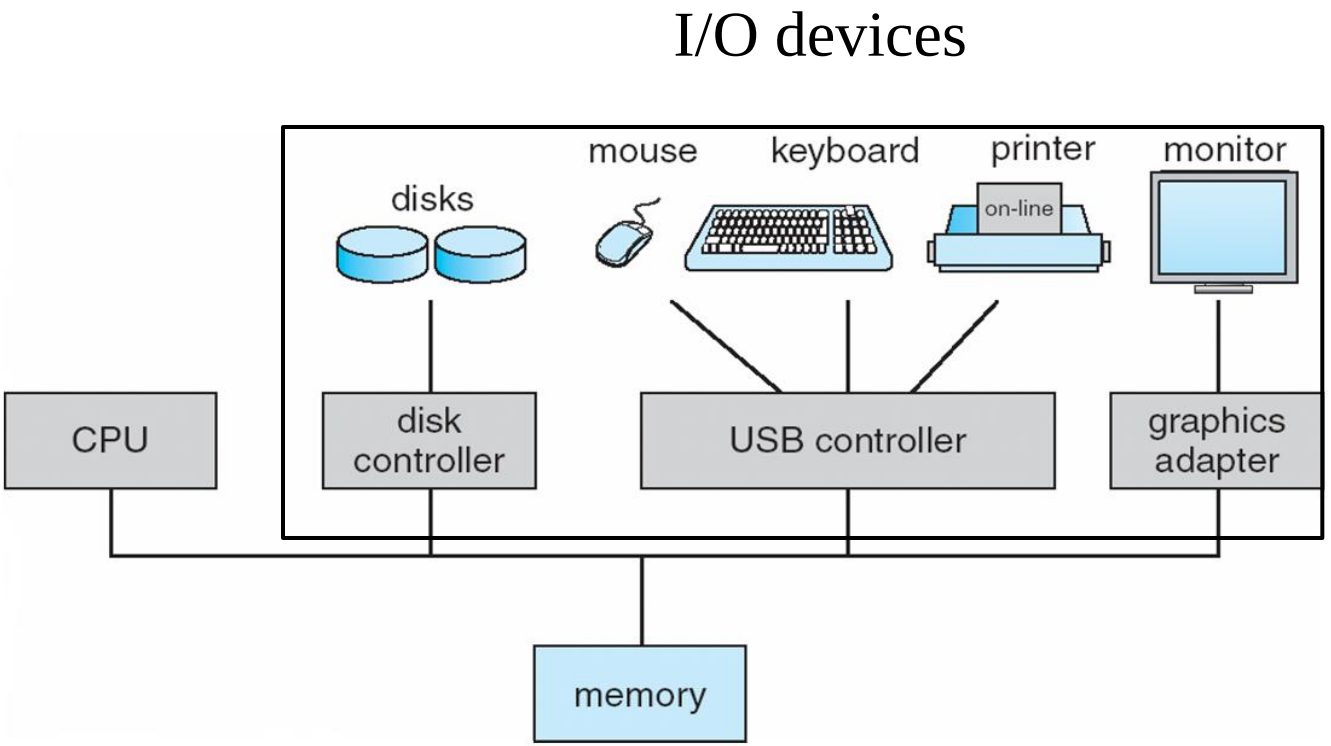


Execution sequence:

- Fetch Instruction at PC
- Decode
- Execute (possibly using registers)
- Write results to registers/mem
- PC = Next Instruction(PC)
- Repeat

From Berkeley OS course

Computer System **Organization**



I/O devices

- I/O devices and the CPU execute concurrently.
- Each device controller is in charge of a particular device type
  - Each device controller has a local buffer. I/O is from the device to local buffer of controller
- CPU moves data from/to main memory to/from the local buffers

Interrupts

- **Interrupt** transfers control to the **interrupt service routine**
  - Interrupt Service Routine: Segments of code that determine action to be taken for interrupt.
- Determining the type of interrupt
  - Polling: same interrupt handler called for all interrupts, which then polls all devices to figure out the reason for the interrupt
  - Interrupt Vector Table: different interrupt handlers will be executed for different interrupts

Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh

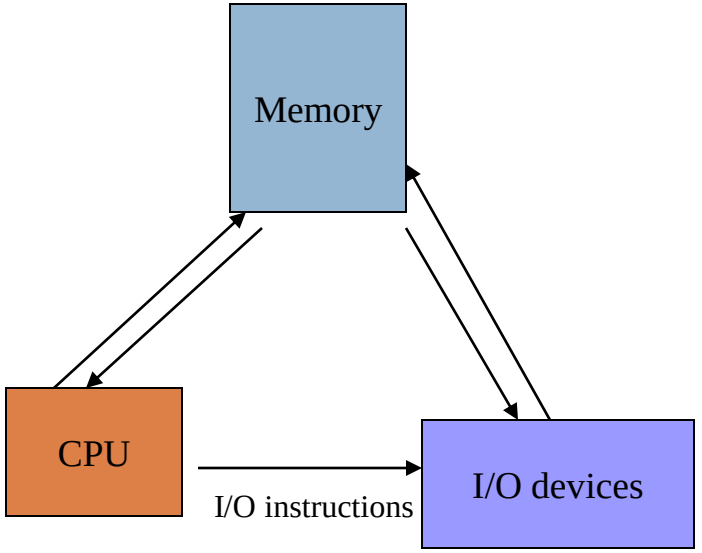
Interrupt Number	Address
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

Interrupt **handling**

- OS preserves the state of the CPU
  - stores registers and the program counter (address of interrupted instruction).
- What happens to a new interrupt when the CPU is handling one interrupt?
  - Incoming interrupts can be disabled while another interrupt is being processed. In this case, incoming interrupts may be lost or may be buffered until they can be delivered.
  - Incoming interrupts can be masked (i.e., ignored) by software.
  - Incoming interrupts are delivered, i.e., nested interrupts.

Direct Memory Access (DMA)

- Typically used for I/O devices with a lot of data to transfer (in order to reduce load on CPU).
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Device controller interrupts CPU on completion of I/O



Process Abstraction

Process Abstraction

- Process: an *instance* of a program, running with limited rights

Process Abstraction and rights

- Process: an *instance* of a program, running with limited rights
- Address space: set of rights of a process
  - Memory that the process can access
- Other permissions the process has (e.g., which system calls it can make, what files it can access)

Hardware Protection

- CPU Protection:
  - Dual Mode Operation
  - Timer interrupts
- Memory Protection
- I/O Protection

How to limit process rights?

Should a process be able to execute any instructions?

Should a process be able to execute any instructions?

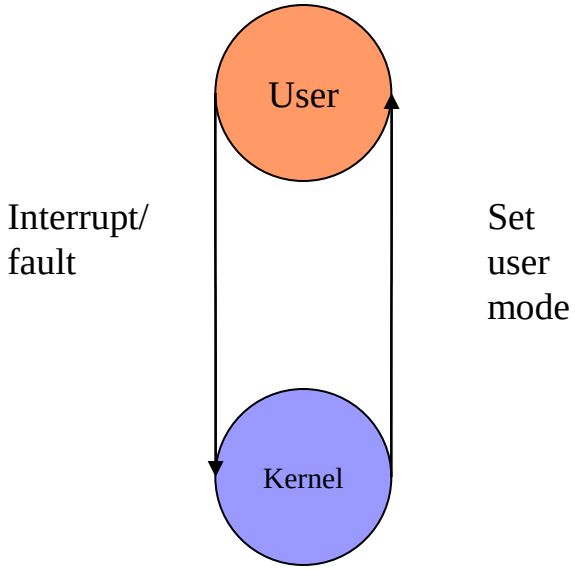
- No
  - Can alter system configuration
  - Can access unauthorized memory
  - Can access unauthorized I/O
  - etc.
- How to prevent?

Dual-mode operation

- Provide hardware support to differentiate between at least two modes of operation:
  1. User mode -- execution done on behalf of a user.
  2. Kernel mode (monitor/supervisor/system mode) -- execution done on behalf of operating system.
- "Privileged" instructions are only executable in the kernel mode
- Executing privileged instructions in the user mode "traps" into the kernel mode

Dual-mode operation(cont.)

- Mode bit added to computer hardware to indicate the current mode: kernel(0) or user(1).
- When an interrupt or trap occurs, hardware switches to kernel mode.



CPU Protection

- How to prevent a process from executing indefinitely?

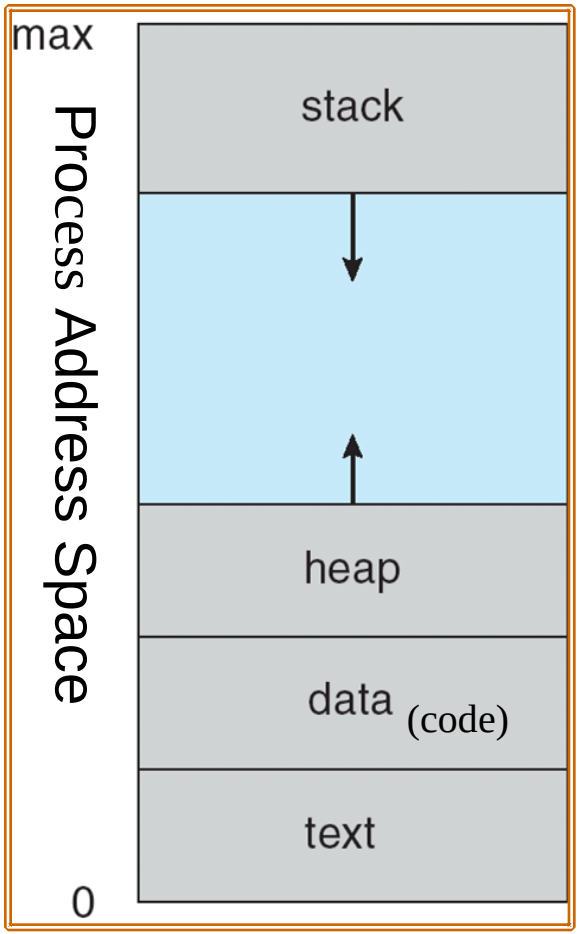
CPU Protection

- Timer - interrupts computer after specified period to ensure that OS maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches a value of 0, an interrupt occurs.
- Timer is commonly used to implement time sharing.
- Timer is also used to compute the current time.
- Programming the timer can only be done in the kernel since it requires privileged instructions.

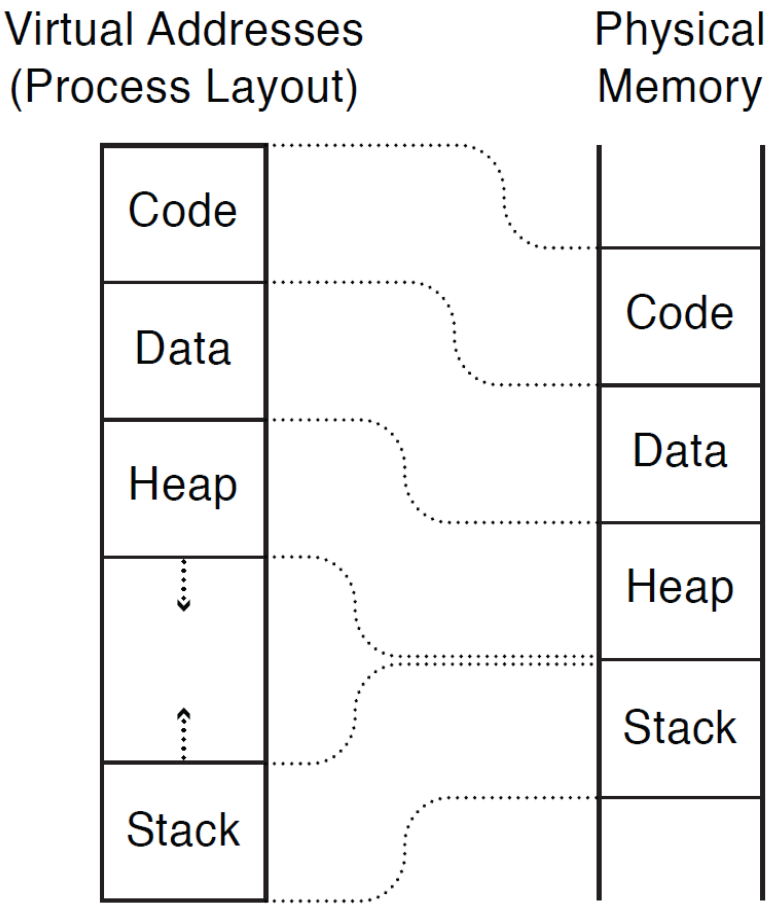
How to isolate memory access?

Process address space

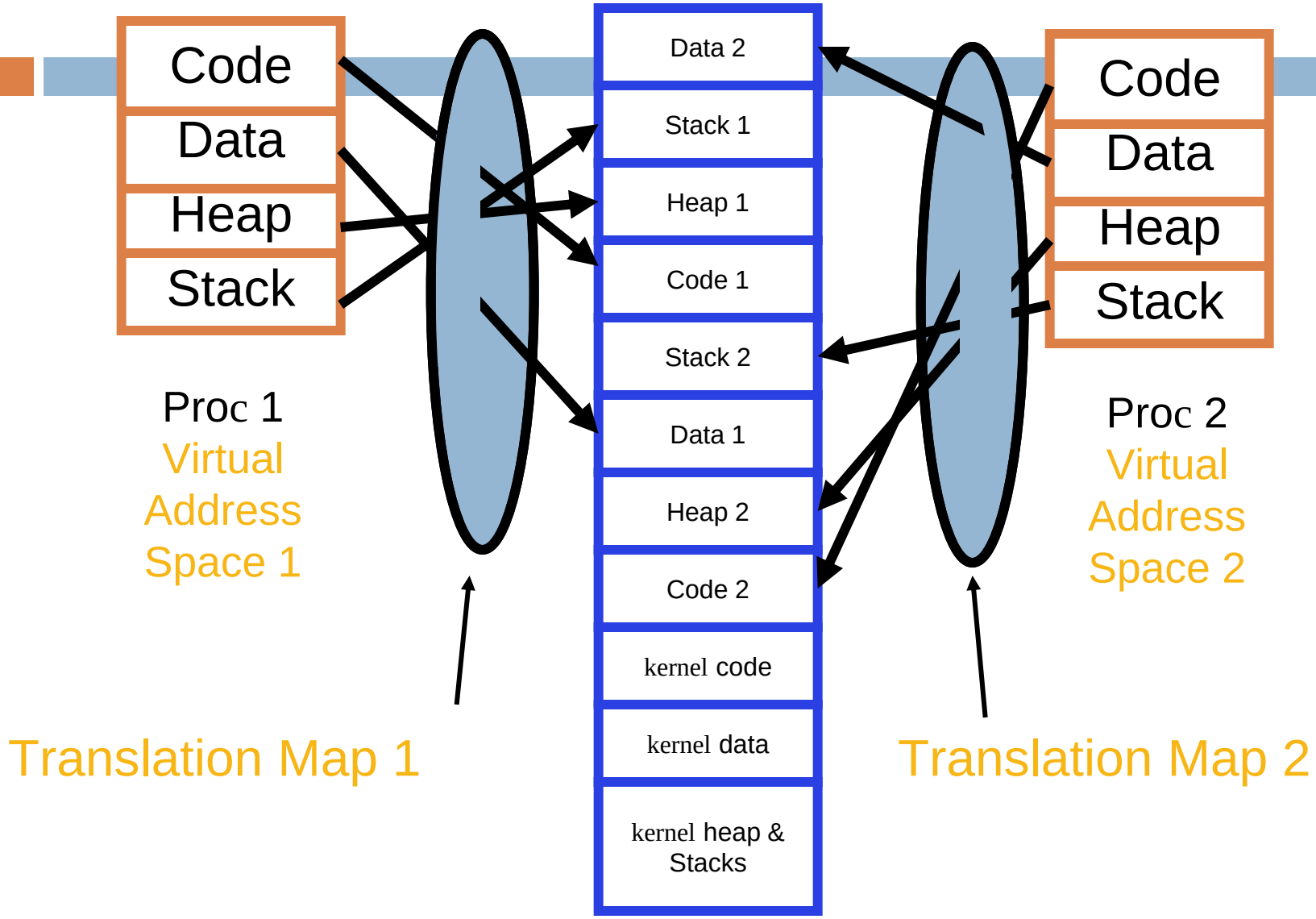
- Address space ⇒ the set of accessible addresses :
  - For a 32-bit processor there are 2^{32} = 4 billion addresses



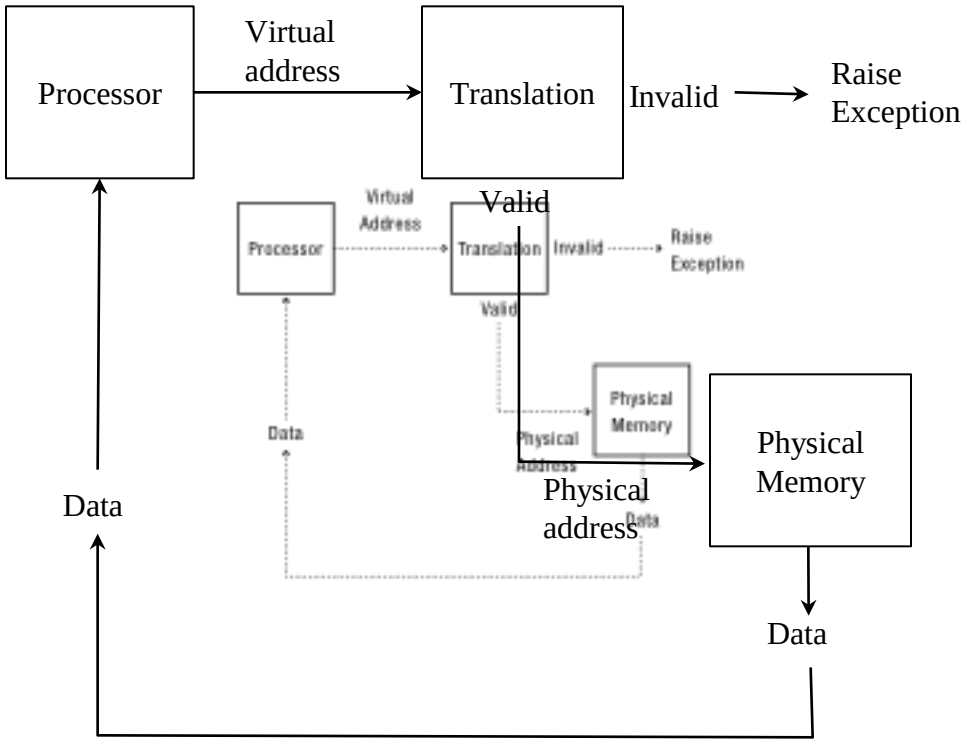
Virtual **Address**



Providing the Illusion of Separate Address Spaces



Address translation and memory protection

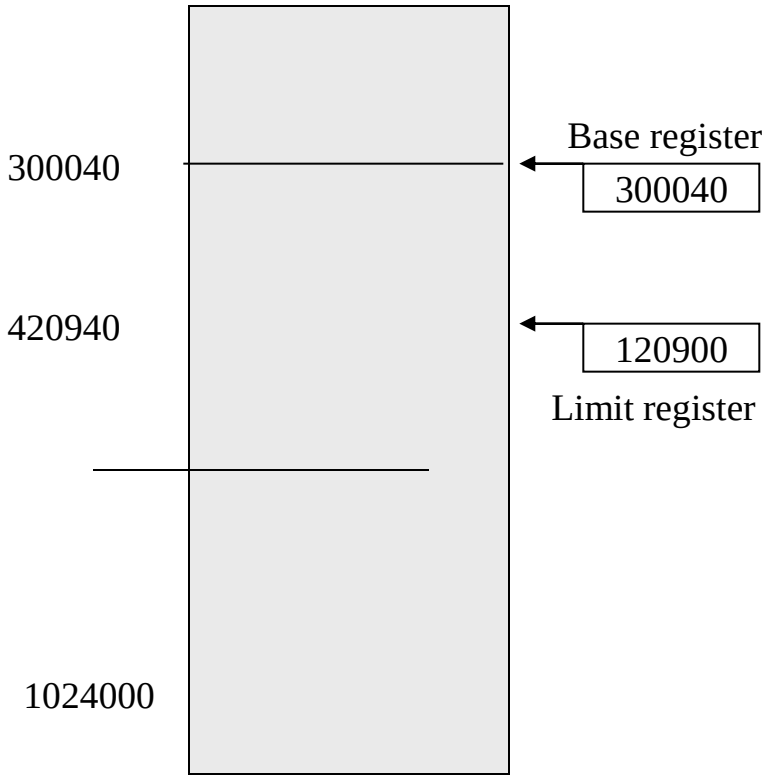


Memory Protection

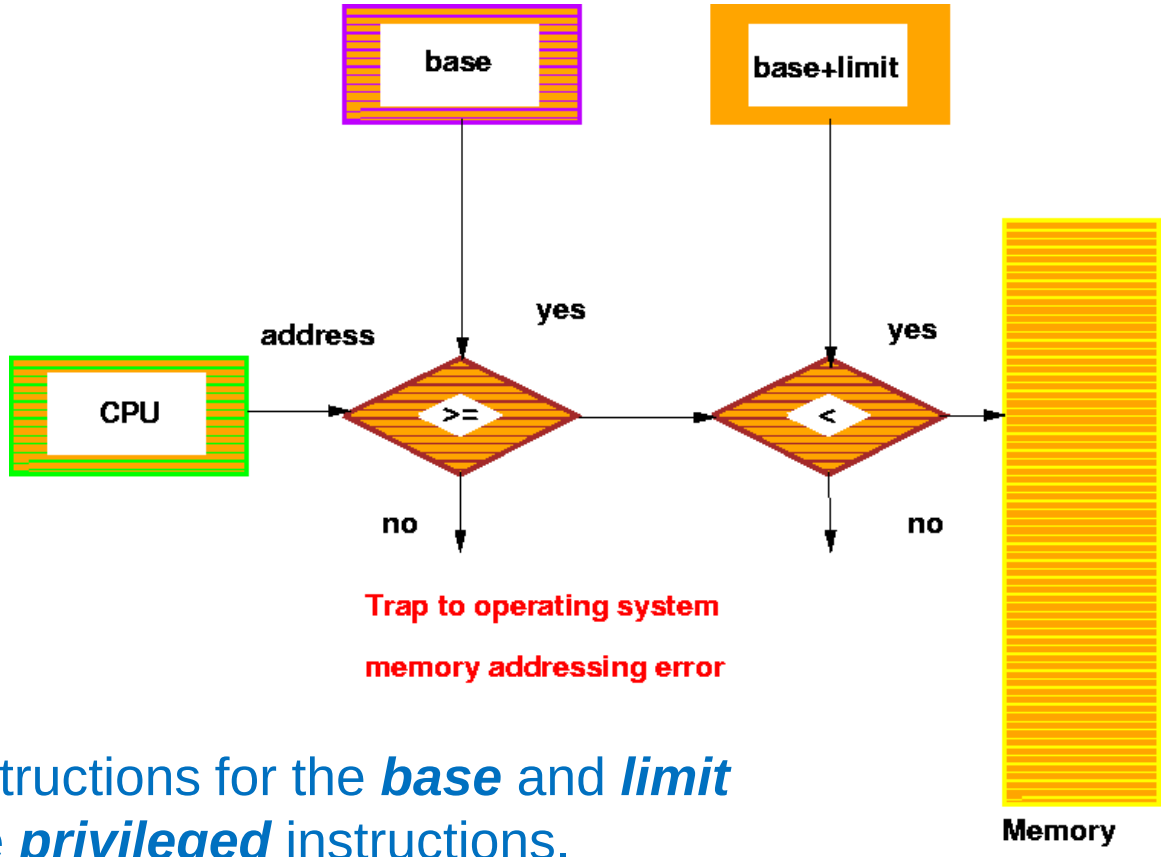
- When a process is running, only memory in that process address space must be accessible.
- When executing in kernel mode, the kernel has unrestricted access to all memory.

