Introduction to Operating Systems

MCT540

Course Objectives & Course Outcomes

Course Objectives

1. To study various elements of operating systems and compare core functionalities of Windows and Linux operating systems.
2. To study concurrent processes problems, understand various memory management techniques and to analyze deadlock handling methodologies.
3. To learn different protection and security concerns of operating system.

Course Outcomes

On successful completion of the course, students will be able to:

1. Identify various elements of operating system and compare core functionalities of Windows and Linux.
2. Identify and synchronize concurrent processes problems, analyze various memory management techniques and deadlock handling methodologies.
3. Understand different protection and security concerns of operating systems.

Syllabus

Section -I (Weightage – 15%, Minimum Teaching Hours -6)

Introduction - Types of OS, Operating system services, system calls.

File system introduction, Access methods, Allocation methods, Directory system, Disk and drum scheduling. Case study on Unix and Windows Operating System.

Section-II (Weightage – 70%, Minimum Teaching Hours -28)

Process - Introduction, Threads, CPU Scheduling algorithms, Inter-process communication, Critical section problem, Semaphores, Classical process coordination problem.

Deadlock -Definition, Necessary and sufficient conditions for Deadlock, Deadlock Prevention, Deadlock Avoidance: Banker's algorithm, Deadlock detection and Recovery.

Memory Management – Concept of Fragmentation, Swapping, Paging, Segmentation.

Virtual memory-Demand Paging, Page replacement algorithm, Thrashing.

Section-III (Weightage – 15%, Minimum Teaching Hours -6)

Protection:-Goal, Domain of protection, Access matrix, Access control.

Security:-The security problem, Program threats, System and network threats, User authentication.

Text Books and Reference Books

Text Books:

1.Operating System Concepts: Siliberschatz Galvin: John Wiley & Sons.

2.Modern Operating Systems: Andrew Tanenbaum, PHI.

3.Operating System, internals and Design Principles: Williams Stallings.

Reference Books :

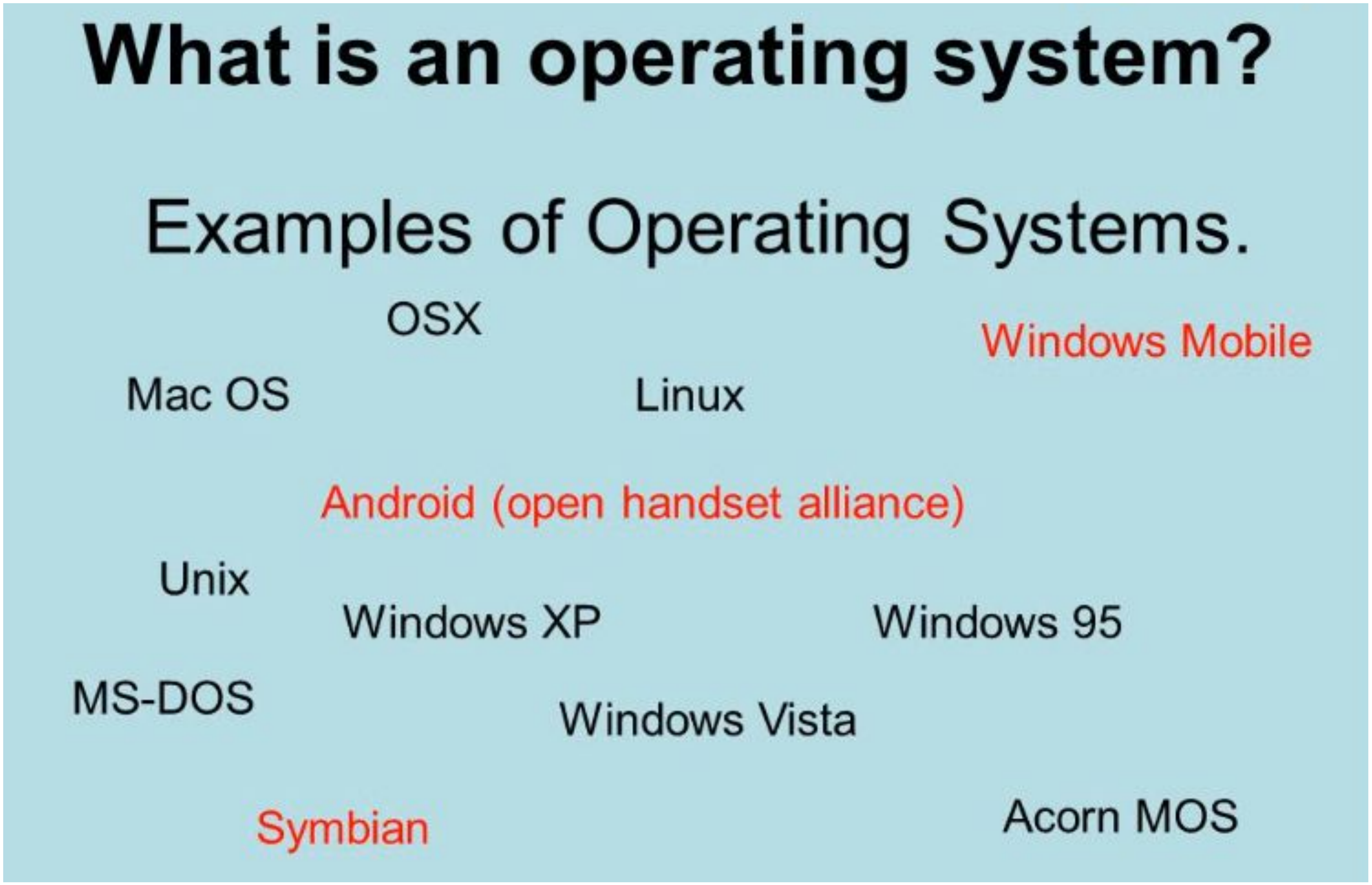
1.An Introduction to Operating System: H.M.Dietel, Pearson Education.