Memory Protection: **base and limit**

- To provide memory protection, add two registers that determine the range of legal addresses a program may address.
  - **Base Register** - holds smallest legal physical memory address.
  - **Limit register** - contains the size of the range.
- Memory outside the defined range is protected.
- Sometimes called ***Base and Bounds*** method

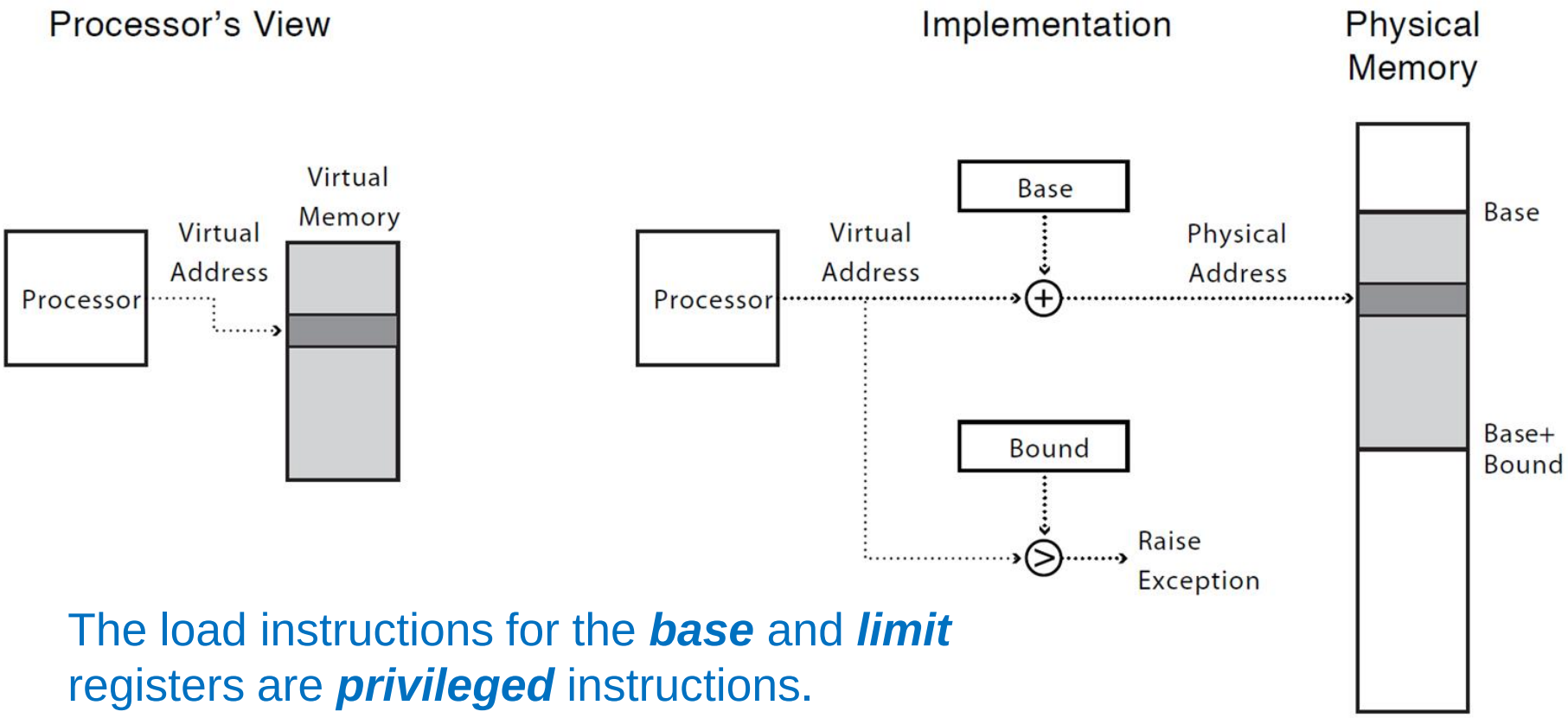


Hardware Address Protection



The load instructions for the ***base*** and ***limit*** registers are ***privileged*** instructions.

Virtual Address translation using the Base and Bounds method



The load instructions for the ***base*** and ***limit*** registers are ***privileged*** instructions.

I/O Protection

- All I/O instructions are privileged instructions.

Question

- Given the I/O instructions are privileged, how do users perform I/O?

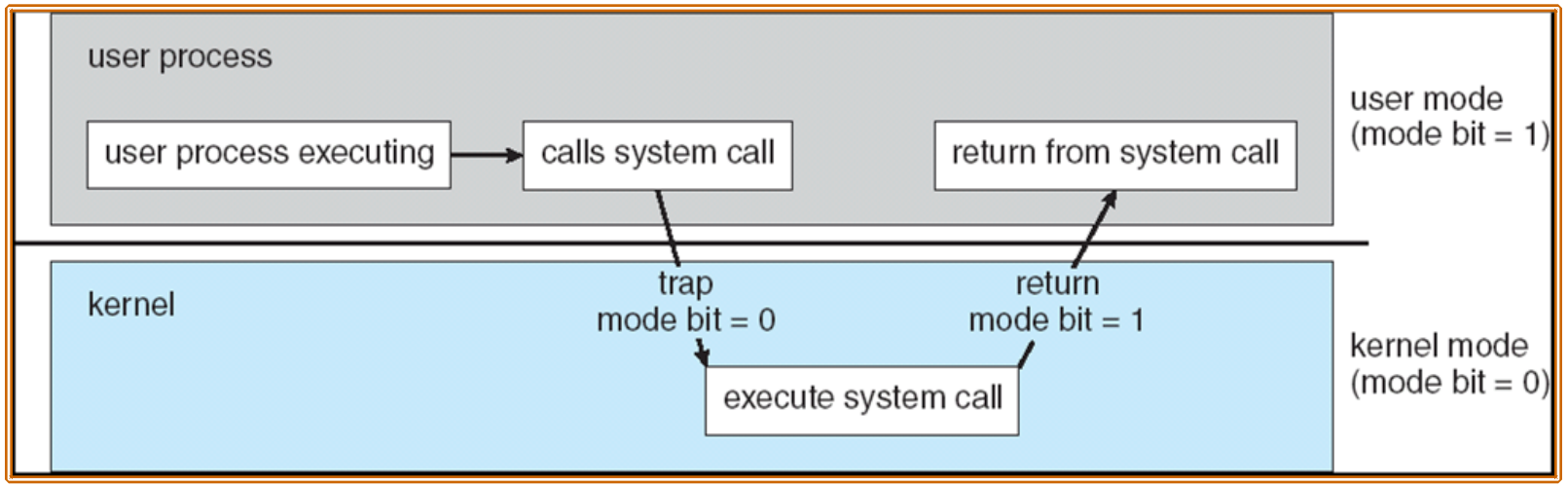
Question

- Given the I/O instructions are privileged, how do users perform I/O?

- Via system calls - the method used by a process to request action by the operating system.

System Calls

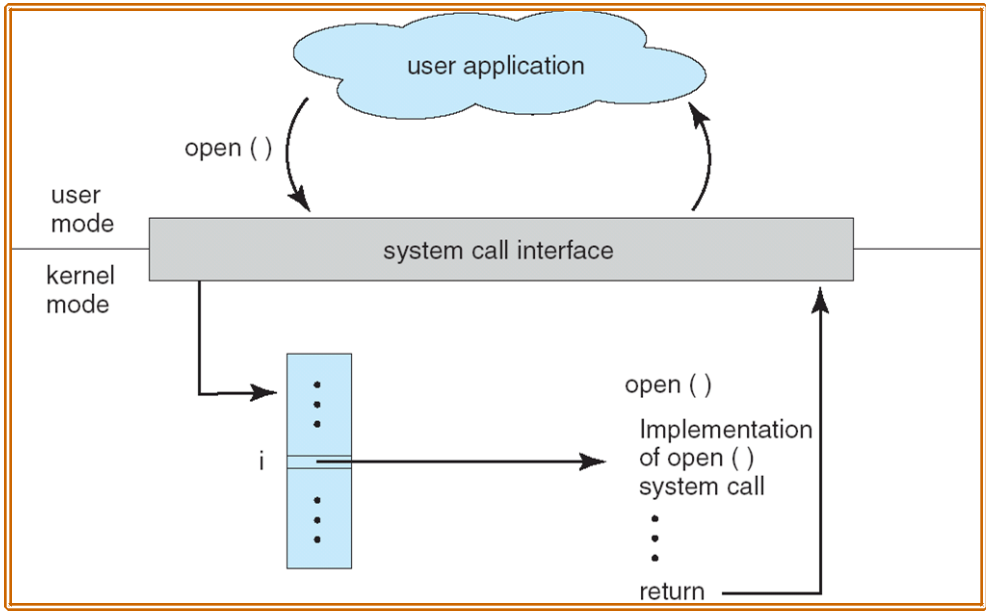
- User code can issue a syscall, which causes a trap
- Kernel handles the syscall



System Calls

- Interface between applications and the OS.
  - Application uses an assembly instruction to trap into the kernel
  - Some higher level languages provide wrappers for system calls (e.g., C)
- System calls pass parameters between an and OS via registers or memory

  .
- Linux has about 300 system calls
  - read(), write(), open(), close(), fork(), exec(), ioctl(),…..



System **services or system programs**

- Convenient environment for program development and execution.
  - Command Interpreter (i.e., shell) - parses/executes other system programs
  - Window management
  - System libraries, e.g., libc

Command Interpreter System

- Commands that are given to the operating system via command statements that execute
  - Process creation and deletion, I/O handling, secondary storage management, main memory Management, file system access, protection, networking, etc.
- Obtains the next command and executes it.
- Programs that read and interpret control statements also called -
  - Command-line interpreter, shell (in UNIX)

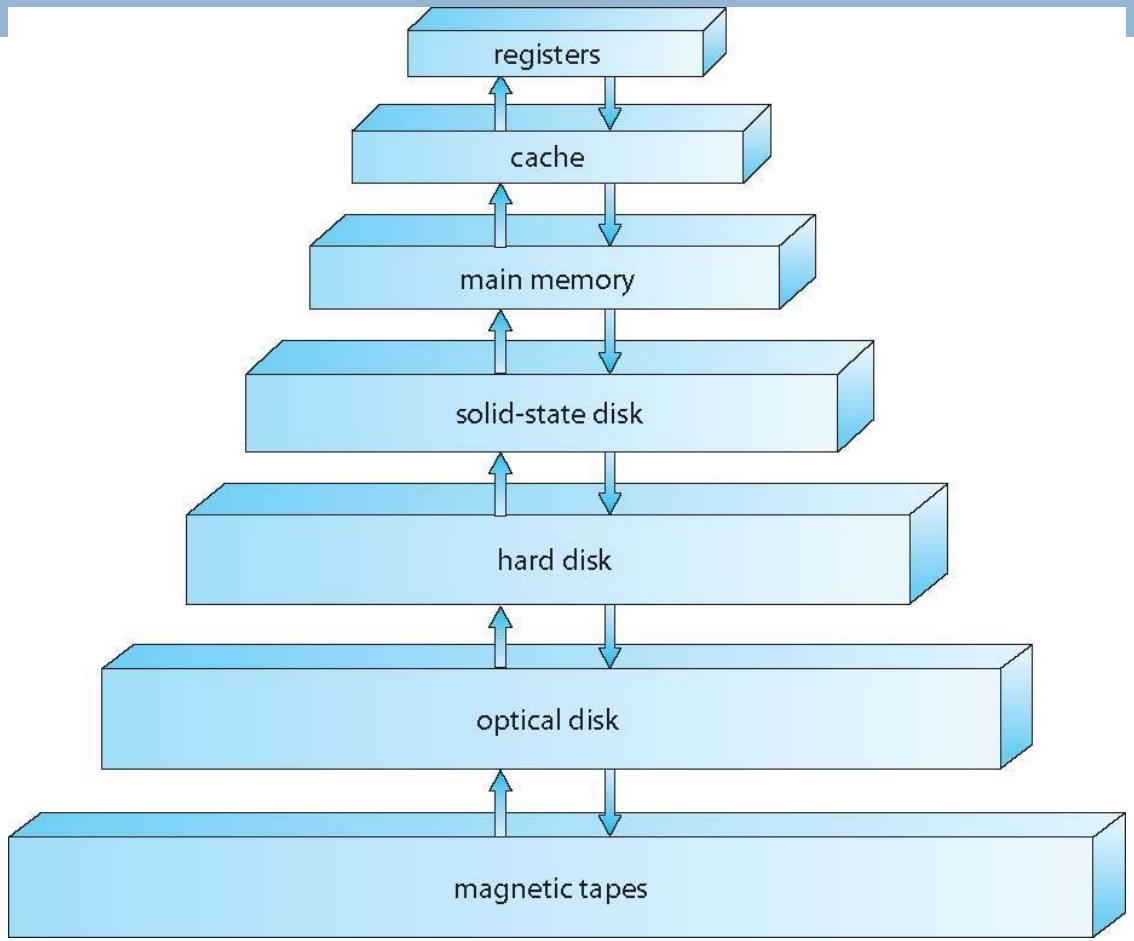
Storage Structure

- Main memory - only large storage media that the CPU can access directly.
- Secondary storage - has large nonvolatile storage capacity.
  - Magnetic disks - rigid metal or glass platters covered with magnetic recording material.
    - Disk surface is logically divided into tracks, subdivided into sectors.
    - Disk controller determines logical interaction between device and computer.

Storage Hierarchy

- Storage systems are organized in a hierarchy based on
  - Speed
  - Cost
  - Volatility
- Caching - process of copying information into faster storage system; main memory can be viewed as fast cache for secondary storage.

Storage Device Hierarchy



OS Task: Process Management

- Process - fundamental concept in OS
  - Process is an instance of a program in execution.
  - Process needs resources - CPU time, memory, files/data and I/O devices.
- OS is responsible for the following process management activities.
  - Process creation and deletion
  - Process suspension and resumption
  - Process synchronization and interprocess communication
  - Process interactions - deadlock detection, avoidance and correction

OS Task: Memory Management

- Main Memory is an array of addressable words or bytes that is quickly accessible.
- Main Memory is volatile.
- OS is responsible for:
  - Allocate and deallocate memory to processes.
  - Managing multiple processes within memory - keep track of which parts of memory are used by which processes. Manage the sharing of memory between processes.
  - Determining which processes to load when memory becomes available.

OS Task: Secondary Storage and I/O Management

- Since primary storage (i.e., main memory) is expensive and volatile, secondary storage is required for backup.
- Disk is the primary form of secondary storage.
  - OS performs storage allocation, free-space management, etc.
- I/O system in the OS consists of
  - Device driver interface that abstracts device details
  - Drivers for specific hardware devices

OS Task: File System Management

- File is a collection of related information - represents programs and data.

- OS is responsible for
  - File creation and deletion
  - Directory creation and deletion
  - Supporting primitives for file/directory manipulation.
  - Mapping files to disks (secondary storage).

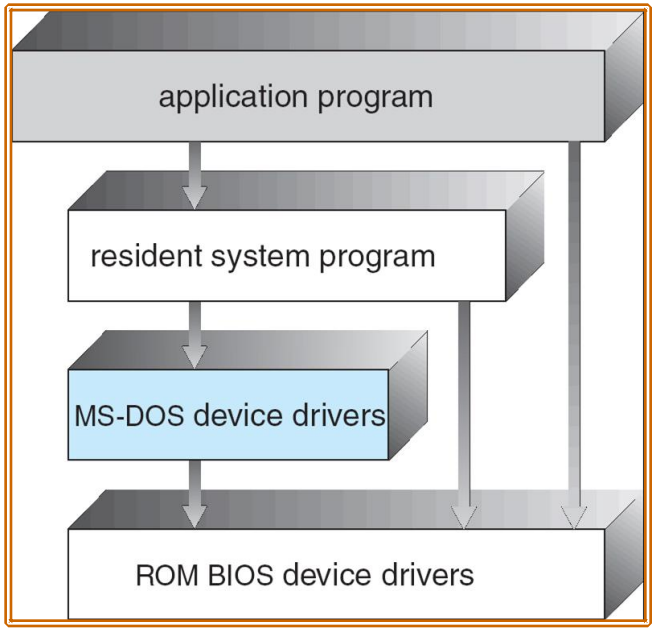
OS Task: Protection and Security

- Protection mechanisms control access of processes to user and system resources.
- Protection mechanisms must:
  - Distinguish between authorized and unauthorized use.
  - Specify access controls to be imposed on use.
  - Provide mechanisms for enforcement of access control.

Operating Systems: How are they organized?

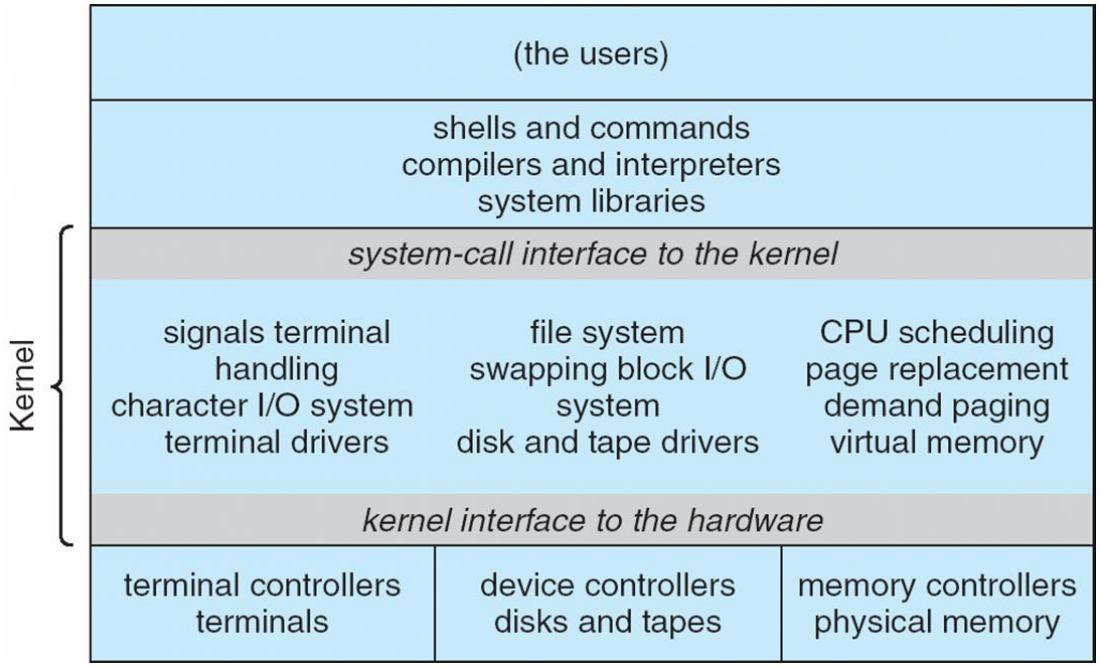
OS Structure - Simple Approach

- MS-DOS - provides a lot of functionality in little space.
  - Not divided into modules, Interfaces and levels of functionality are not well separated



Original UNIX System Structure

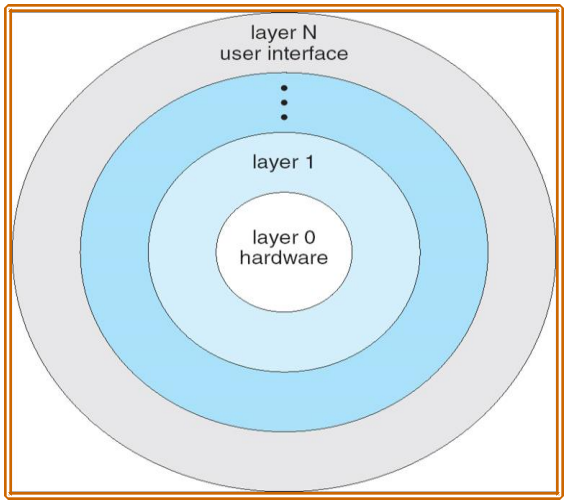
- Limited structuring, has 2 separable parts
  - Systems programs
  - Kernel
    - everything below system call interface and above physical hardware.
    - Filesystem, CPU scheduling, memory management



Layered OS Structure

- OS divided into number of layers - bottom layer is hardware, highest layer is the user interface.
- Each layer uses functions and services of only lower-level layers.
- THE Operating System and Linux Kernel has successive layers of abstraction.

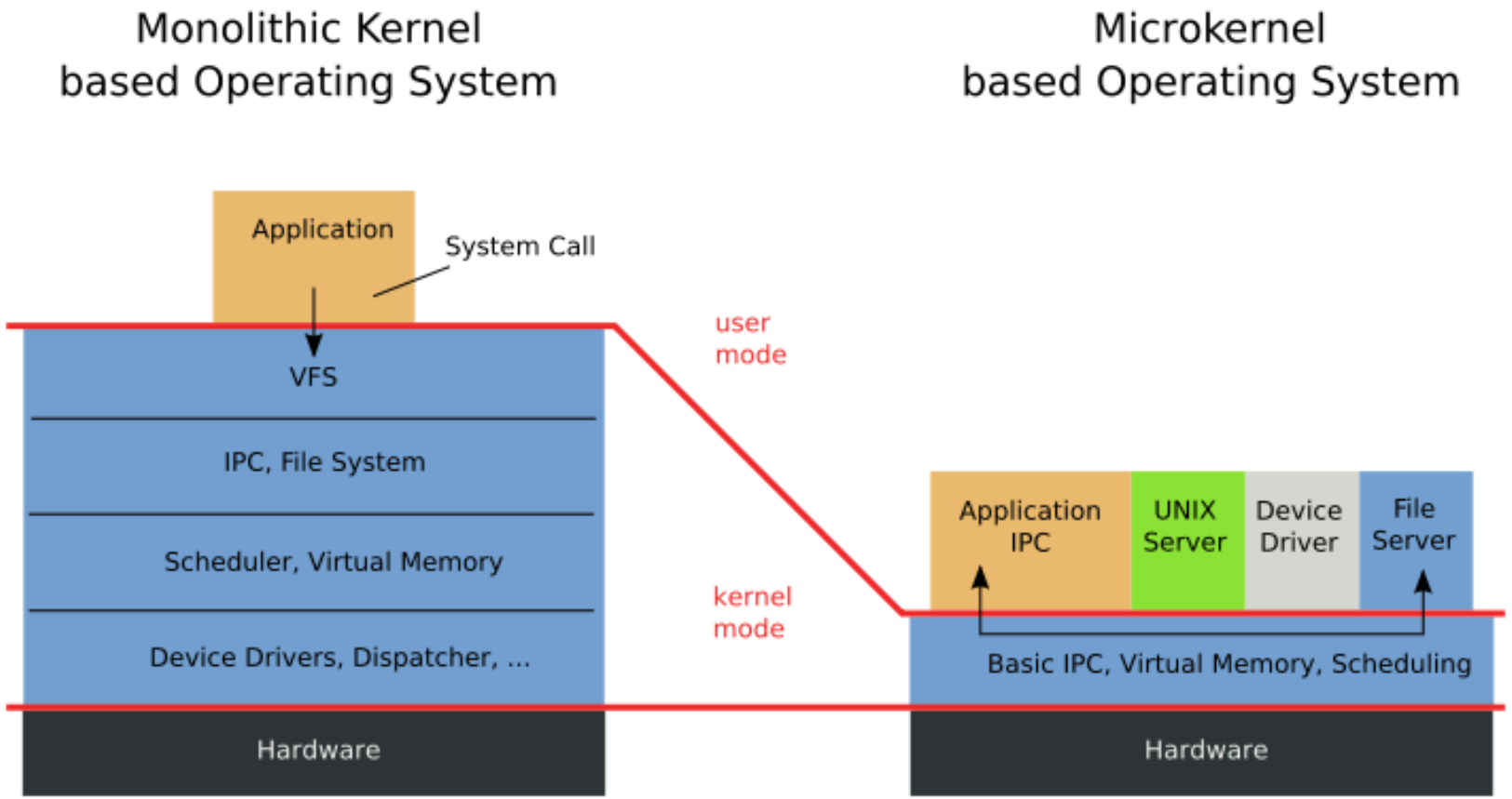
User Programs
Interface Primitives
Device Drivers and Schedulers
Virtual Memory
I/O
CPU Scheduling
Hardware



Monolithic vs. Microkernel OS

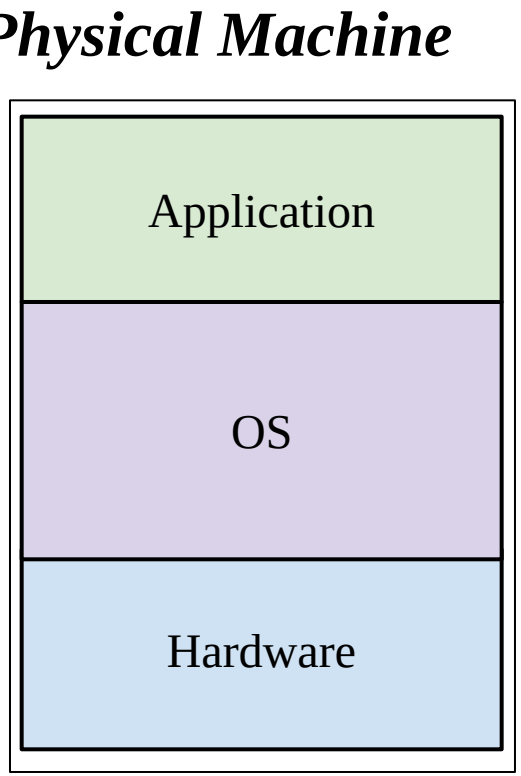
- Monolithic OSes have large kernels with a lot of components
  - Linux, Windows, Mac
- Microkernels moves as much from the kernel into "*user*" space
  - Small core OS components running at kernel level
  - OS Services built from many independent user-level processes
- Communication between modules with message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port OS to new architectures
  - More reliable and more secure (less code is running in kernel mode)
- Detriments:
  - Performance overhead severe for naïve implementation

A microkernel OS

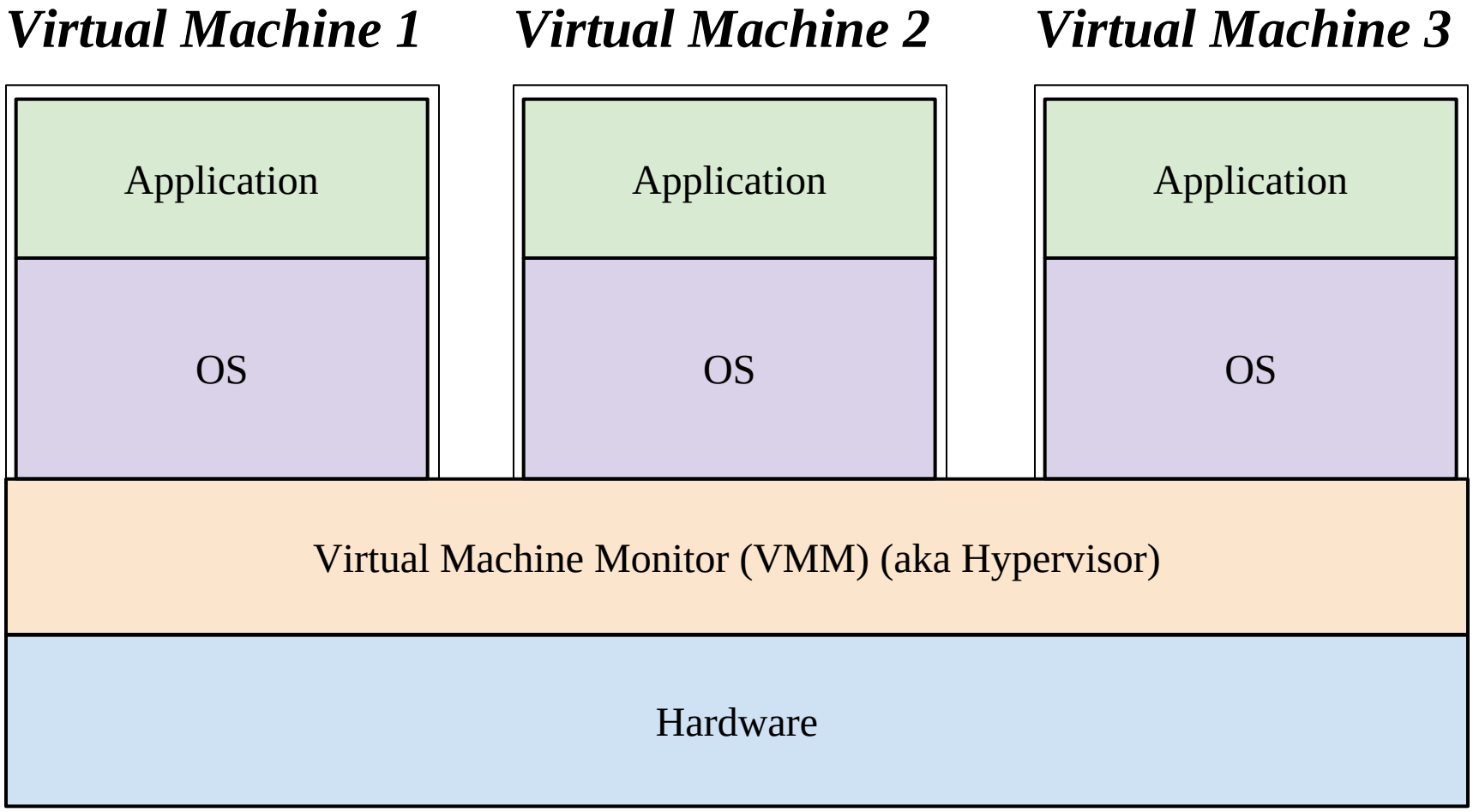


Slide adapted from http://web.cecs.pdx.edu/~walpole/class/cs533/fall2015/home.html

Virtual Machines



Virtual Machines



Virtual Machines

- Use cases
  - Resource configuration
  - Running multiple OSes, either the same or different OSes
  - Run existing OS binaries on different architecture

Summary of Lecture set 1

- What is an operating system?
- Operating systems history
- Computer system and operating system structure

SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS

(20CS0507)

COURSE OBJECTIVES:

1. Explain main components of an OS & their functions.

2. Describe the process management and scheduling.

3. Discuss various issues in Inter Process Communication (IPC) and the role of OS in IPC.

4. Illustrate the concepts and implementation of Memory management policies and virtual memory

5. Explain working of an OS as a resource manager, file system manager, process manager, memory manager and I/O manager and methods used to implement the different parts of OS.

COURSE OUTCOMES (CO's):

On successful completion of the course, the student will be able to

1. Describe the important computer system resources and the role of operating system in their management policies and algorithms.
2. Understand the process management policies and scheduling of processes by CPU.
3. Analyze the requirement for process synchronization and coordination handled by operating system.
4. Describe and analyze the memory management and its allocation policies. Technologies
5. Categorize the storage management policies with respect to different storage management technologies
6. Study the need for special purpose operating system with the advent of new emerging technologies

Topics To Be Covered:

Processes:

Process, Process state, Process Scheduling,-Schedulers - Inter process Communication synchronization -- Scheduling Algorithms : FCFS-SJF-Priority- RR - Message Passing

Threads:

Definition, Multithreading-Advantages- ULTs - KLTs- Thread Libraries - Difference between ULTs and KLTs.

OPERATING SYSTEMS

Processes:

Process, Process state, Process Scheduling,-Schedulers - Inter process Communication synchronization -- Scheduling Algorithms : FCFS-SJF-Priority- RR - Message Passing

Threads:

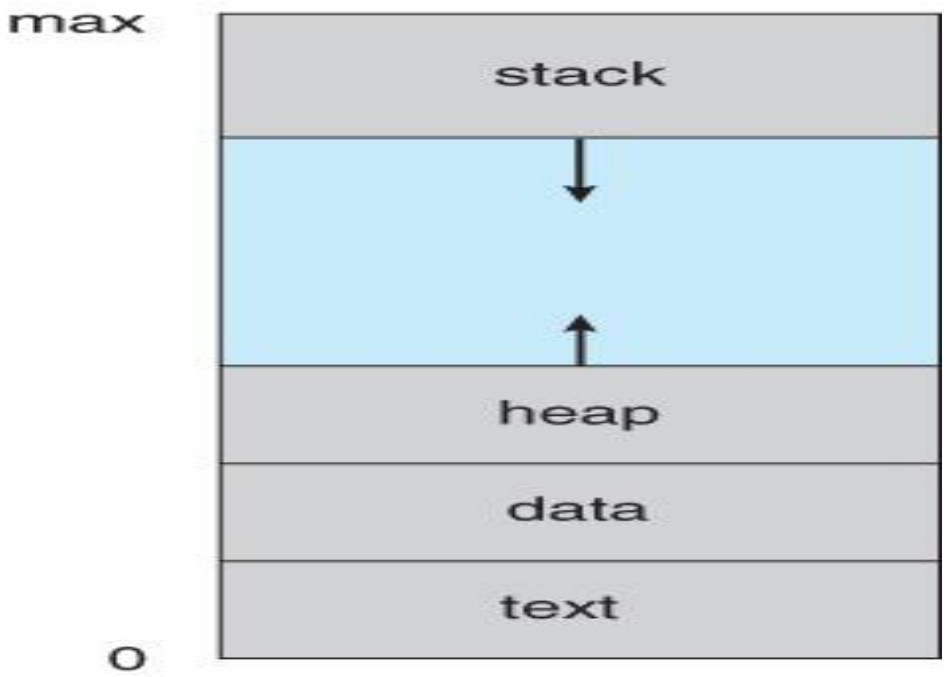
Definition, Multithreading-Advantages- ULTs - KLTs- Thread Libraries - Difference between ULTs and KLTs.

Process:

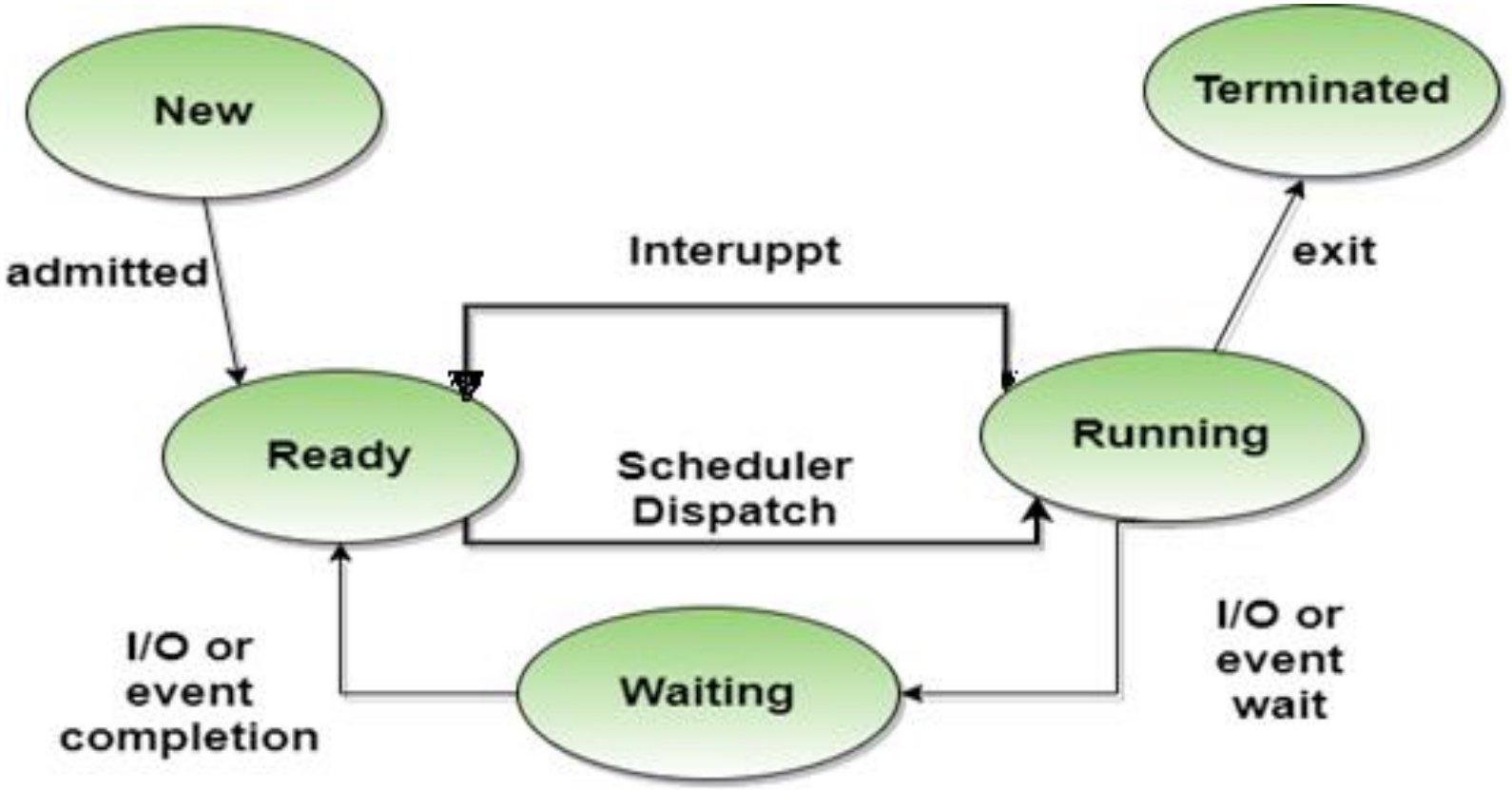
A process is a program in execution. The process is not as same as program code but a lot more than it. A process is an 'active' entity as opposed to the program which is considered to be a 'passive' entity because programs contains set of instructions only does not perform any task but process has to execute those instructions in program and stores the resultant into memory.

Process:

- The structure of Process in memory is divided into four sections as shown in Figure:



Process state:



Process Control Block:

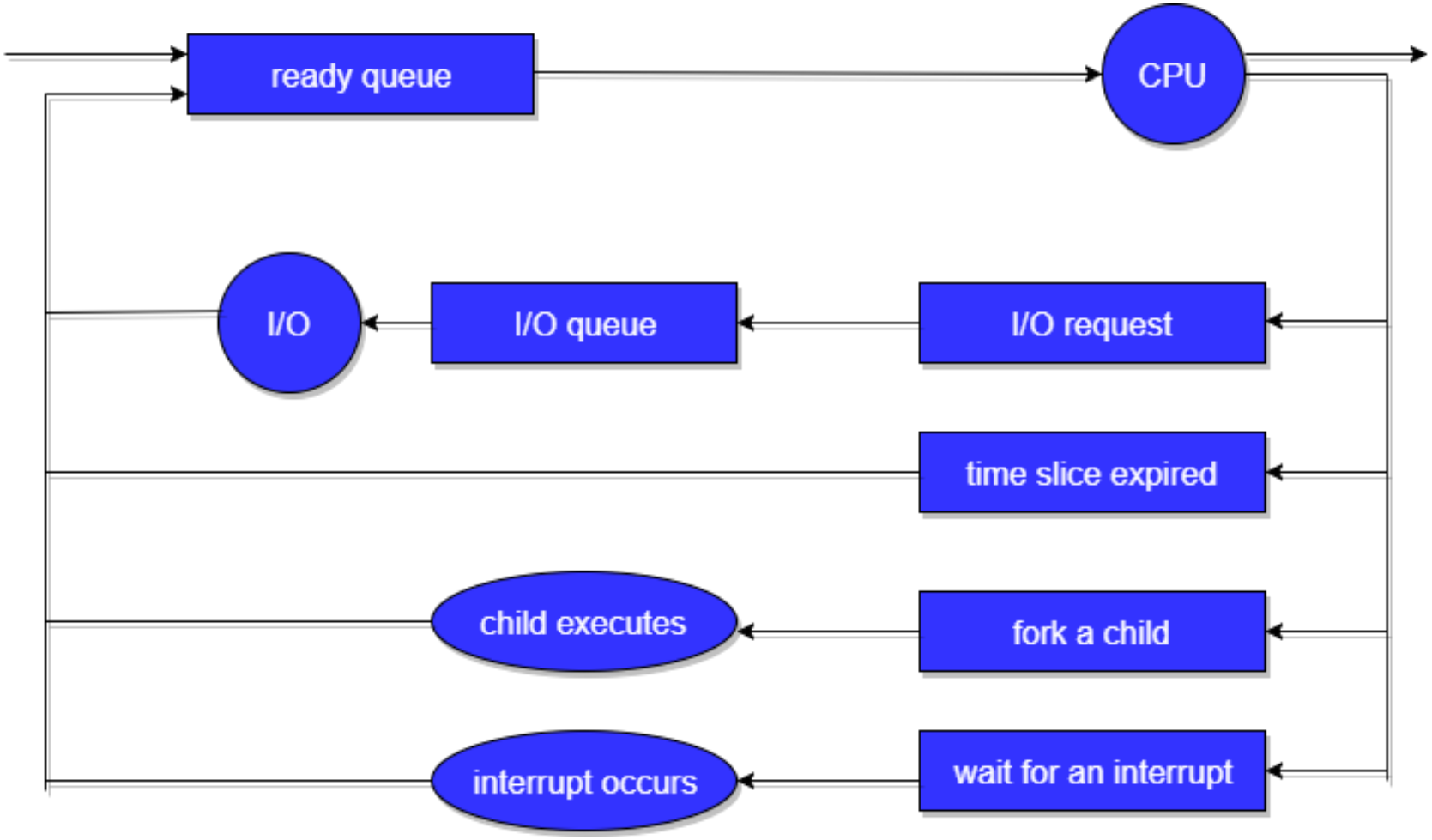
- There is a Process Control Block for each process, enclosing all the information about the process. It is also known as the task control block. It is a data structure, which contains the following:

Process Scheduling:

Process Scheduling-

- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running.
- To meet these objectives, the process scheduler selects an available process (possibly from a set of several available processes) for program execution on the CPU.
    - For a single-processor system, there will never be more than one running process.
    - If there are more processes, the rest will have to wait until the CPU is free and can be rescheduled.

Process Scheduling:



Schedulers

There are three types of schedulers available:

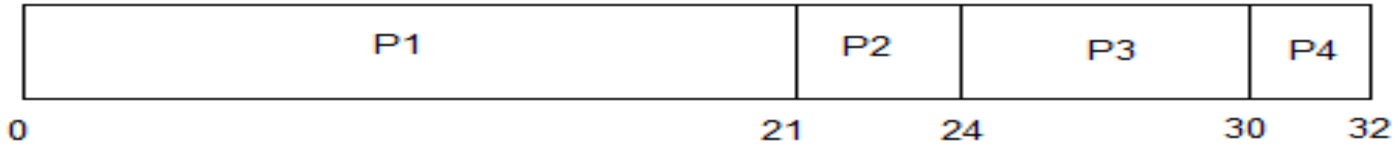
- Long Term Scheduler
- Short Term Scheduler
- Medium Term Scheduler

Scheduling Algorithms:

1. First Come First Serve Scheduling

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The average waiting time will be = (0 + 21 + 24 + 30)/4 = 18.75 ms



This is the GANTT chart for the above processes

Scheduling Algorithms:

2. Shortest Job First (SJF) Scheduling

Process	Arrival time	Burst time
P1	3 ms	5 ms
P2	0 ms	4 ms
P3	4 ms	2 ms
P4	5 ms	4 ms

Gantt Chart

P2	P3	P4	P1
0ms 4ms	4ms 6ms	6ms 10ms	10ms 15ms

Scheduling Algorithms:

3. Priority Scheduling (Non-preemptive)

Process	Arrival time	Burst time	Priority
P1	0 ms	5 ms	1
P2	1 ms	3 ms	2
P3	2 ms	8 ms	1
P4	3 ms	6 ms	3

NOTE: In this example, we are taking higer priority number as higher priority.

Gantt Chart

P1	P4	P2	P3
0ms 5ms	5ms 11ms	11ms 14ms	14ms 22ms

Scheduling Algorithms:

3. Round-Robin Algorithm

Process	Arrival time	Burst time
P1	0 ms	10 ms
P2	0 ms	5 ms
P3	0 ms	8 ms

Gantt Chart

P1		P2		P3		P1		P2		P3	
1	2	2	4	4	6	6	8	8	10	10	12

P1		P2		P3		P1		P3		P1	
12	14	14	15	15	17	17	19	19	21	21	23

Inter Process Communication (IPC):

Inter-process communication (IPC) is a mechanism that allows processes to communicate with each other and synchronize their actions.

A process can be of two types:

- Independent process.
- Co-operating process.

Inter Process Communication (IPC):

An ***independent process*** is not affected or affected by the execution of other processes. Any process that does not share data with any other process is independent.

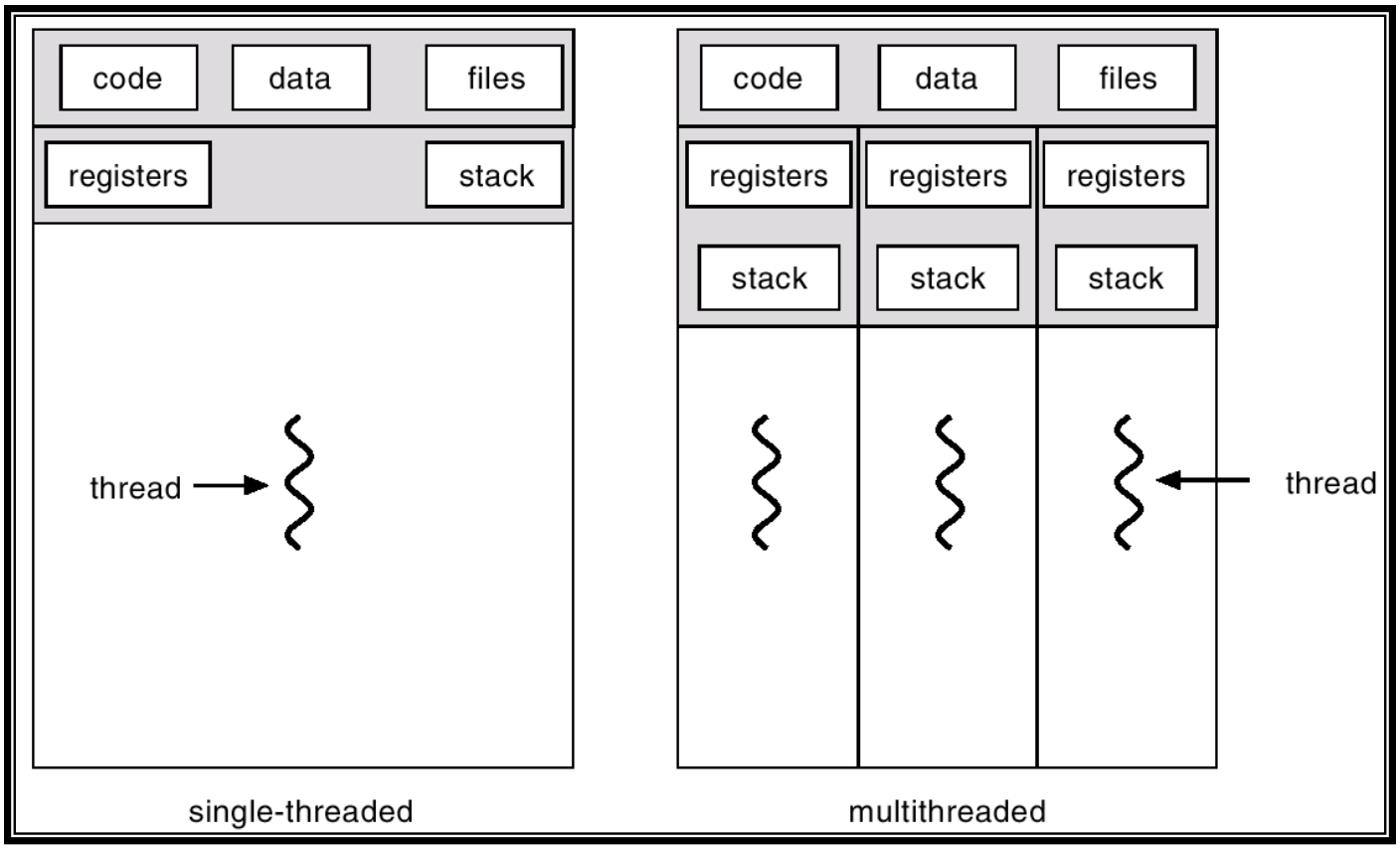
A ***co-operating process*** can be affected by other executing processes. Any process that shares data with any other process is co-operating process.