2.Operating System: Charles Crowley, IRWIN Publications.

3.Operating systems: Archer J. Harris, Schaum's Outline, McGraw Hill Publication

Introduction

- OS makes bare hardware usable
- It is essential part of computer system.
- It is essential part of computer science industry.
- Used in computer games
- (packages:Adobe Animate, Unity, Android Studio, pygame, Adventure Game Studio, GameMaker Studio, Godot, Unreal Engine, Pixel Game Maker MV, or Construct)
- What is an Operating System?
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner



What is an operating system?

Examples of Operating Systems.

OSX

Windows Mobile

Mac OS

Linux

Android (open handset alliance)

Unix

Windows XP

Windows 95

MS-DOS

Windows Vista

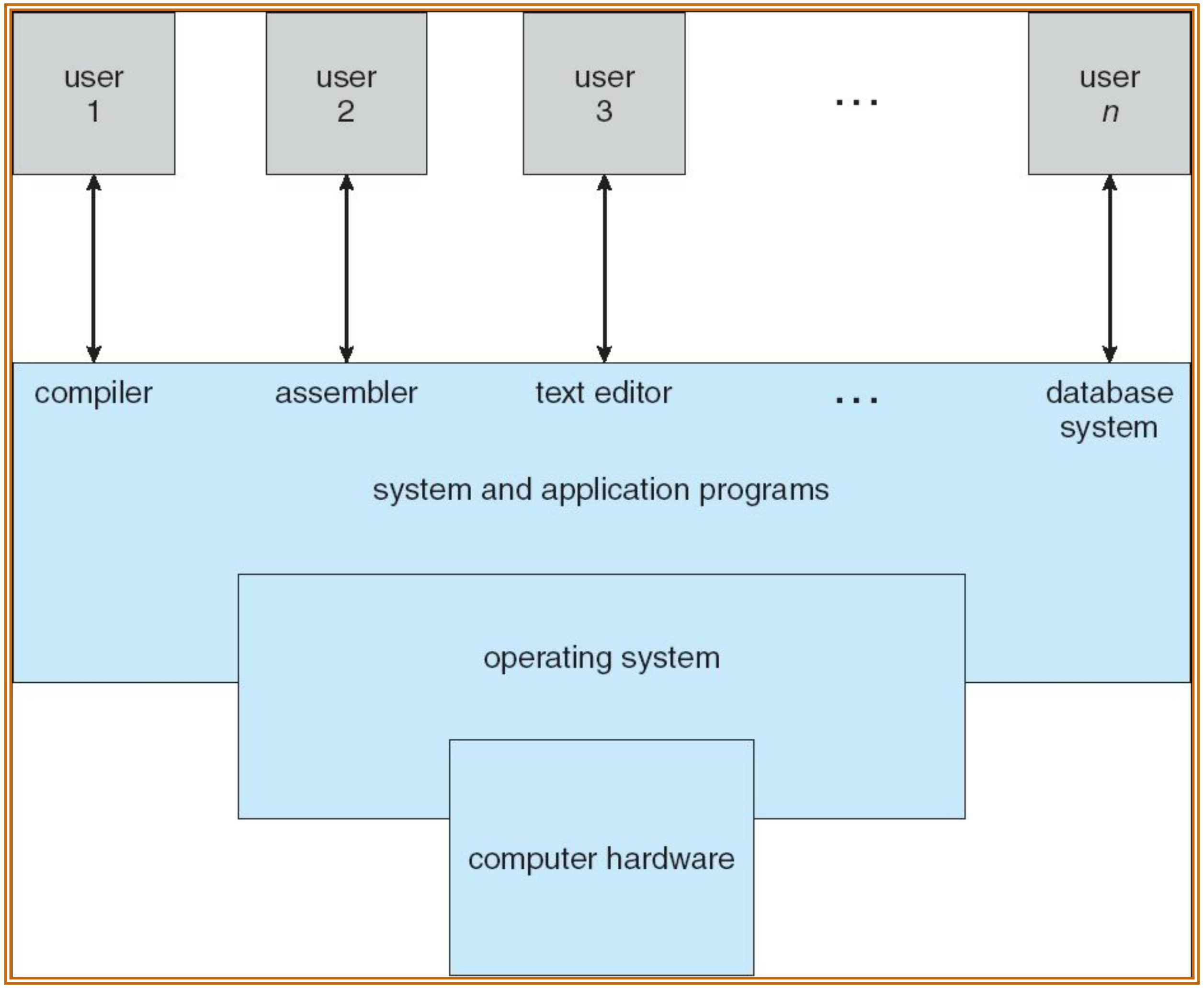
Symbian

Acorn MOS

Components of Computer System

- Computer system can be divided into four components
  - Hardware
    - provides basic computing resources
      - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
    - Act as a government : provide environment to execute programs for solving user problems.
  - System & Application programs
    - Define the ways in which the system resources are used to solve the computing problems of the users
      - System programs (utility programs), compiler, assembler
      - Application programs like word processor, spreadsheet, web browser
  - Users
    - People, machines, other computers

Four Components of a Computer System

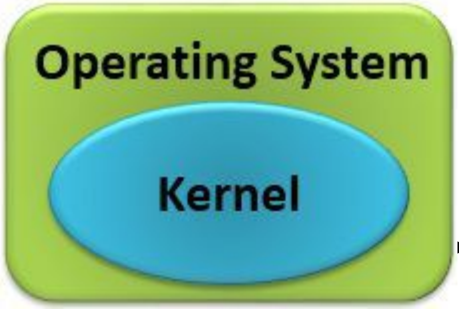


Operating system makes computer system usable

OS as Government

- Govt. control and manage the resources.
- Government provides an environment to perform day to day work and fulfil basic needs of people.
- Operating systems provide an environment for execution of programs and services to programs and users

Kernel