Processes can communicate with each other through both:

- Shared Memory
- Message passing

THREADS

Single and Multithreaded Processes



THREADS

Benefits

- Responsiveness
- Resource Sharing
- Economy
- Utilization of MP Architectures

THREADS

User Threads

- Thread management done by user-level threads library

- Examples
  - POSIX *Pthreads*
  - Mach *C-threads*
  - Solaris *threads*

Kernel Threads

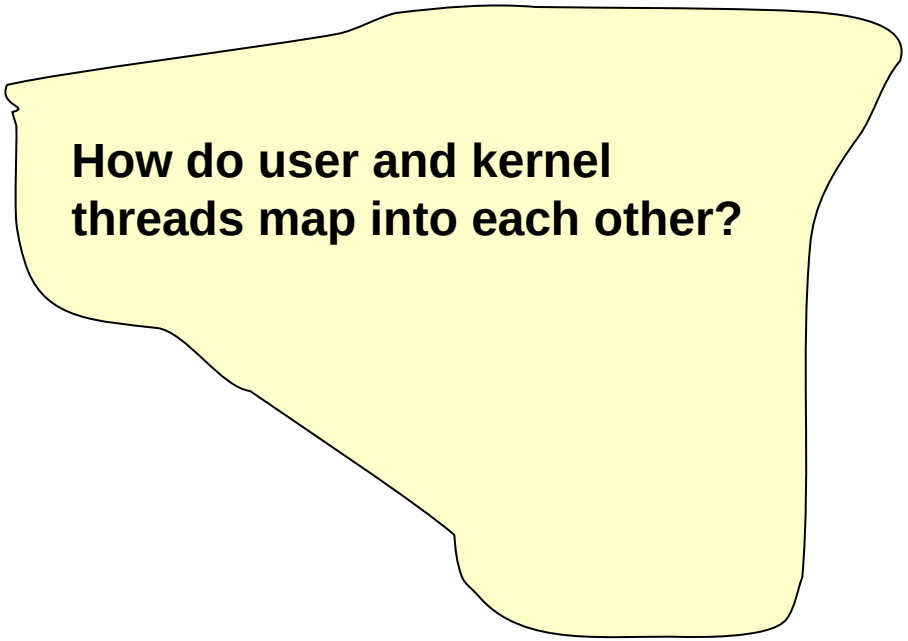
- Supported by the Kernel

- Examples
  - Windows 95/98/NT/2000
  - Solaris
  - Tru64 UNIX
  - BeOS
  - Linux

THREADS

Multithreading Models

- Many-to-One
- One-to-One
- Many-to-Many



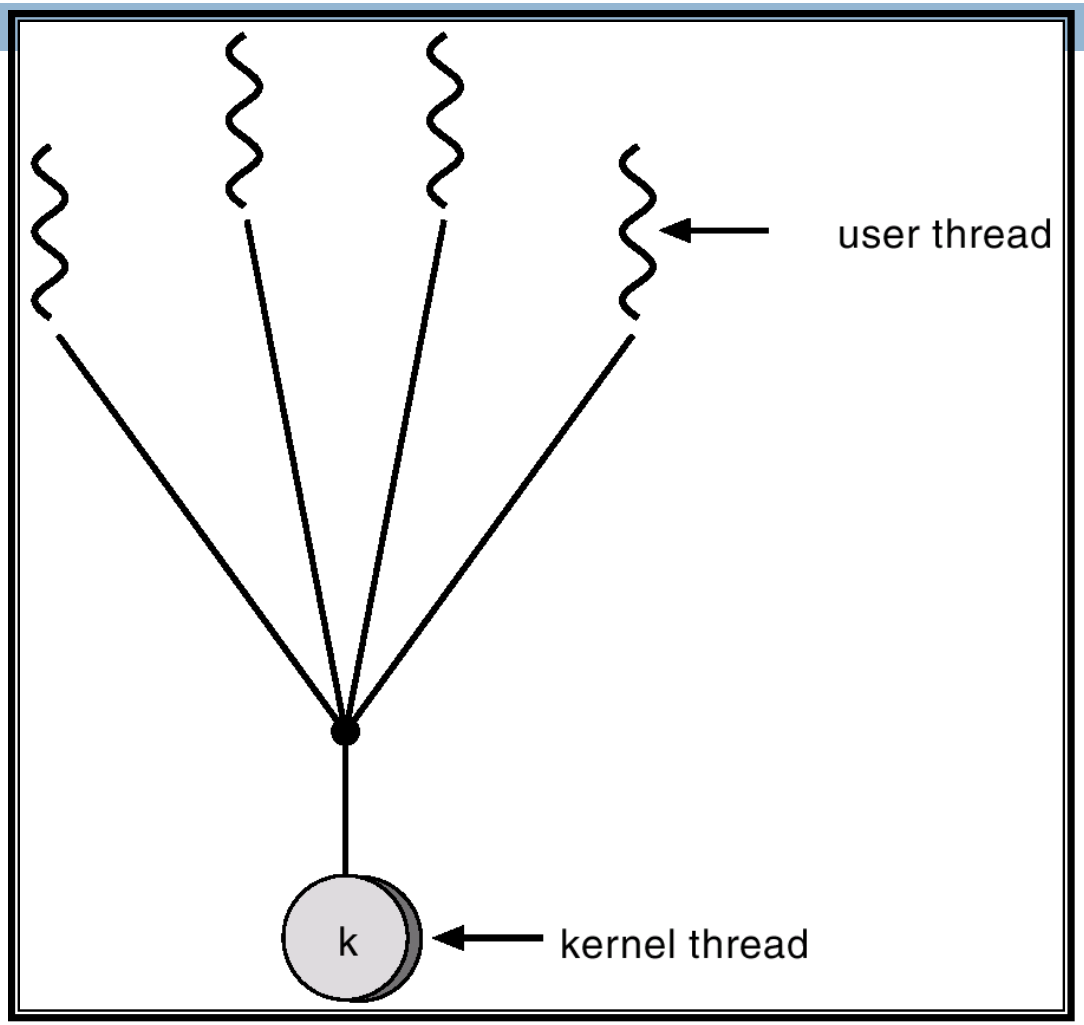
THREADS

Many-to-One

- Many user-level threads mapped to single kernel thread.
- Used on systems that do not support kernel threads.
- Examples:

  Solaris Green Threads

  GNU Portable Threads



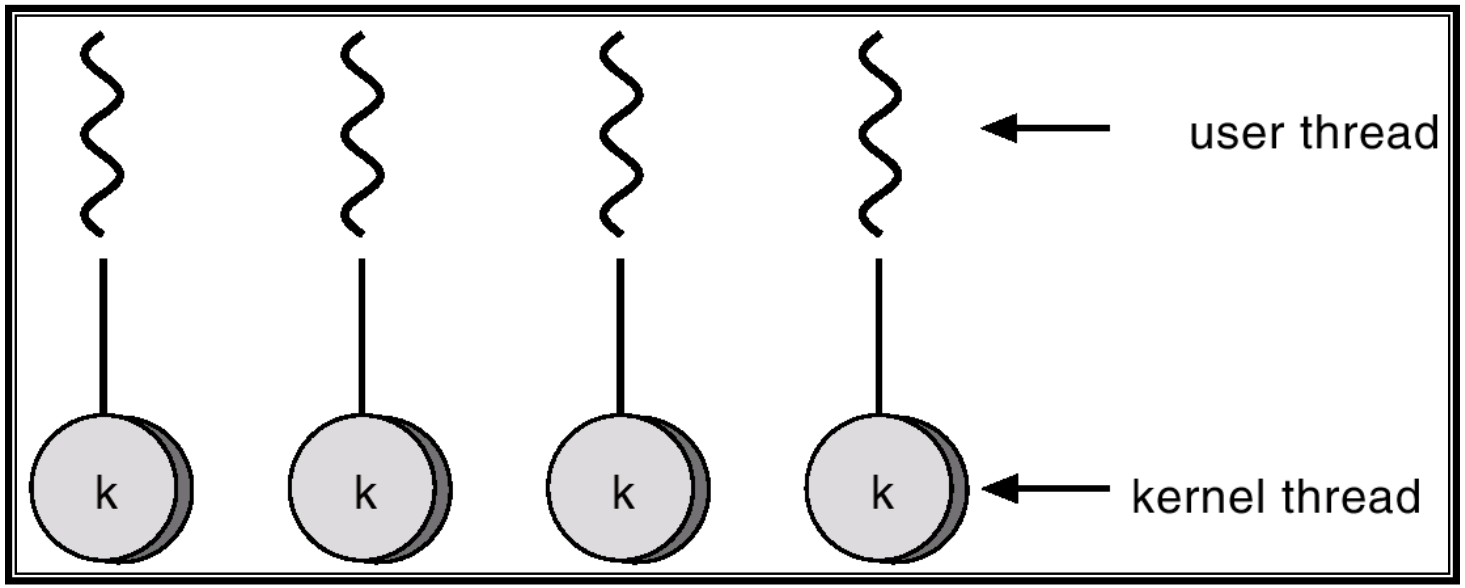
THREADS

One-to-One

- Each user-level thread maps to kernel thread.

Examples

- Windows 95/98/NT/2000
- Linux



THREADS

Threading Issues

Semantics of fork() and exec() system calls

- Does **fork()** duplicate only the calling thread or all threads?

Thread cancellation

- Terminating a thread before it has finished
- Two general approaches:
  - **Asynchronous cancellation** terminates the target thread immediately
  - **Deferred cancellation** allows the target thread to periodically check if it should be cancelled

THREADS

Threading Issues

Signal handling

- Signals are used in UNIX systems to notify a process that a particular event has occurred
- A **signal handler** is used to process signals
  1. Signal is generated by particular event
  2. Signal is delivered to a process
  3. Signal is handled
- Options:
  - Deliver the signal to the thread to which the signal applies
  - Deliver the signal to every thread in the process
  - Deliver the signal to certain threads in the process
  - Assign a specific threa to receive all signals for the process

THREADS

Threading Issues

Thread pools

- Create a number of threads in a pool where they await work
- Advantages:
  - Usually slightly faster to service a request with an existing thread than create a new thread
  - Allows the number of threads in the application(s) to be bound to the size of the pool

Thread specific data

- Allows each thread to have its own copy of data
- Useful when you do not have control over the thread creation process (i.e., when using a thread pool)

THREADS

Threading Issues

Scheduler activations

- Many:Many models require communication to maintain the appropriate number of kernel threads allocated to the application
- Scheduler activations provide **upcalls** - a communication mechanism from the kernel to the thread library
- This communication allows an application to maintain the correct number kernel threads.

SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS

(20CS0507)

COURSE OBJECTIVES:

1. Explain main components of an OS & their functions.

2. Describe the process management and scheduling.

3. Discuss various issues in Inter Process Communication (IPC) and the role of OS in IPC.

4. Illustrate the concepts and implementation of Memory management policies and virtual memory

5. Explain working of an OS as a resource manager, file system manager, process manager, memory manager and I/O manager and methods used to implement the different parts of OS.

COURSE OUTCOMES (CO's):

On successful completion of the course, the student will be able to

1. Describe the important computer system resources and the role of operating system in their management policies and algorithms.
2. Understand the process management policies and scheduling of processes by CPU.
3. Analyze the requirement for process synchronization and coordination handled by operating system.
4. Describe and analyze the memory management and its allocation policies. Technologies
5. Categorize the storage management policies with respect to different storage management technologies
6. Study the need for special purpose operating system with the advent of new emerging technologies

Topics To Be Covered:

Process Synchronization:

Critical Section- Mutual Exclusion- Semaphores- Monitors. Classical Problems of Synchronization: The Producer- Consumer Problem -Dinning Philosopher Problem- Reader‟s & Writer Problem

Deadlocks:

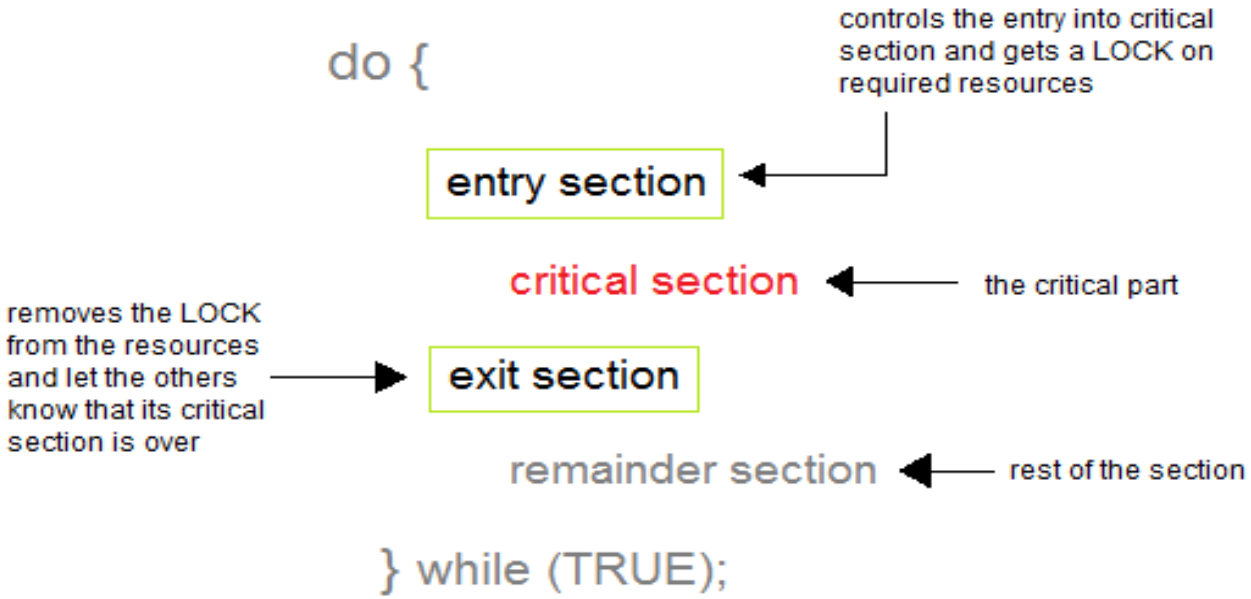
Definition- Deadlock Characteristics- Deadlock Prevention and Deadlock Avoidance: Banker‟s algorithm- Deadlock detection and Recovery.

Process Synchronization:

- It is the task phenomenon of coordinating the execution of processes in such a way that no two processes can have access to the same shared data and resources.
- Process Synchronization is mainly needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or any data at the same time.
- It is a procedure that is involved in order to preserve the appropriate order of execution of cooperative processes. In order to synchronize the processes, there are various synchronization mechanisms.

The Critical-Section Problem:

- Consider a system consisting of n processes {$P0$, $P1$, ..., $Pn-1$}. Each process has a segment of code, called a **critical section**, in which the process may be changing shared variables, updating a table, writing a file, and so on.



The Critical-Section Problem:

- Consider a system consisting of n processes {$P0$, $P1$, ..., $Pn-1$}. Each process has a segment of code, called a **critical section**, in which the process may be changing shared variables, updating a table, writing a file, and so on.

Solution to the Critical Section Problem

The solution to the critical section problem must satisfy the following conditions −

- **Mutual Exclusion**

  Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

The Critical-Section Problem:

- **Progress**

  If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.

- **Bounded Waiting**

  There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

Semaphore:

- Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations such as
  - **wait(s)** or p(s) or Down(s) and
  - **signal(s)** or v(s) or Up(s)

These are used for process synchronization. The definitions of wait and signal are as follows −

- **Wait**

  The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

Semaphore:

```
wait(S)
{
   while (S<=0);
   S--;
}
```

- **Signal**

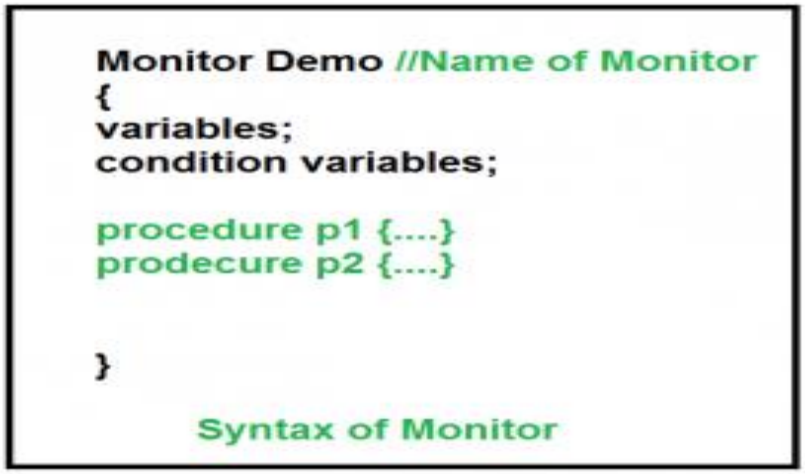
The signal operation increments the value of its argument S.

```
signal(S)
{
   S++;
}
```

Monitors:

- It is the collection of condition variables and procedures combined together in a special kind of module or a package.
- The processes running outside the monitor can't access the internal variable of the monitor but can call procedures of the monitor.
- Only one process at a time can execute code inside monitors.

Syntax:



Monitors:

- **Advantages of Monitor:**

Monitors have the advantage of making parallel programming easier and less error prone than using techniques such as semaphore.

- **Disadvantages of Monitor:**

Monitors have to be implemented as part of the programming language . The compiler must generate code for them. This gives the compiler the additional burden of having to know what operating system facilities are available to control access to critical sections in concurrent processes. Some languages that do support monitors are Java,C#,Visual Basic,Ada and concurrent Euclid.

Critical section problems:

The Reader and Writer Problem using semaphore:

Writer process:

1. Writer requests the entry to critical section.
2. If allowed i.e. wait() gives a true value, it enters and performs the write. If not allowed, it keeps on waiting.
3. It exits the critical section.

```
do {            // writer requests for critical section
   wait(wrt);                    // performs the write
    // leaves the critical section
   signal(wrt);
} while(true);
```

Critical section problems:

Reader process:

1. Reader requests the entry to critical section.
2. If allowed:
   - it increments the count of number of readers inside the critical section. If this reader is the first reader entering, it locks the **wrt** semaphore to restrict the entry of writers if any reader is inside.
   - It then, signals mutex as any other reader is allowed to enter while others are already reading.
   - After performing reading, it exits the critical section. When exiting, it checks if no more reader is inside, it signals the semaphore "wrt" as now, writer can enter the critical section.
3. If not allowed, it keeps on waiting.

Critical section problems:

```
do {
    // Reader wants to enter the critical section
   wait(mutex);
    // The number of readers has now increased by 1
   readcnt++;
    // there is atleast one reader in the critical section
          // this ensure no writer can enter if there is even one
              reader
    // thus we give preference to readers here
   if (readcnt==1)
     wait(wrt);
              // other readers can enter while this current reader
              is   inside
    // the critical section
```

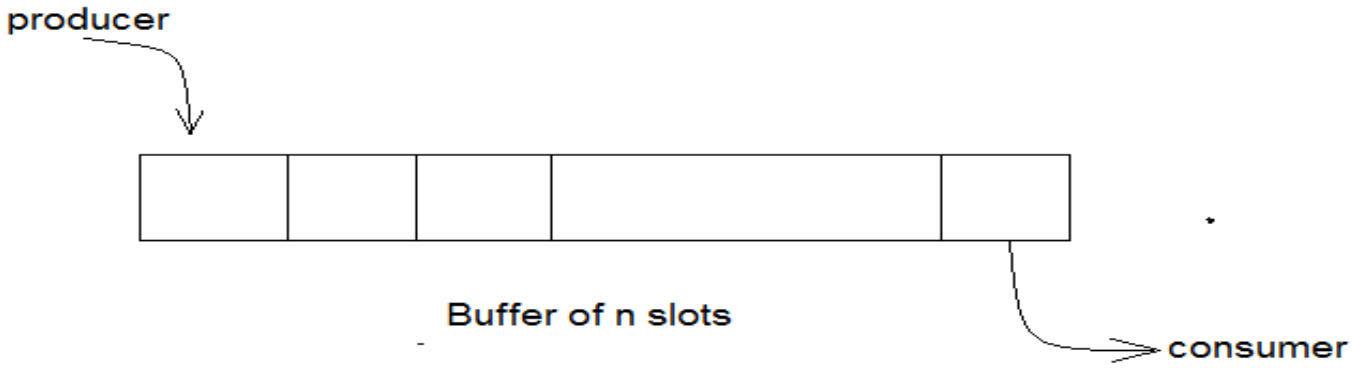
Critical section problems:

```
signal(mutex);
    // current reader performs reading here
   wait(mutex);   // a reader wants to leave
   readcnt--;
    // that is, no reader is left in the critical section,
   if (readcnt == 0)
       signal(wrt);         // writers can enter
   signal(mutex); // reader leaves
} while(true);
```

Critical section problems:

Bounded Buffer Problem using semaphore:

- Bounded buffer problem, which is also called **producer consumer problem**, is one of the classic problems of synchronization.
- There is a buffer of n slots and each slot is capable of storing one unit of data. There are two processes running, namely, **producer** and **consumer**, which are operating on the buffer.



Critical section problems:

- The Producer Operation

The pseudocode of the producer function looks like this:

```
do {                    // wait until empty > 0 and then decrement
'empty'    wait(empty);      // acquire lock    wait(mutex);      /*
enter into critical section          *//* perform the insert operation
                                     in a slot */    // release lock
signal(mutex);              // increment 'full'    signal(full);
 }
while(TRUE)
```

Critical section problems:

- The Consumer Operation

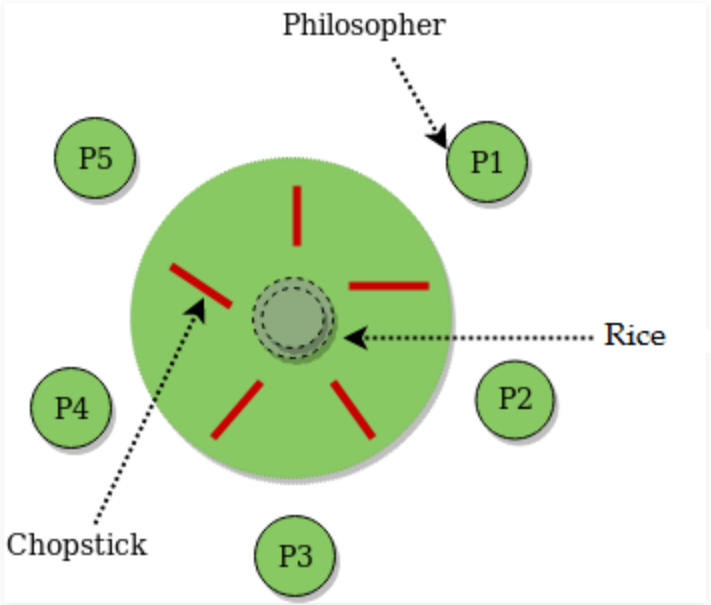
The pseudocode for the consumer function looks like this:

```
do {                    // wait until full > 0 and then decrement
                       'full'
wait(full);             // acquire the lock
 wait(mutex);
/* perform the remove operation in a slot */
// release the lock
 signal(mutex);         // increment 'empty'
 signal(empty);
 }
while(TRUE);
```

Critical section problems:

The Dining-Philosophers Problem:

- Consider five philosophers who spend their lives thinking and eating. The philosophers share a circular table surrounded by five chairs, each belonging to one philosopher. In the center of the table is a bowl of rice, and the table is laid with five single chopsticks.



Critical section problems:

Solution of Dining Philosophers Problem

A solution of the Dining Philosophers Problem is to use a semaphore to represent a chopstick. A chopstick can be picked up by executing a wait operation on the semaphore and released by executing a signal semaphore.

The structure of the chopstick is shown below −

```
semaphore chopstick [5];
```

Initially the elements of the chopstick are initialized to 1 as the chopsticks are on the table and not picked up by a philosopher.

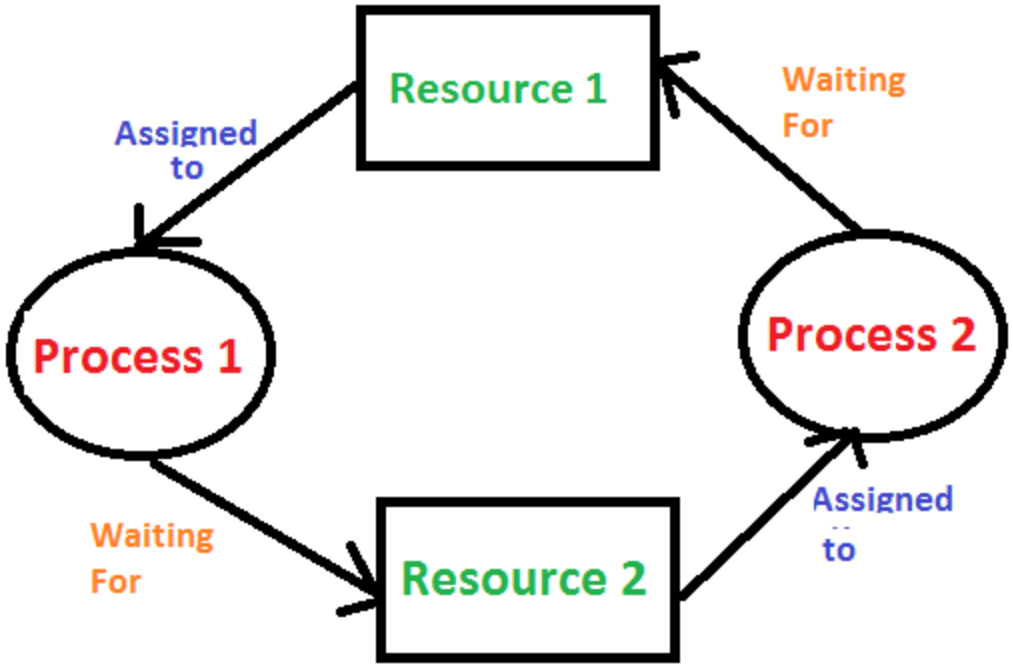
Critical section problems:

The structure of a random philosopher i is given as follows −

```
do {
   wait( chopstick[i] );     //left chopstic
   wait( chopstick[ (i+1) % 5] );      //right chopstic
   . .    .
EATING THE RICE
   .
   signal( chopstick[i] );       //left chopstic
   signal( chopstick[ (i+1) % 5] );     //right chopstic
   .
   . THINKING
   .
} while(1);
```

Deadlock:

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process (no other process executing the complete resources).



Deadlock characteristics:

Deadlock can arise if the **following four conditions hold simultaneously (Necessary Conditions)**

- **Mutual exclusion**. At least one resource must be held in a nonsharable mode; that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.
- **Hold and wait**. A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
- **No preemption**. Resources cannot be preempted; that is, a resource can be released only voluntarily by the process holding it, after that process has completed its task.

Deadlock characteristics:

- **No preemption**. Resources cannot be preempted; that is, a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- **Circular wait**. A set $\{P0, P1, ..., Pn\}$ of waiting processes must exist such that $P0$ is waiting for a resource held by $P1$, $P1$ is waiting for a resource held by $P2$, ..., $Pn-1$ is waiting for a resource held by Pn, and Pn is waiting for a resource held by $P0$.

SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS

(20CS0507)

COURSE OBJECTIVES:

1. Explain main components of an OS & their functions.

2. Describe the process management and scheduling.

3. Discuss various issues in Inter Process Communication (IPC) and the role of OS in IPC.

4. Illustrate the concepts and implementation of Memory management policies and virtual memory

5. Explain working of an OS as a resource manager, file system manager, process manager, memory manager and I/O manager and methods used to implement the different parts of OS.

COURSE OUTCOMES (CO's):

On successful completion of the course, the student will be able to

1. Describe the important computer system resources and the role of operating system in their management policies and algorithms.
2. Understand the process management policies and scheduling of processes by CPU.
3. Analyze the requirement for process synchronization and coordination handled by operating system.
4. Describe and analyze the memory management and its allocation policies. Technologies
5. Categorize the storage management policies with respect to different storage management technologies
6. Study the need for special purpose operating system with the advent of new emerging technologies

Topics To Be Covered:

Memory Management:
Main Memory-Swapping-Contiguous Memory allocation Paging Segmentation.

Virtual memory:
Basics of Virtual Memory- Demand paging-Page-replacement-Page Replacement algorithms-Thrashing. **Disk scheduling:**
FCFS- SSTF- SCAN- C-SCAN- Disk Management.

Memory Management :

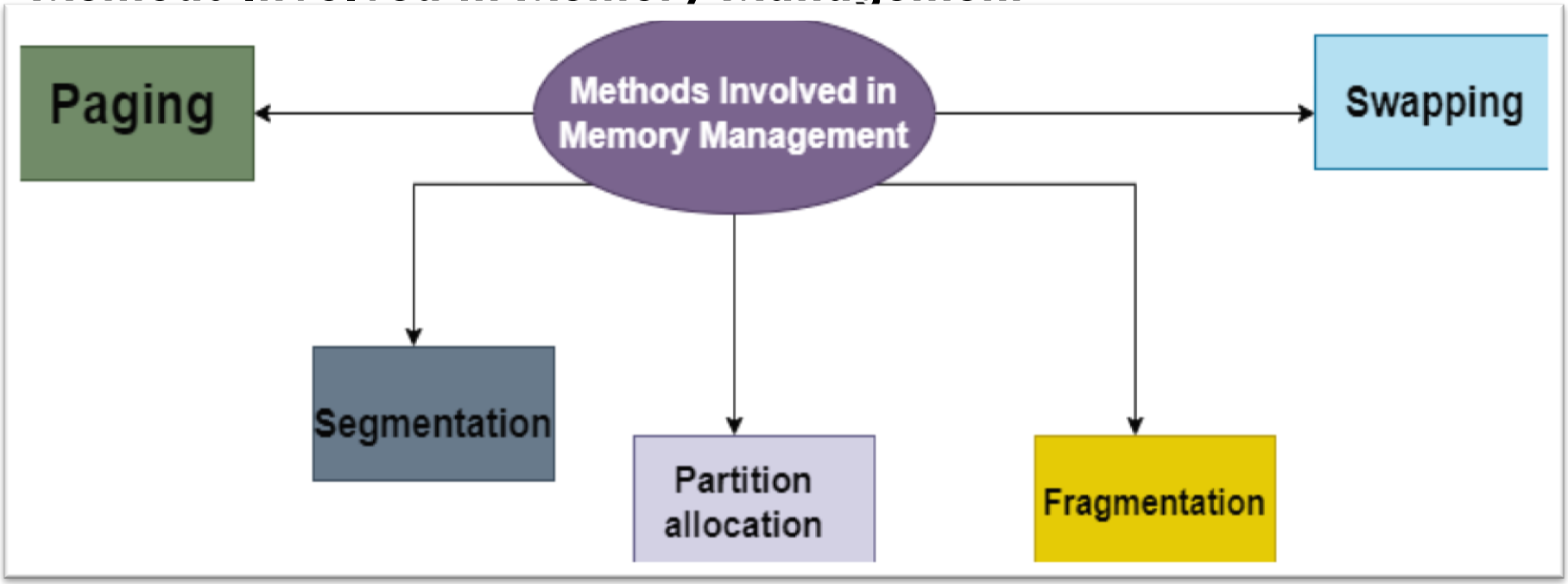
- **Memory Management** is the process of coordinating and controlling the memory in a computer, Blocks are assigning portions that are assigned to various running programs in order to optimize the overall performance of the system. This technique helps in keeping the track of each and every memory location, whether the memory location is allocated to some process or it is free.
- This technique decides which process will get memory at what time. It also keeps the count of how

Main Memory:

Computer Memory is basically known as a collection of data that is represented in binary format.

Main Memory refers to a physical memory that is the internal memory of the computer.

- **Methods Involved in Memory Management**



Swapping:

Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization. In secondary memory, the place where the swapped-out process is stored is called swap space.

The concept of swapping has divided into two more concepts: Swap-in and Swap-out.

- Swap-out is a method of removing a process from RAM and adding it to the hard disk.
- Swap-in is a method of removing a program from a hard disk and putting it back into the main memory or RAM.

Swapping:

Advantages of Swapping

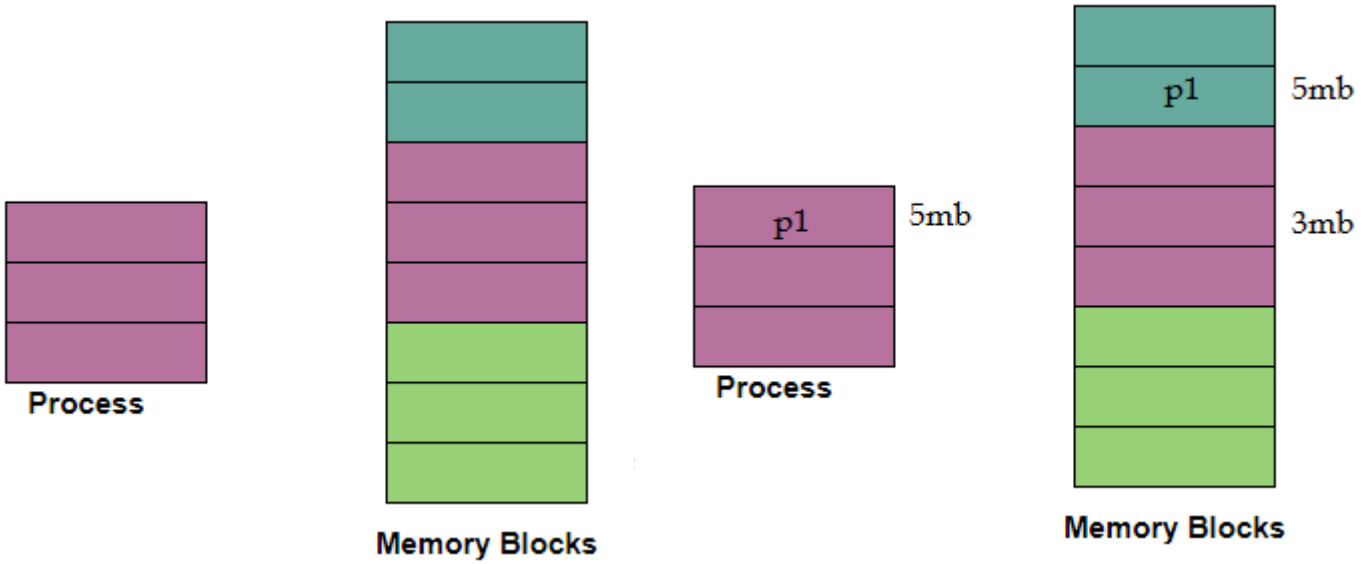
- It helps the CPU to manage multiple processes within a single main memory.
- It helps to create and use virtual memory.

Disadvantages of Swapping

- If the computer system loses power, the user may lose all information related to the program in case of substantial swapping activity.
- If the swapping algorithm is not good, the composite method can increase the number of Page Fault and decrease the overall processing performance.

Contiguous Memory Allocation:

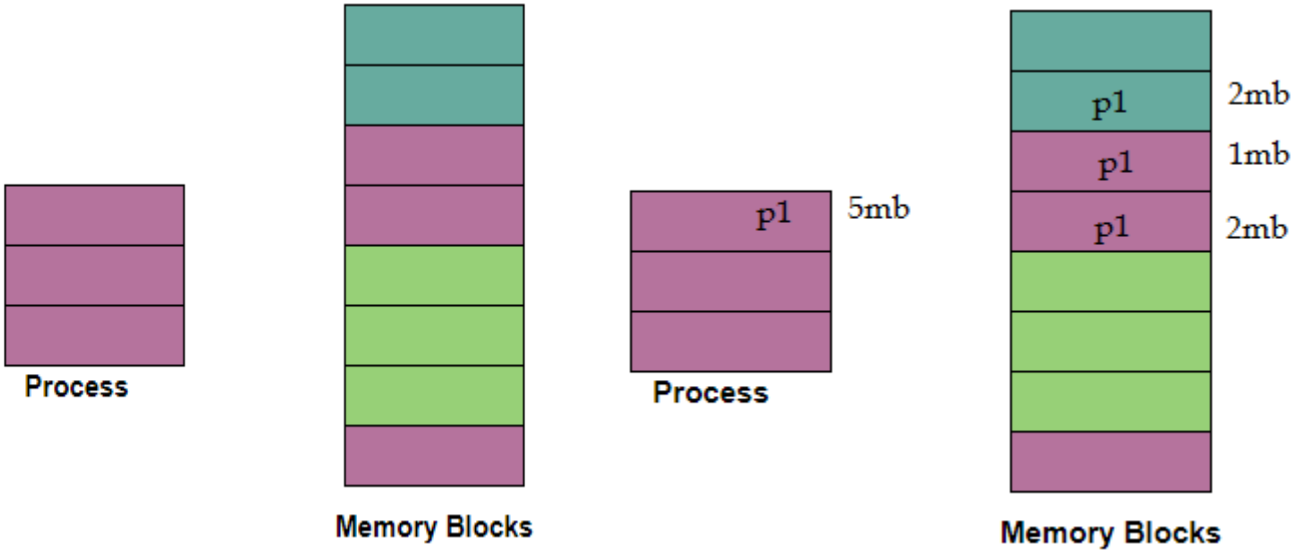
In **Contiguous Memory Allocation** whenever any user process request for the memory then a single section of the contiguous memory block is allocated to that process according to the requirements of the process. Contiguous memory allocation is achieved just by **dividing the memory** into the **fixed-sized partition**.



Contiguous Memory Allocation:

- **Non-Contiguous Memory Allocation**

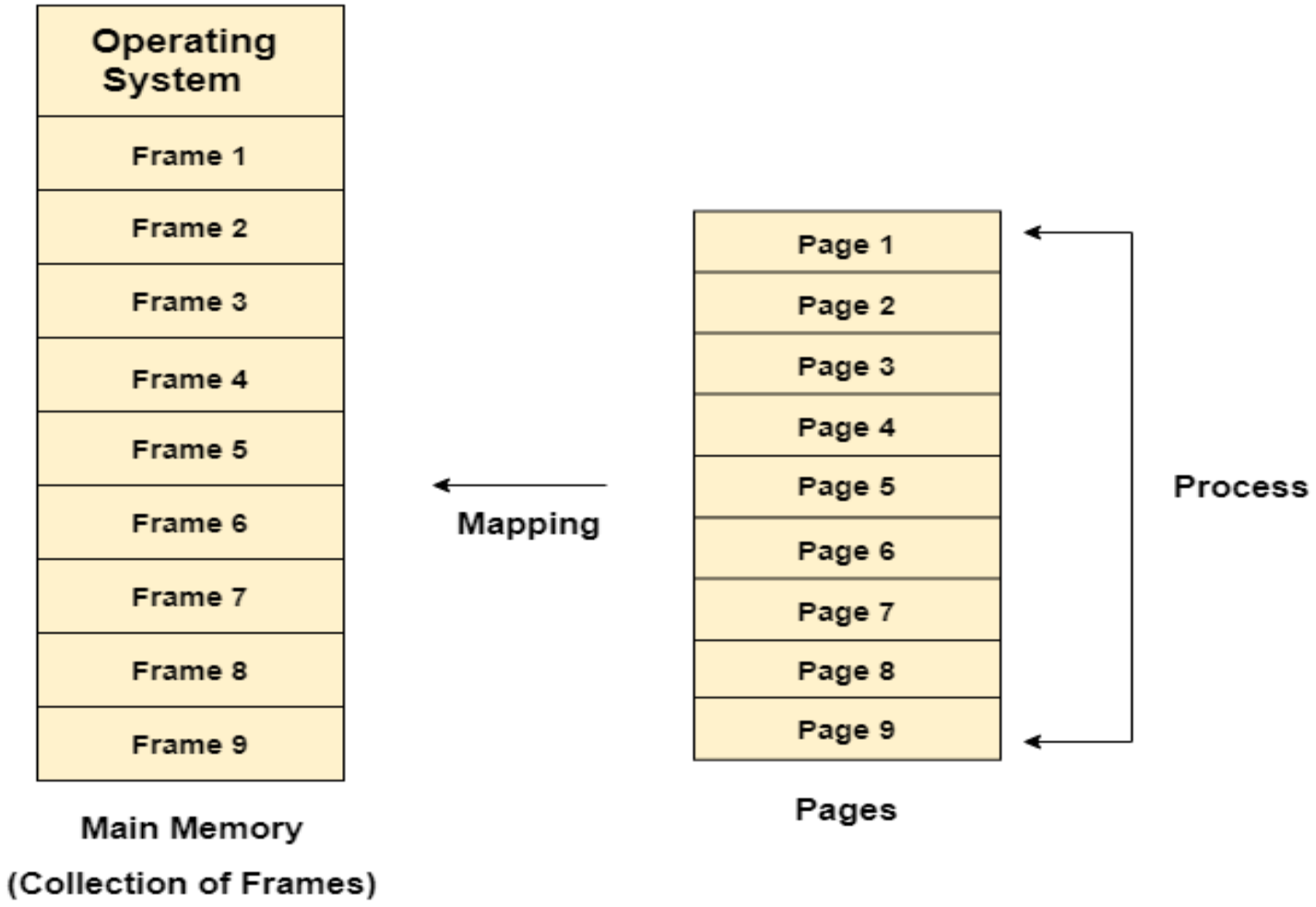
With the help of Non-contiguous memory allocation, a process is allowed to **acquire several memory blocks at different locations in the memory** according to its need.



Paging:

- **A process can be divided into no.of partitions called pages. All pages should coantin equal size is called logical memory. Paging** is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages.
- 5MThere are 4 separate processes in the system that is A1, A2, A3, and A4 of 4 KB each. Here, all the processes are divided into pages of 1 KB each so that operating system can store one page in one frame.

Paging:


Operating System
Frame 1
Frame 2
Frame 3
Frame 4
Frame 5
Frame 6
Frame 7
Frame 8
Frame 9
Main Memory
(Collection of Frames)
Mapping
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Page 7
Page 8
Page 9
Pages
Process

Paging:

The logical address has two parts.

- Page Number
- Offset

where,

- **Page Number** is used to specify the specific page of the process from which the CPU wants to read the data. and it is also used as an index to the page table.
- and **Page offset** is mainly used to specify the specific word on the page that the CPU wants to read.
- Page Table in OS

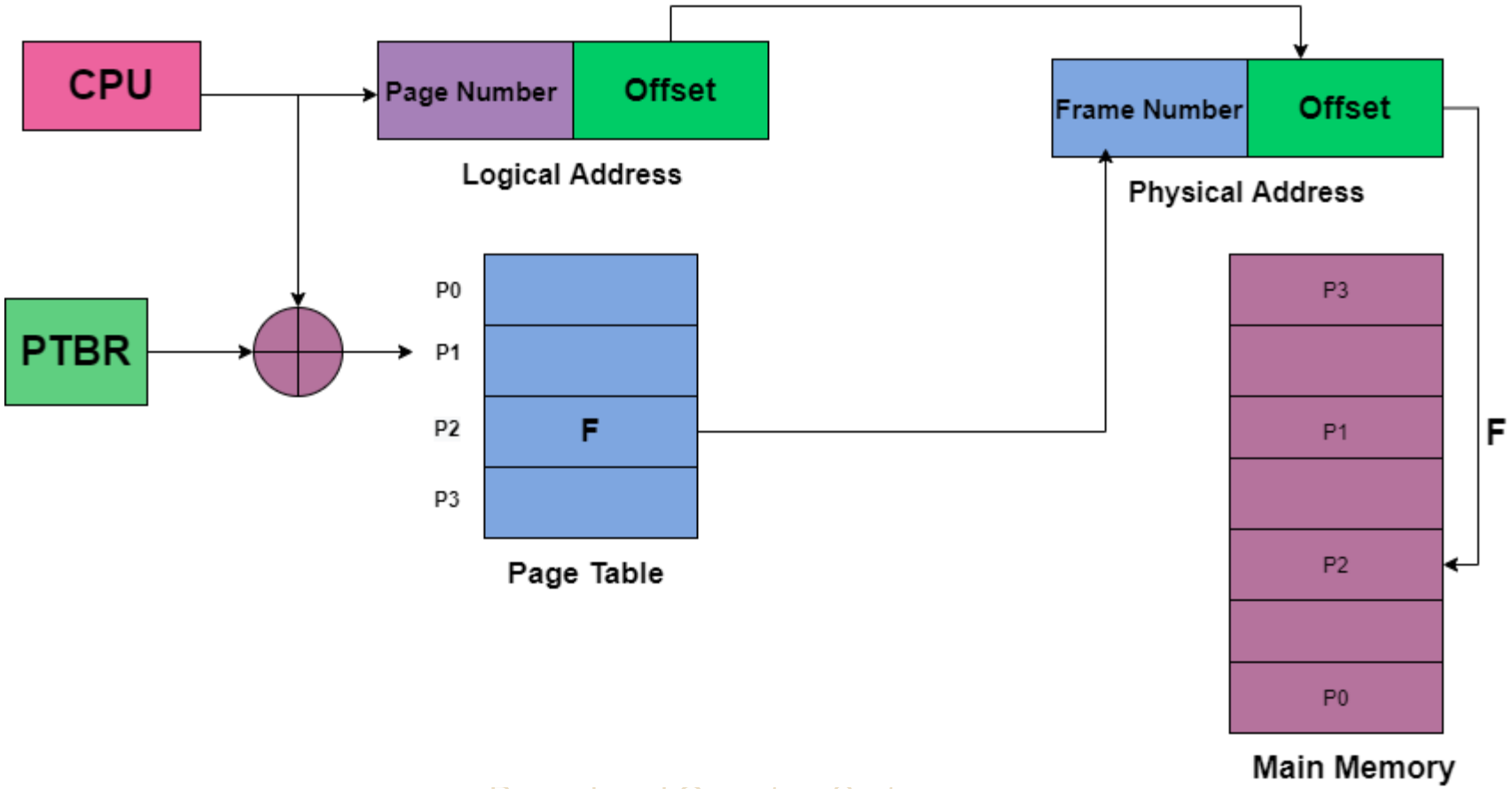
Paging:

The physical address consists of two parts:

- Page offset(d)
- Frame Number(f)

- where,
  - The **Frame number** is used to indicate the specific frame where the required page is stored.
  - and **Page Offset** indicates the specific word that has to be read from that page.

Paging:


CPU
Page Number
Offset
Logical Address
Frame Number
Offset
Physical Address
PTBR
P0
P1
P2
P3
F
Page Table
P3
P1
P2
P0
F
Main Memory

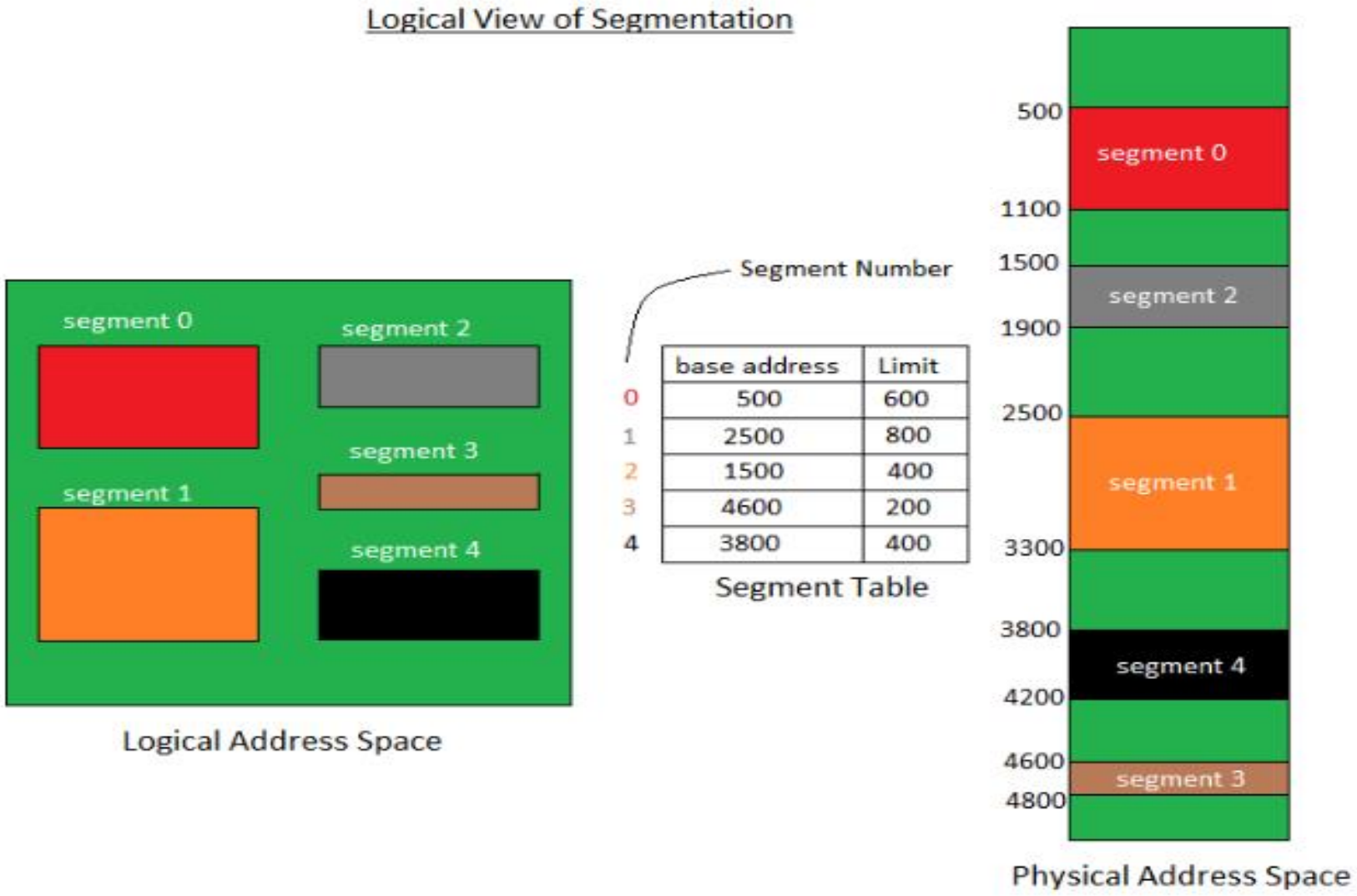
Segmentation:

- In Operating Systems, Segmentation is a memory management technique in which, the memory is divided into the variable size parts. Each part is known as segment which can be allocated to a process.
- The details about each segment are stored in a table called as segment table. Segment table is stored in one (or many) of the segments.

Segment table contains mainly two information about segment:

- Base: It is the base address of the segment
- Limit: It is the length of the segment.

Segmentation:



Virtual memory:

- The processes size is more than the main memory in such case virtual memory is needed to store processes.
- **Virtual Memory** is a storage mechanism which offers user an illusion of having a very big main memory. It is done by treating a part of secondary memory as the main memory. In Virtual memory, the user can store processes with a bigger size than the available main memory.

Virtual memory:

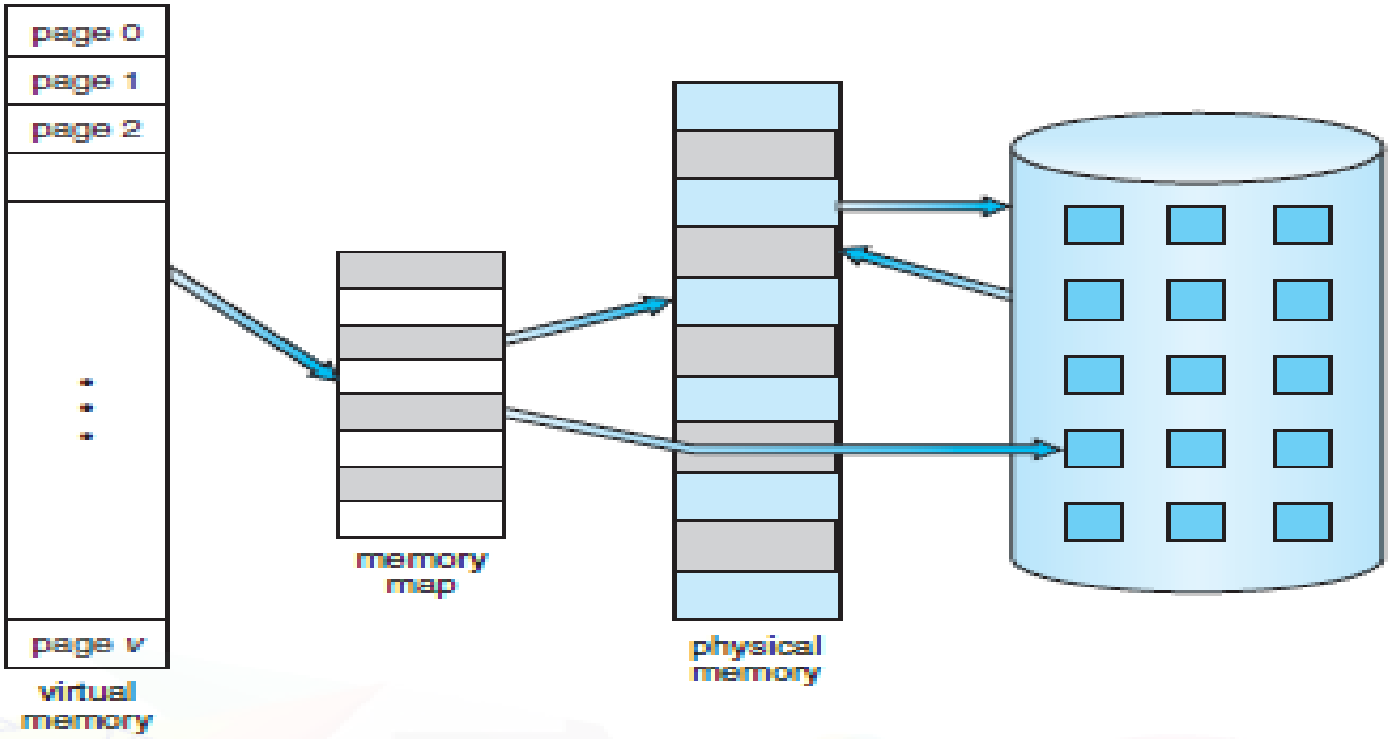
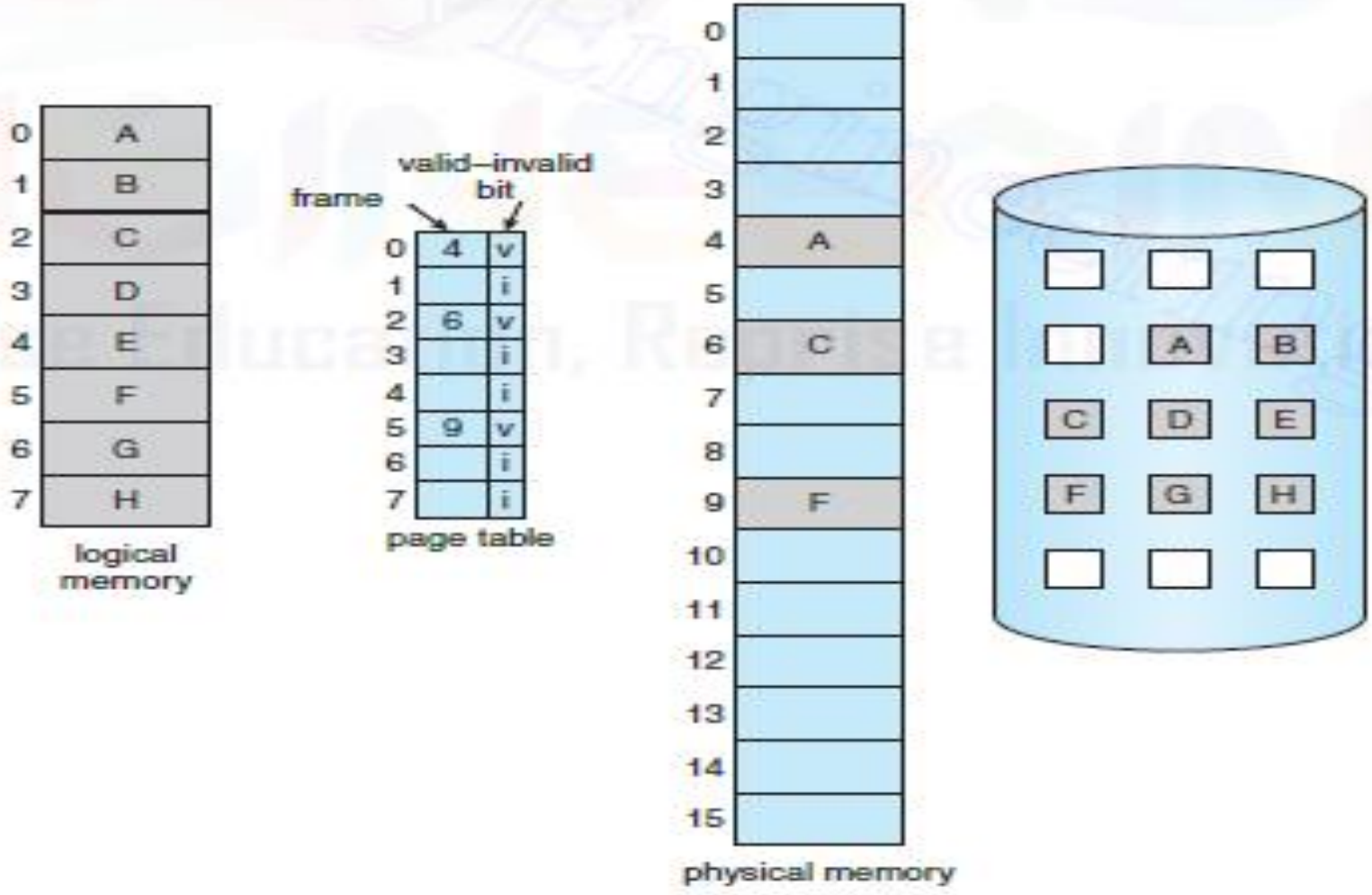


Diagram showing virtual memory that is larger than physical memory.

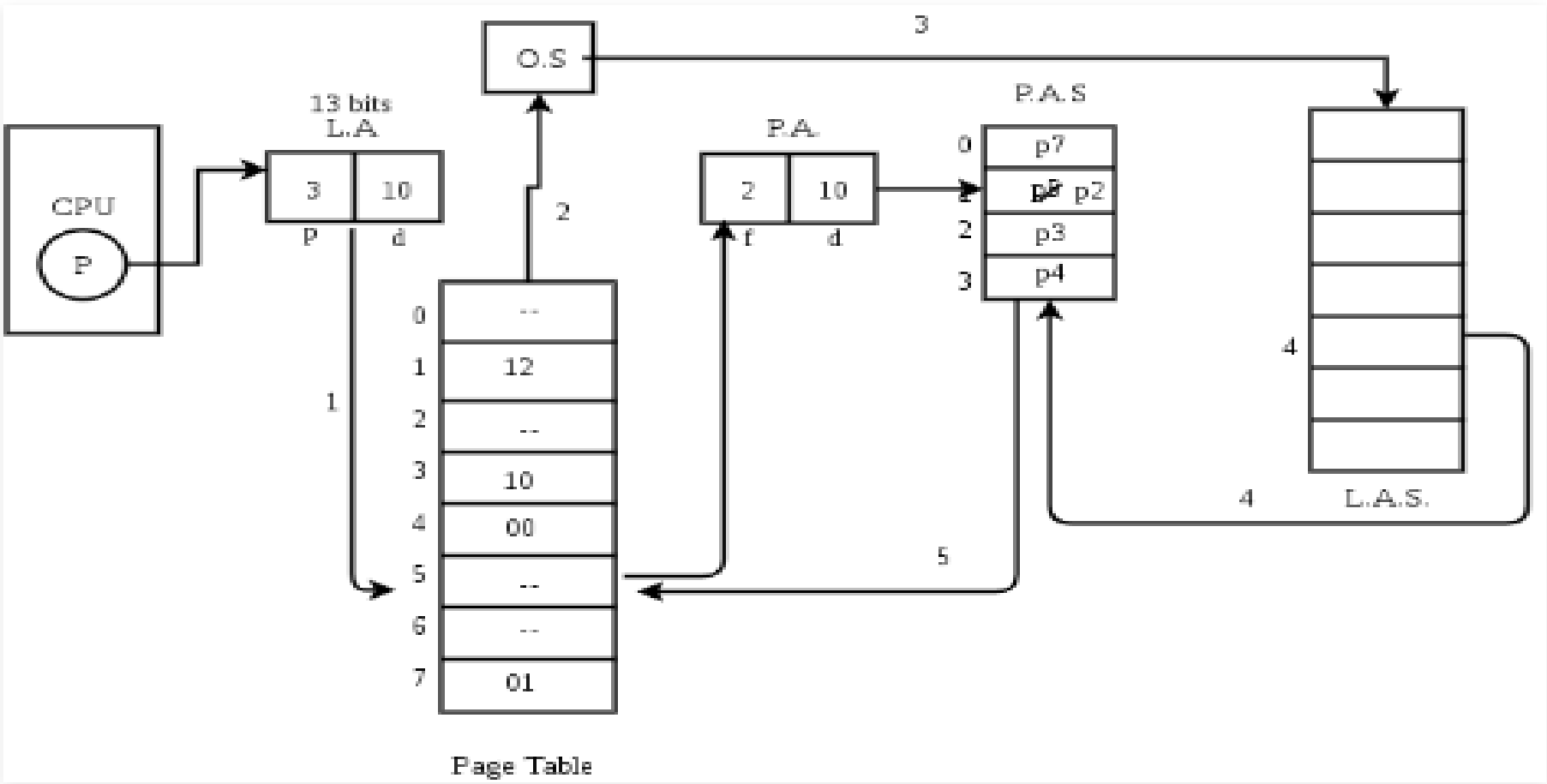
Virtual memory:



Page table when some pages are not in main memory.

Virtual memory:

The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.

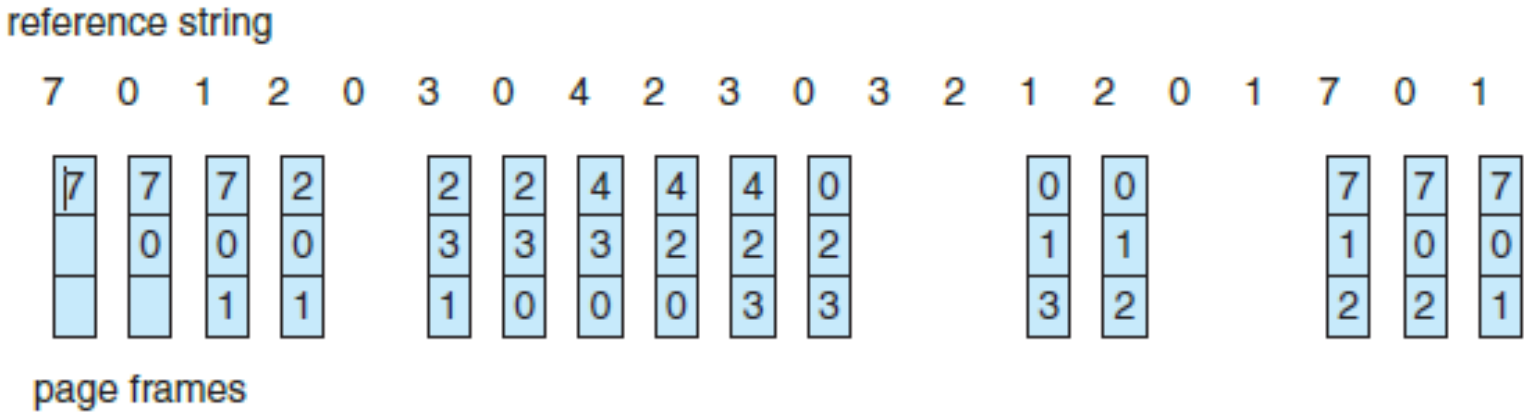


Page Replacement Algorithms:

A page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

Page Fault – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

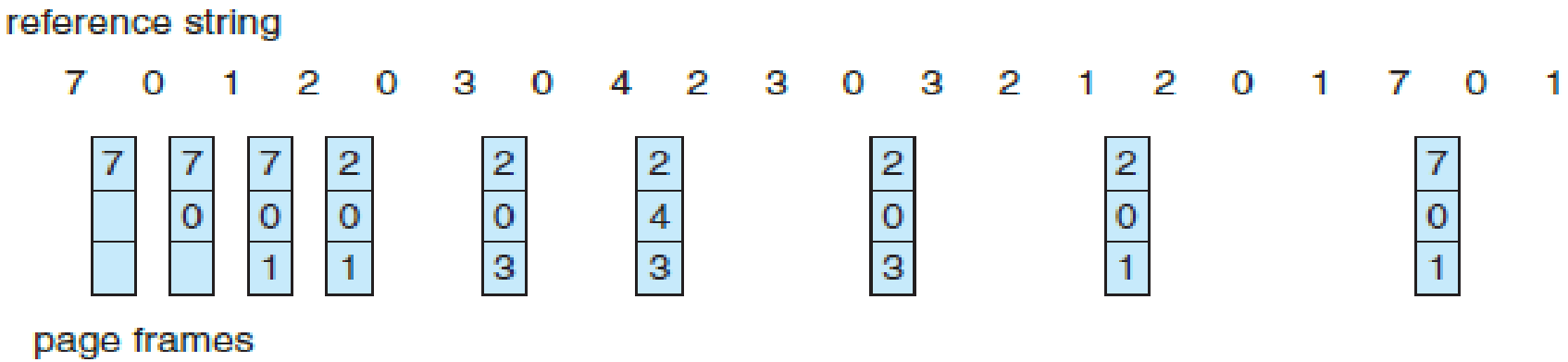
First In First Out (FIFO) –



FIFO page-replacement algorithm.

Page Replacement Algorithms:

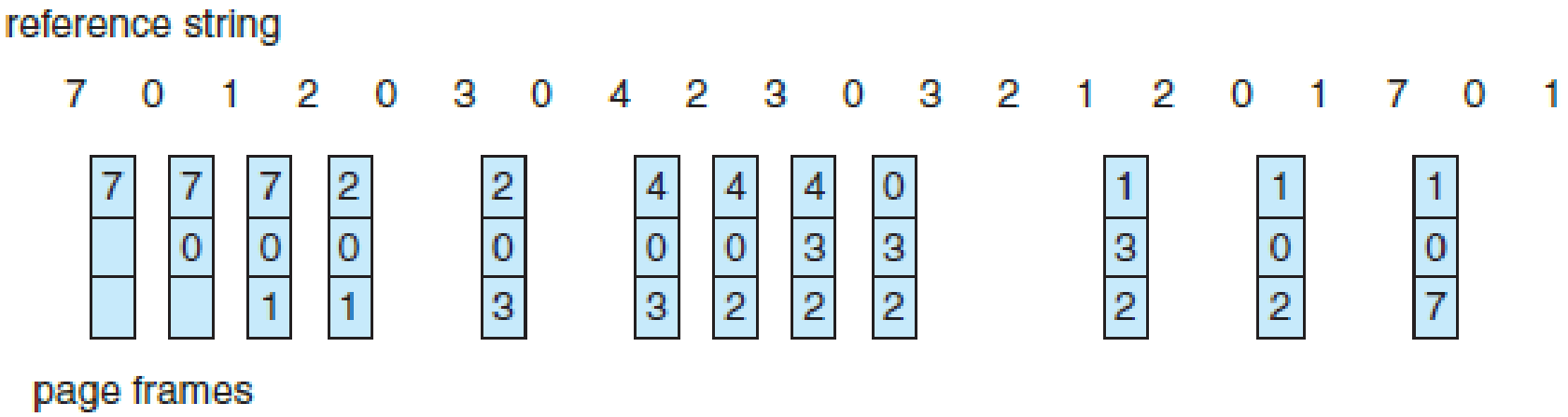
Optimal Page replacement:



Optimal page-replacement algorithm.

Page Replacement Algorithms:

Least recent used (LRU):

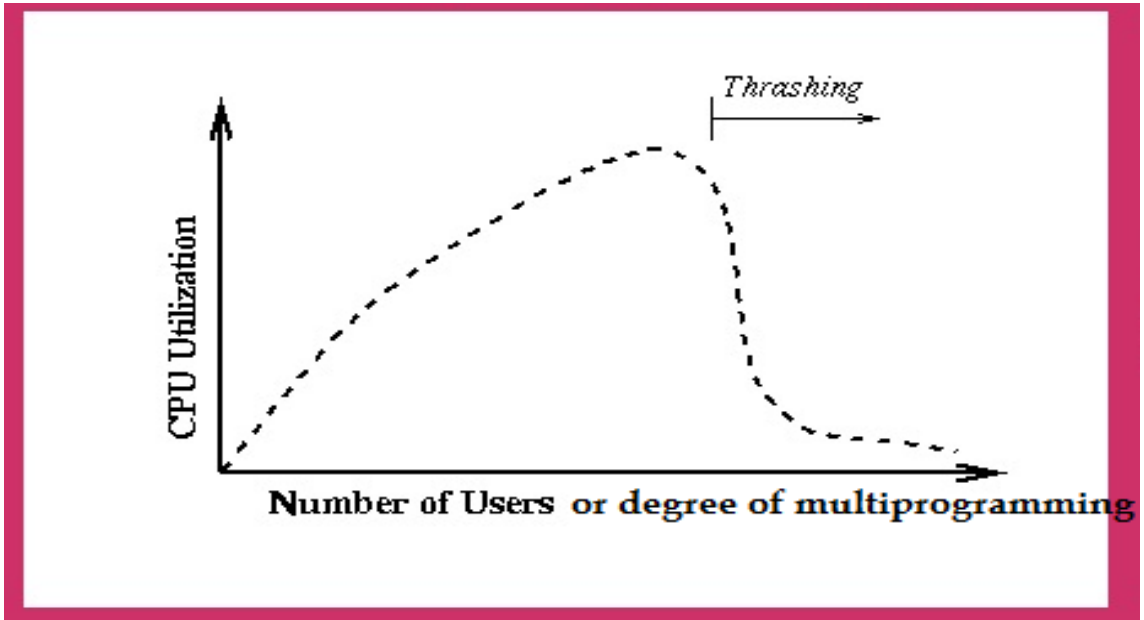


LRU page-replacement algorithm.

Thrashing:

- Thrashing occurs when a **Process** is spending more time in paging or **Swapping** activities rather than its execution.
- In Thrashing, the state CPU is so much busy swapping that it cannot respond to the user program as much as it required.

As the degree of multiprogramming increases to increase the CPU Utilization, it causes Thrashing after some time. This is as shown in the above picture.



Disk scheduling:

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

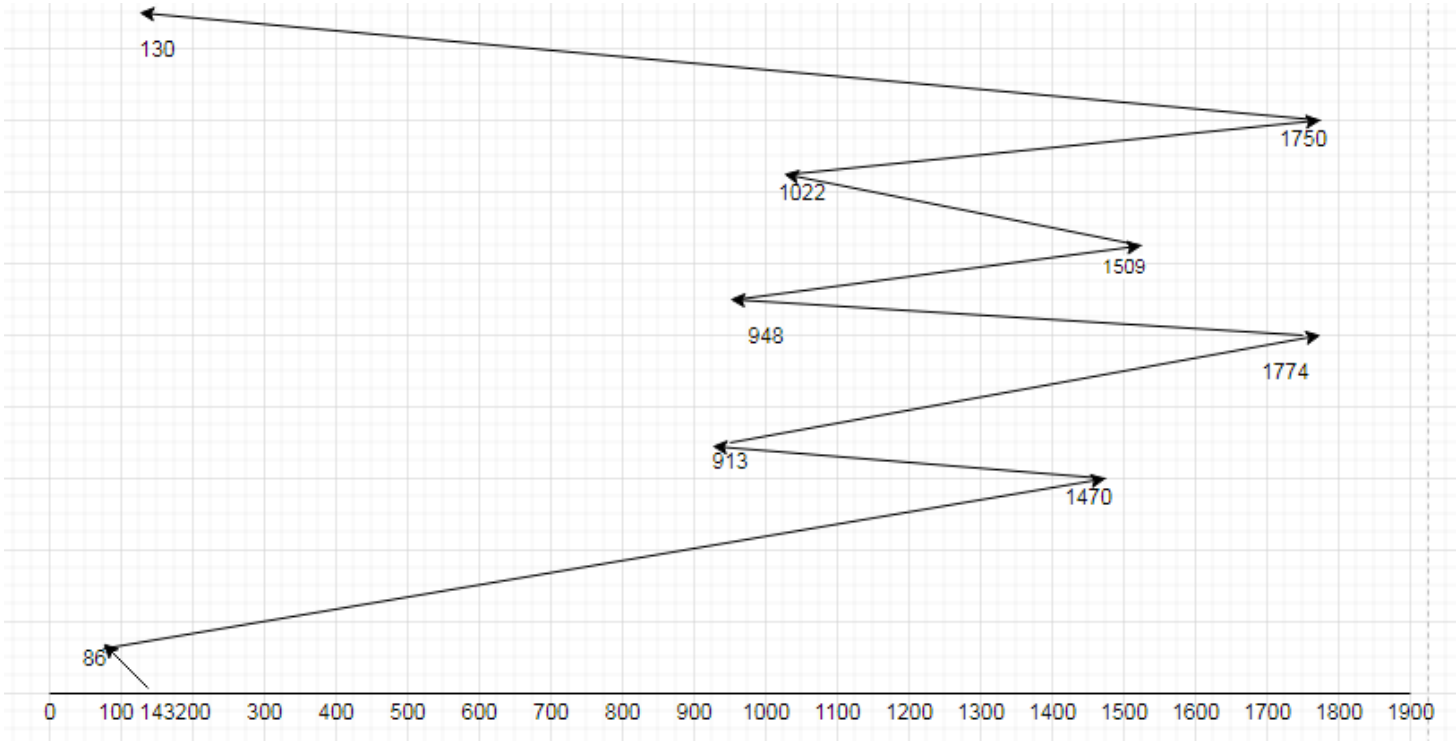
- Seek Time:
- Rotational Latency:
- Transfer Time:
- Disk Access Time: Disk Response Time:

Disk scheduling:

- **Disk Scheduling Algorithms:**
  - FCFS scheduling algorithm
  - SSTF (shortest seek time first) algorithm
  - SCAN scheduling
  - C-SCAN scheduling
  - LOOK Scheduling

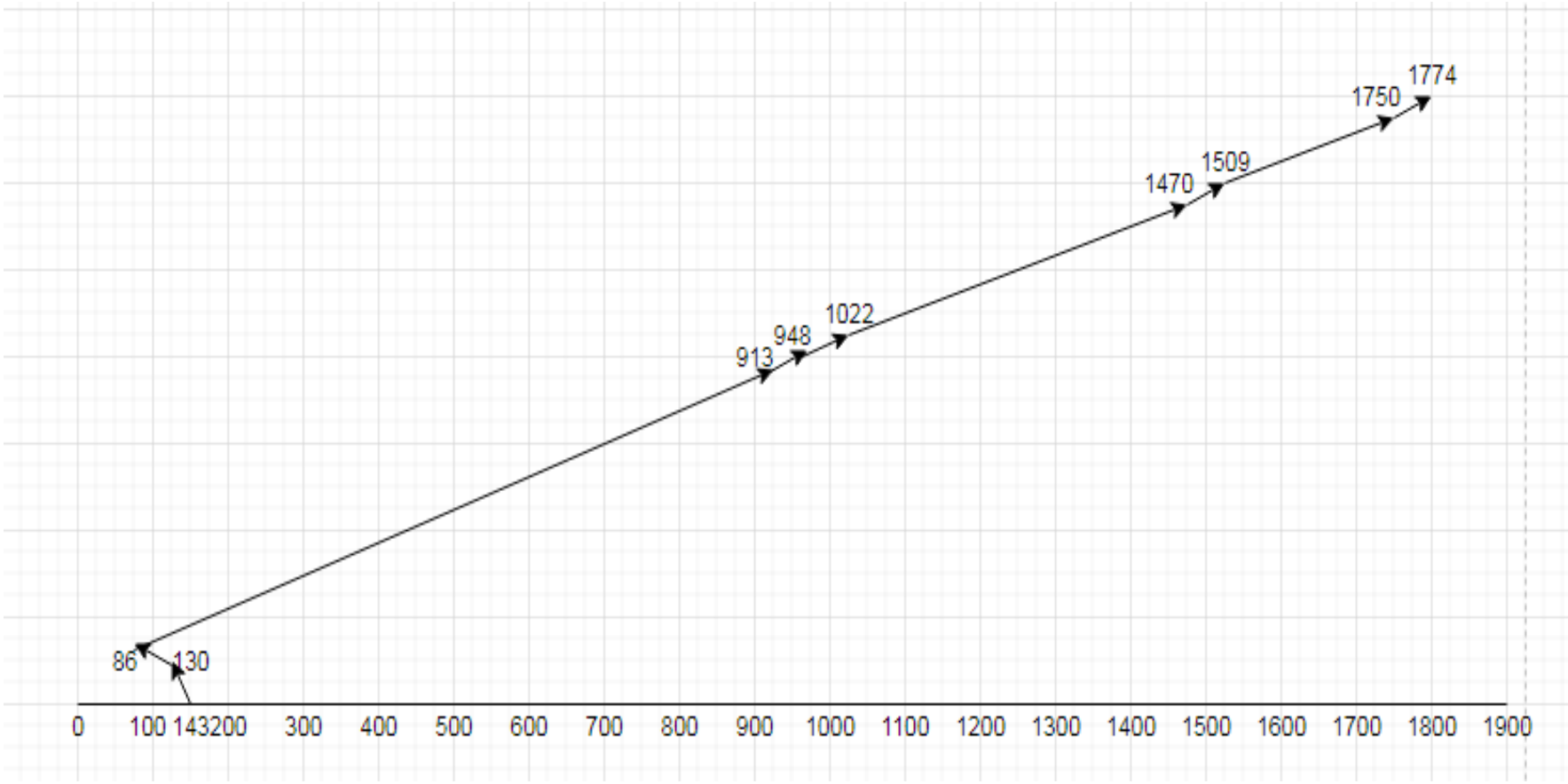
Disk scheduling:

- FCFS scheduling algorithm:



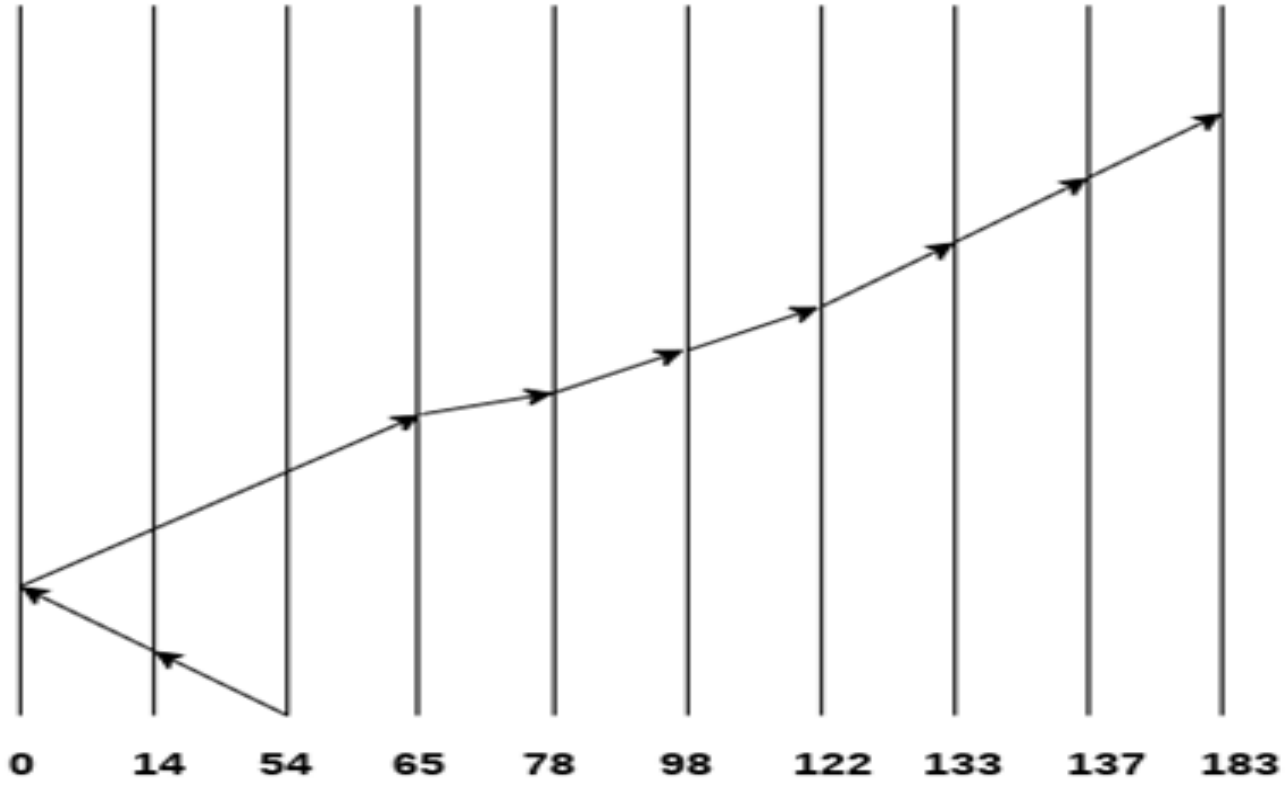
Disk scheduling:

- **SSTF:**



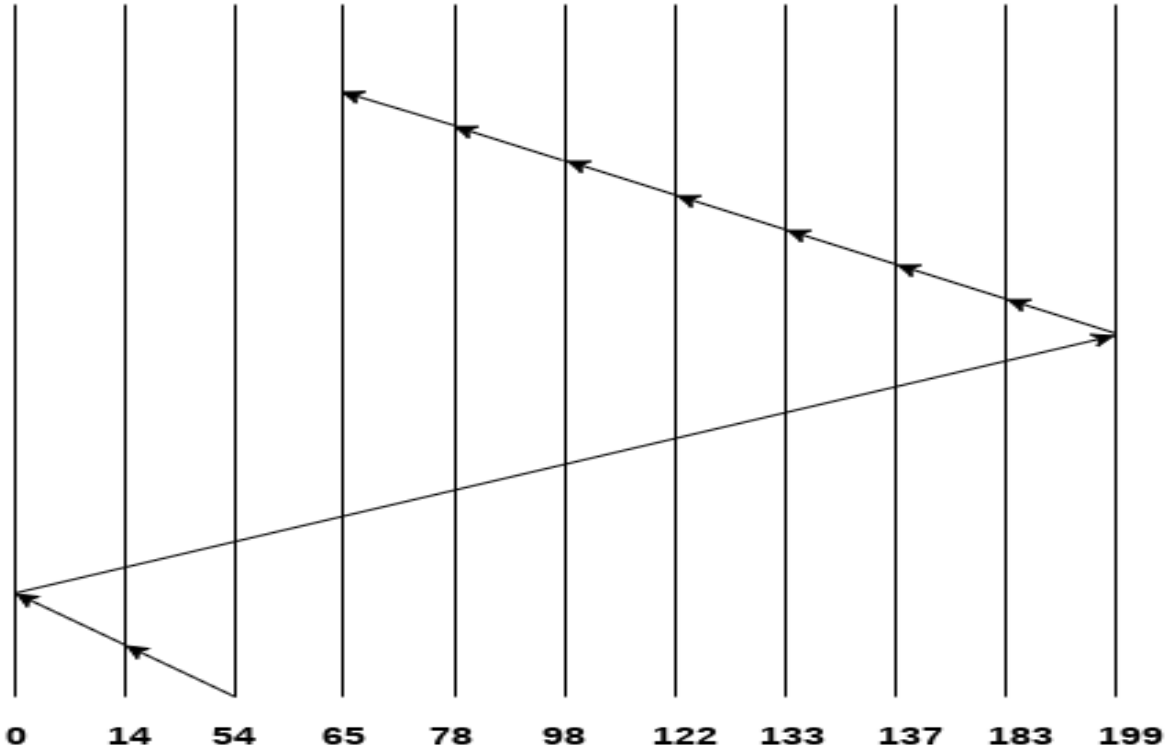
Disk scheduling:

- Scan Algorithm:



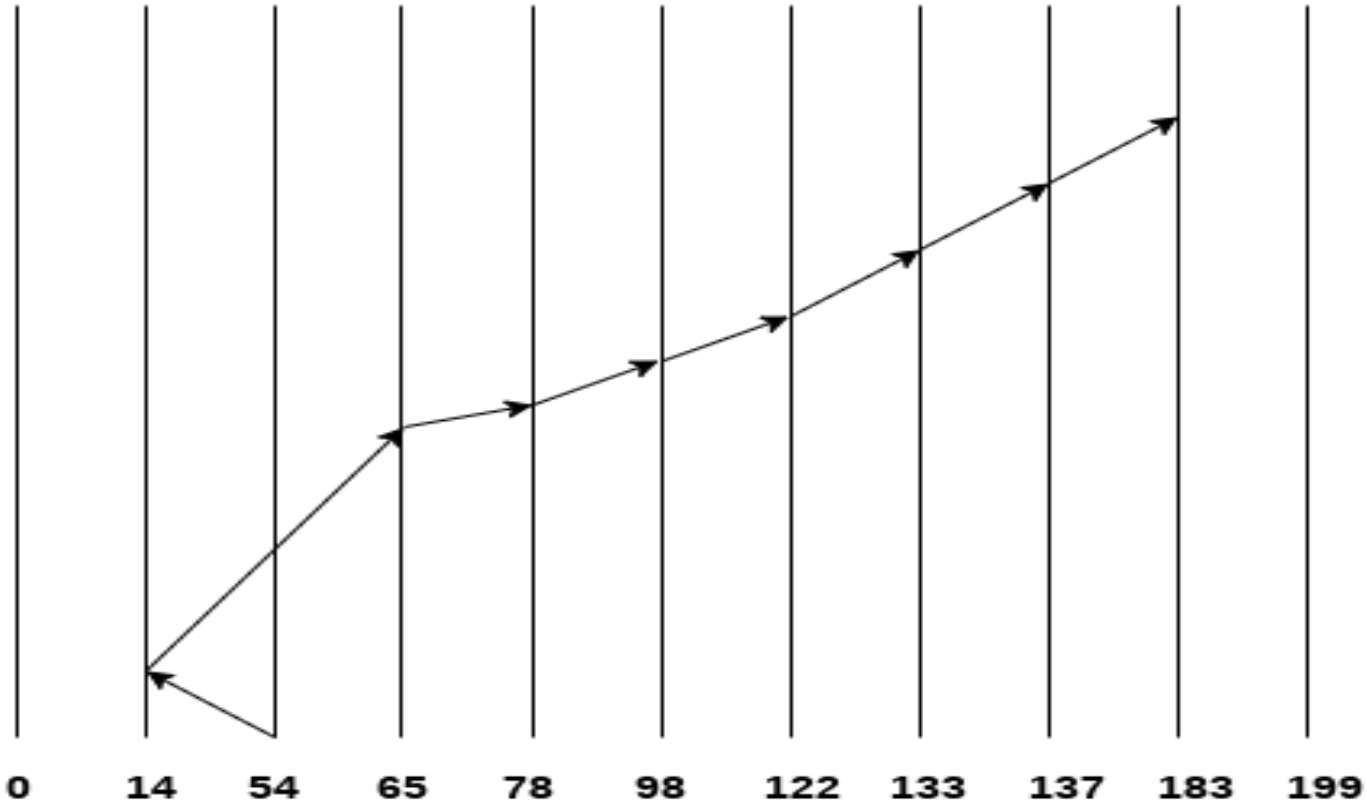
Disk scheduling:

- C-SCAN algorithm:



Disk scheduling:

- **Look Scheduling**



SIDDARTHA INSTITUTE OF SCIENCE AND TECHNOLOGY
(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS

(20CS0507)

COURSE OBJECTIVES:

1. Explain main components of an OS & their functions.

2. Describe the process management and scheduling.

3. Discuss various issues in Inter Process Communication (IPC) and the role of OS in IPC.

4. Illustrate the concepts and implementation of Memory management policies and virtual memory

5. Explain working of an OS as a resource manager, file system manager, process manager, memory manager and I/O manager and methods used to implement the different parts of OS.

COURSE OUTCOMES (CO's):

On successful completion of the course, the student will be able to

1. Describe the important computer system resources and the role of operating system in their management policies and algorithms.
2. Understand the process management policies and scheduling of processes by CPU.
3. Analyze the requirement for process synchronization and coordination handled by operating system.
4. Describe and analyze the memory management and its allocation policies. Technologies
5. Categorize the storage management policies with respect to different storage management technologies
6. Study the need for special purpose operating system with the advent of new emerging technologies

Topics To Be Covered:

File Management:

Concept of File- Access methods-File types-File operation-Directory structure-File System structure- Allocation methods (contiguous, linked, indexed)- Freespace management (bit vector, linked list, grouping)

Protection & Security:

Protection Mechanisms- Protection matrix- Authentication Techniques- Threats-intruders-Basics of Cryptography-Secret key-public key- One-Way Function-Digital

File concept:

- A file is named collection of related info. That is recorded on secondary storage.
- An executable file is a series of code sections that the loader can bring into the memory & execute.
- **File attributes**
  - Name: The symbolic file name is the only information kept in human readable form.
  - Identifier: This unique tag, usually a number, identifies the file within the file system;
  - it is the non-human-readable name for the file.

File concept:

- **File Operations:**
  - Creating a file
  - Writing a file. Reading a file.
  - Repositioning within a file.
  - Deleting a file.
  - Truncating a file.

File Types:

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

Figure 11.3 Common file types.

Access Methods:

- **1. Sequential Access:**

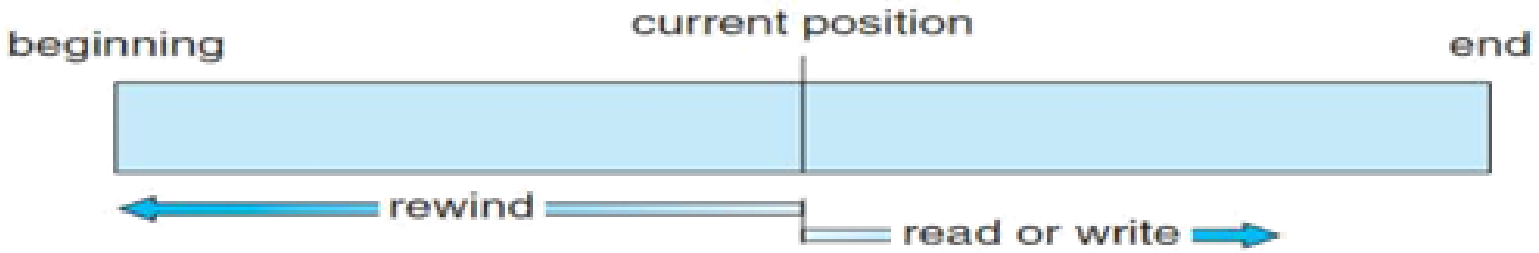


Figure 11.4 Sequential-access file.

- **2. Direct Access:**

sequential access	implementation for direct access
reset	cp = 0;
read_next	read cp ; cp = cp + 1;
write_next	write cp; cp = cp + 1;

Figure 11.5 Simulation of sequential access on a direct-access file.

File Structure:

- Some files contain an internal structure, which may or may not be known to the OS.
- For the OS to support particular file formats increases the size and complexity of the OS.
- UNIX treats all files as sequences of bytes, with no further consideration of the internal structure.
- Macintosh files have two ***forks*** - a ***resource fork***, and a ***data fork***.
- The resource fork contains information relating to the UI, such as icons and button images, and can be modified independently of the data fork, which contains the code or data as appropriate.

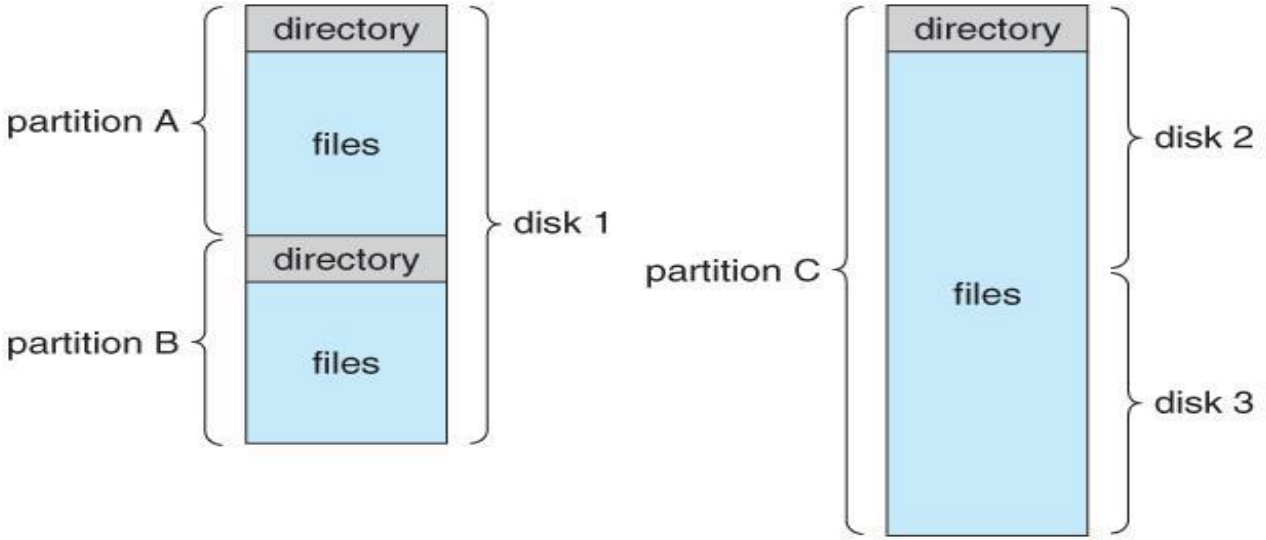
File Structure:

- **Internal File Structure**
  - Disk files are accessed in units of physical blocks, typically 512 bytes or some power-of-two multiple thereof.
  - Internally files are organized in units of logical units, which may be as small as a single byte, or may be a larger size corresponding to some data record or structure size.

Directory Structure:

Storage Structure

A disk can be used in its entirety for a file system. Alternatively a physical disk can be broken up into multiple ***partitions, slices, or mini-disks,*** each of which becomes a virtual disk and can have its own filesystem.



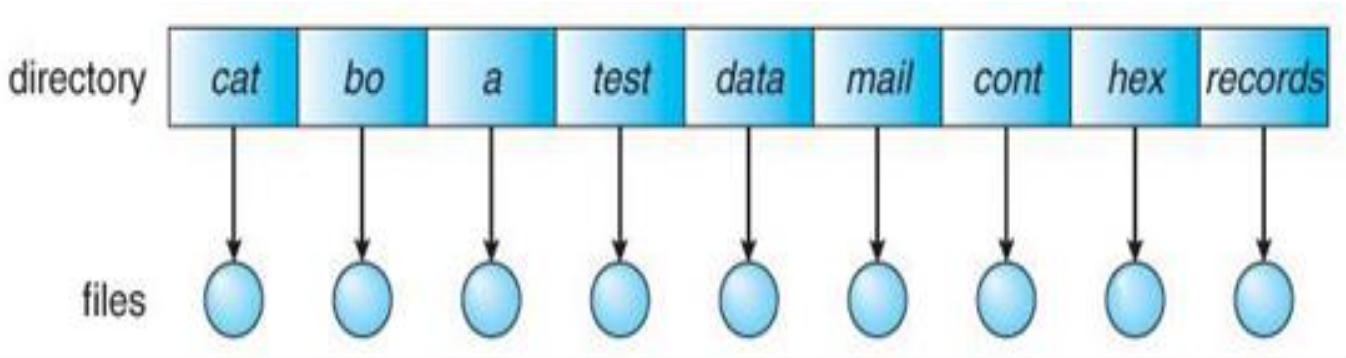
Directory Structure:

Directory Overview

- Directory operations to be supported include:
- Search for a file
- Create a file - add to the directory
- Delete a file - erase from the directory
- List a directory - possibly ordered in different ways.
- Rename a file - may change sorting order
- Traverse the file system.

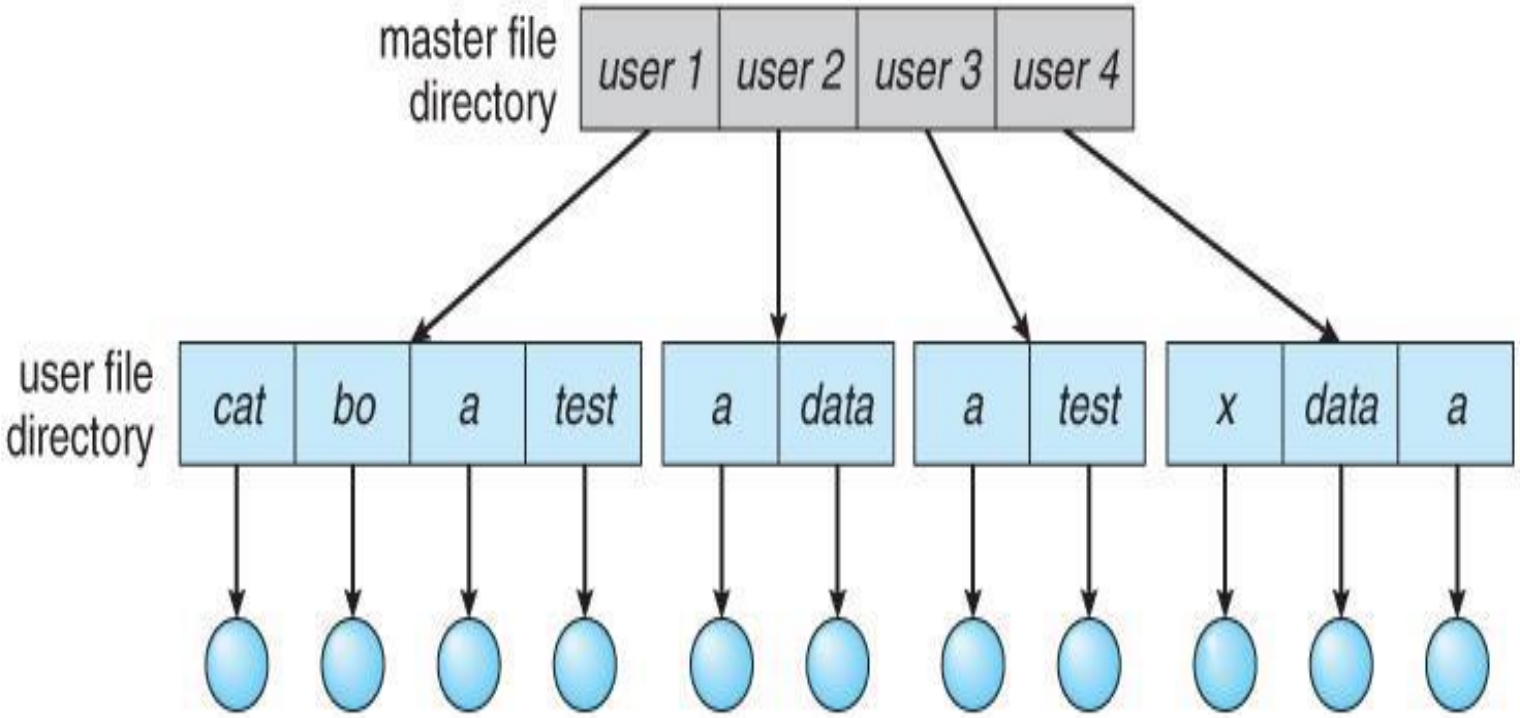
Directory Structure:

Single-Level Directory



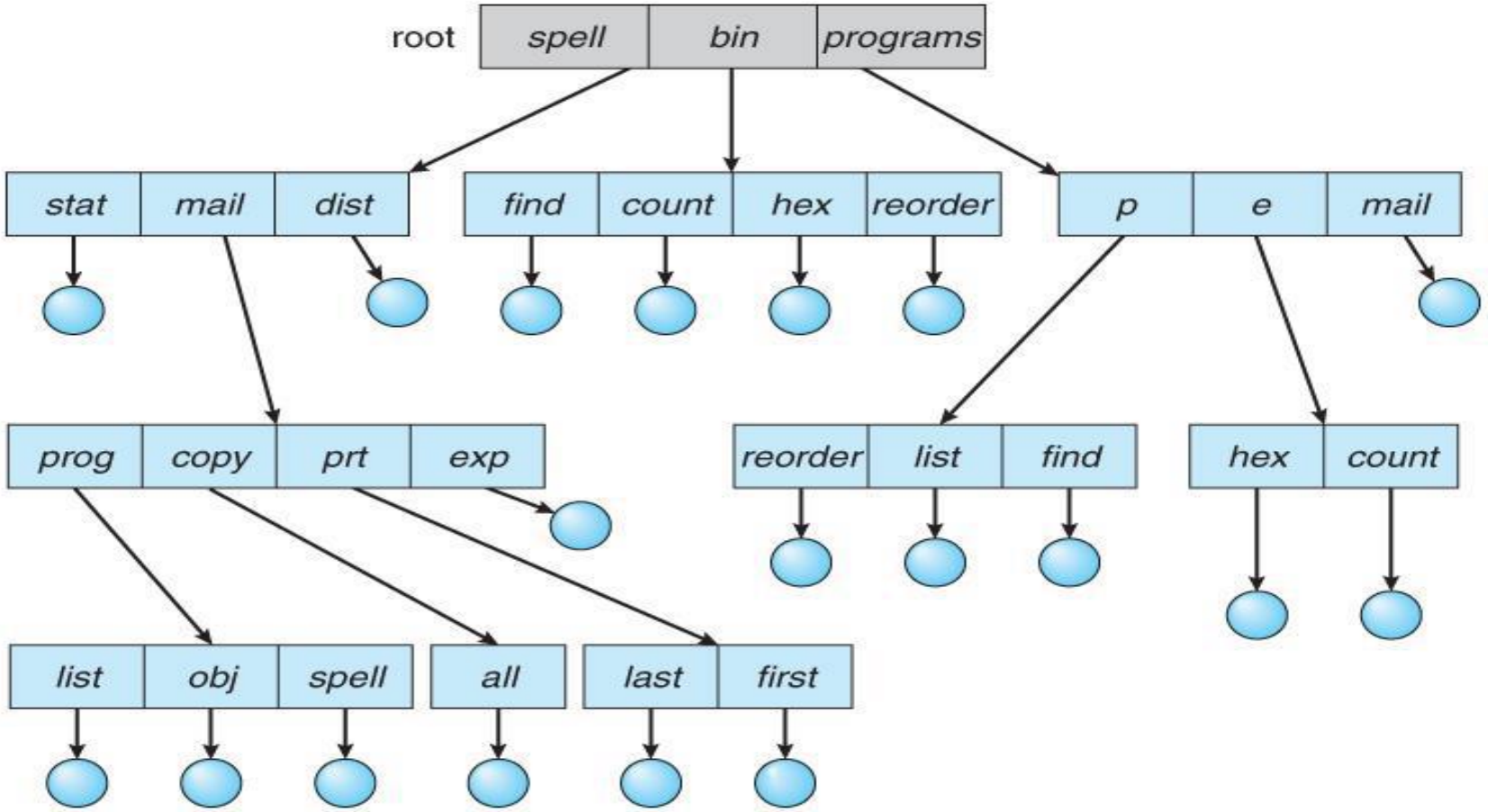
Directory Structure:

Two-Level Directory



Directory Structure:

Tree-Structured Directories

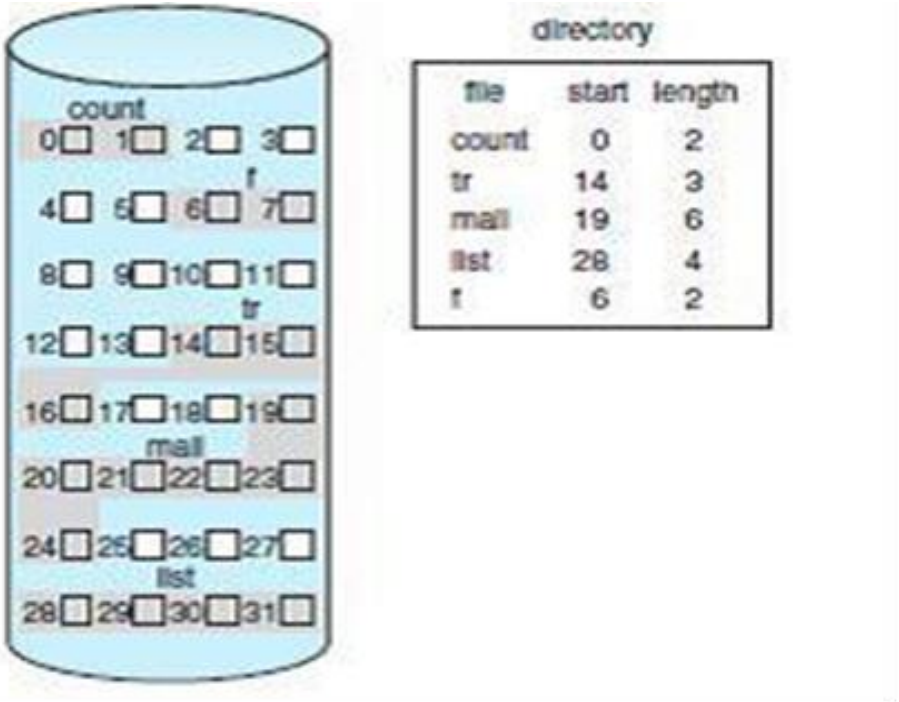


Allocation Methods:

Three major methods of allocating disk spaces are

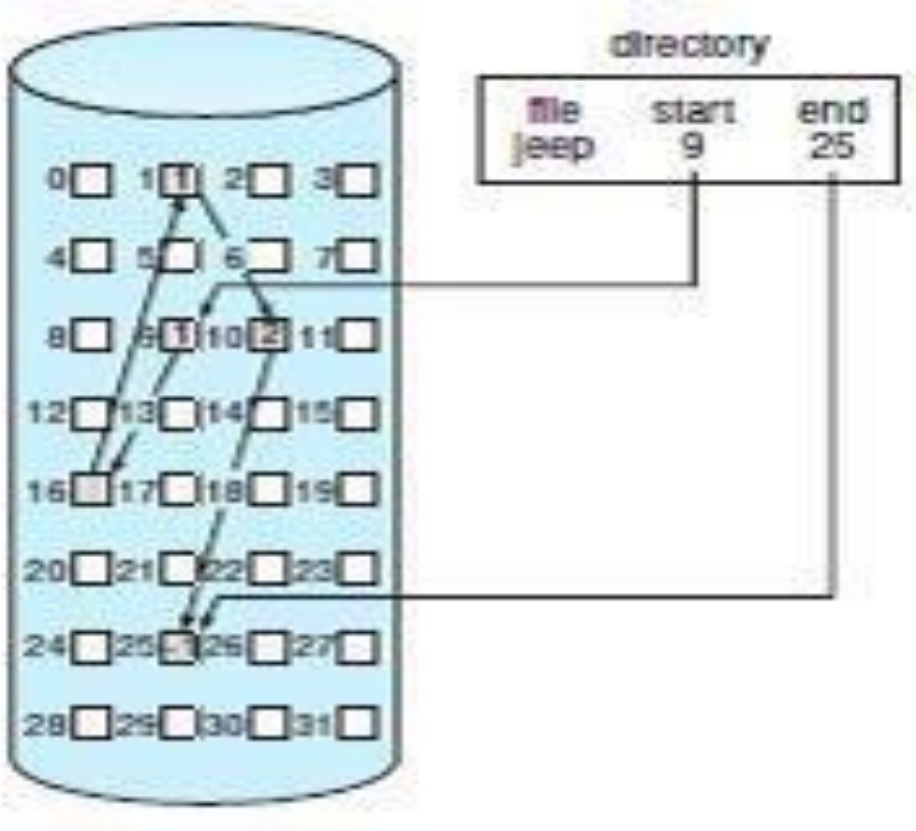
- Contiguous allocation
- Linked allocation
- Indexed allocation

Contiguous Allocation



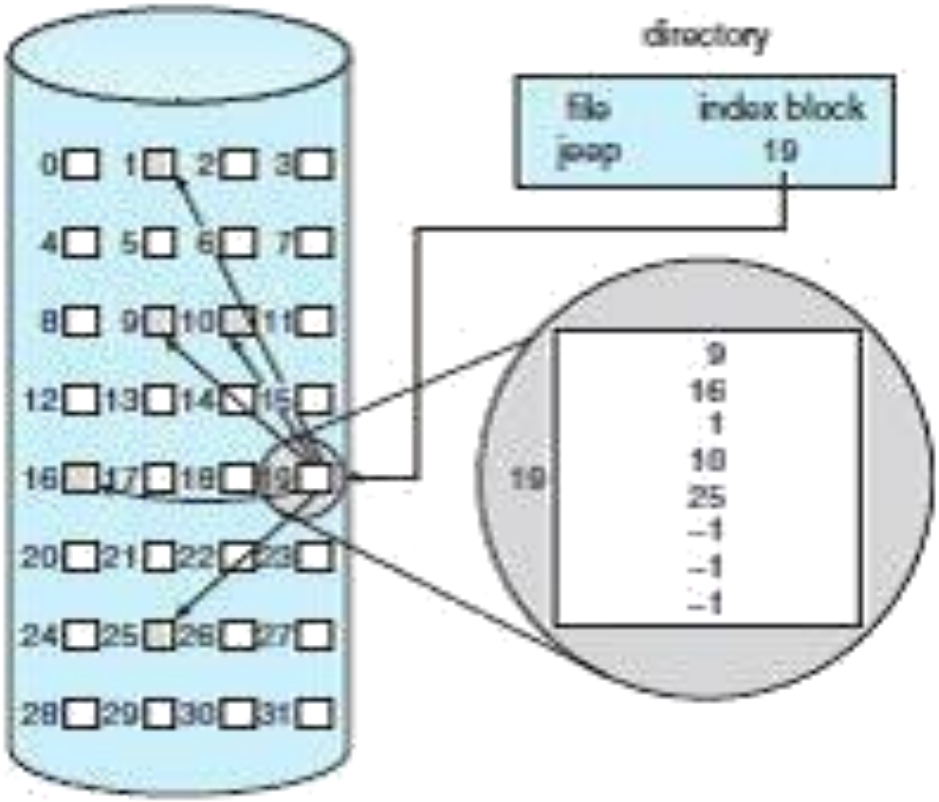
Allocation Methods:

Linked Allocation



Allocation Methods:

Indexed Allocation



Free Space Management:

The mechanisms used for free space management are:

- **Bit Vector:**
  - Frequently, the free-space list is implemented as a bit **map** or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.
  - For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25,26, and 27 are free and the rest of the blocks are allocated.

Free Space Management:

- **Linked List:**
  - Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
  - This first block contains a pointer to the next free disk block, and so on. In our earlier example, we would keep a pointer to block 2 as the first free block. Block *2* would contain a pointer to block3, which would point to block 4, which would point to block 5, which would point to block 8, and so on

Free Space Management:

- **Grouping:**
  - A modification of the free-list approach is to store the addresses of *n* free blocks in the first free block. The first n−1 of these blocks is actually free.
  - The last block contains the addresses of another *n* free block, and so on.
  - The addresses of a large number of free blocks can now be found quickly, unlike the situation when the standard linked-list approach is used.

Protection Mechanism:

- **Goals of Protection**
  - To ensure that each shared resource is used only in accordance with system *policies,* which may be set either by system designers or by system administrators.
- **Principles of Protection**
  - The ***principle of least privilege*** dictates that programs, users, and systems be given just enough privileges to perform their tasks.
  - This ensures that failures do the least amount of harm and allow the least of harm to be done.

Protection Mechanism:

- **Domain of Protection**
  - A computer can be viewed as a collection of *processes* and *objects* (both HW & SW).
  - The ***need to know principle*** states that a process should only have access to those objects it needs to accomplish its task, and furthermore only in the modes for which it needs access and only during the time frame when it needs access.

Access Matrix:

The model of protection that we have been discussing can be viewed as an ***access matrix,*** in which columns represent different system resources and rows represent different protection domains. Entries within the matrix indicate what access that domain has to that resource.

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Authentication:

- Authentication involves verifying the identity of the entity that transmitted a message.
- For example, if D (Kd) (c) produces a valid message, then we know the sender was in possession of E (Ke).
- ***User Authentication***
  - *Protection, dealt with making sure that only certain users were allowed to perform certain tasks, i.e. that a users privileges were dependent on his or her identity. But how does one verify that identity to begin with?*

Authentication:

- ***User Authentication***
  - *1 Passwords*
  - 2 Password Vulnerabilities
  - 3 Encrypted Passwords
  - 4 One-Time Passwords
  - 5 Biometrics

Threats:

- From a security perspective, computer systems have four general goals, with corresponding threats to them, as listed in Fig.

Goal	Threat
Data confidentiality	Exposure of data
Data integrity	Tampering with data
System availability	Denial of service
Exclusion of outsiders	System takeover by viruses

Threats:

- The first, ***data confidentiality***, is concerned with having secret data remain secret. More specifically, if the owner of some data has decided that these data are only to be made available to certain people and no others, the system should guarantee that release of the data to unauthorized people never occurs.
- The second goal, **data integrity,** means that unauthorized users should not be able to modify any data without the owner's permission.

Intruders:

- people who are nosing around places where they have no business being are called **intruders** or sometimes **adversaries.**
- Some common categories are:
  - ***Casual prying by nontechnical users***
  - ***Snooping by insiders***
  - ***Determined attempts to make money***
  - ***Commercial or military espionage***

Accidental Data Loss:

In addition to threats caused by malicious intruders, valuable data can be lost by accident. Some of the common causes of accidental data loss are

- 1. ***Acts of God:*** fires, floods, earthquakes, wars, riots, or rats gnawing backup tapes.
- 2. ***Hardware or software errors:*** CPU malfunctions, unreadable disks or tapes, telecommunication errors, program bugs.
- 3. ***Human errors:*** incorrect data entry, wrong tape or CD-ROM mounted, wrong program run, lost disk or tape, or some other mistake.

Digital Signatures:

- Frequently it is necessary to sign a document digitally. For example, suppose a bank customer instructs the bank to buy some stock for him by sending the bank an e-mail message.
- Digital signatures make it possible to sign e-mails and other digital documents in such a way that they cannot be repudiated by the sender later.
- One common way is to first run the document through a one-way cryptographic hashing algorithm that is very hard to invert.

Digital Signatures:

- The most popular hashing functions used are **MD5 (Message Digest 5),** which produces a 16-byte result (Rivest, 1992) and **SHA-1 (Secure Hash Algorithm),** which produces a 20-byte result (NIST, 1995).
- Newer versions of SHA-1 are **SHA-256** and **SHA-512,** which produce 32-byte and 64- byte results, respectively, but they are less widely used to date.
- This value, called the signature block, is appended to the document and sent to the receiver, as shown in Fig. The application of D to the hash is sometimes referred to as decrypting the hash, but it is not really a decryption because the hash has not been encrypted. It is just a mathematical transformation on the hash.

Digital Signatures:

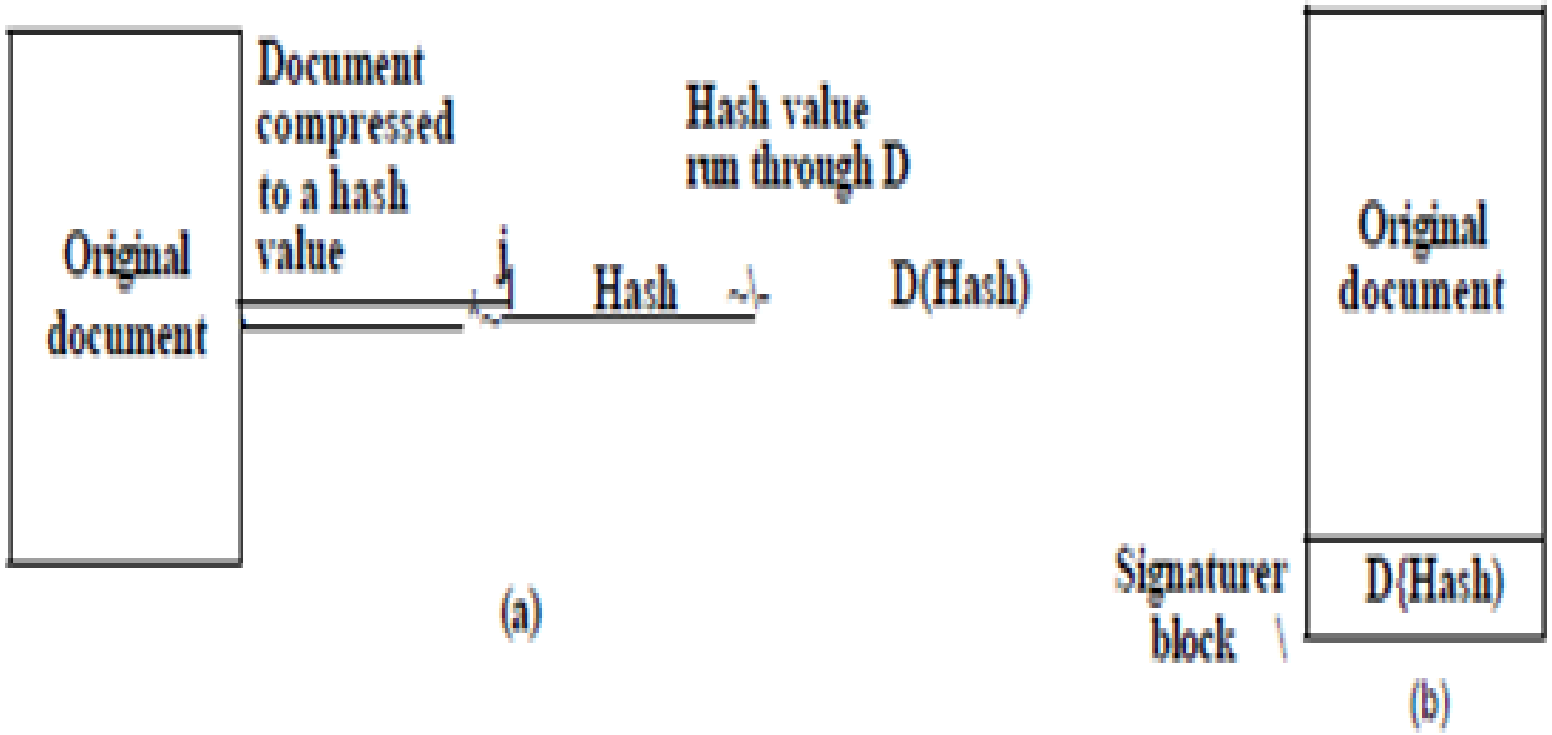


Figure 9-3. (a) Computing a signature block, (b) What the receiver gets.