- “The one program running at all times on the computer” is the kernel.
- Everything else is either a system program (ships with the operating system) , or an application program.
- **Operating system** is system program that runs on the computer.
- **Kernel** is also a system program that controls all programs running on the computer.
- Kernel is basically a bridge between software and hardware of the system.
- Kernel is the core of the operating system.
- It is the **first** program of operating system that is **loaded into the main memory** to start the working of the system.
- Kernel remains in the main memory till the system is shut down.
- Kernel basically translates the commands entered by the user in a way to make the computer understand that what has user requested.

Kernel

- Kernel directly communicates with the hardware and let it know what the application software has requested. An operating system is unable to run without the kernel as it is the important program for the working of the system.
- Kernel takes care of the **memory management**, **process management**, **task management** and **disk management**.
- Kernel checks out the memory space for the proper execution of the application program.
- It creates and destructs memory which helps in execution of the software.
- Kernel is classified as **Monolithic kernel** and **Microkernel**.
- In a Monolithic kernel, all the services of the operating system run along the main thread of the kernel that resides in the same area of memory where the kernel is placed.
- Monolithic kernel provides rich access to the hardware of the system. Microkernel is an abstraction over the hardware that uses the primitives or system calls to implement the services of operating system.

OS definition

- Operating System is a software, which makes hardware to actually work.
- It is an interface between h/w & s/w
  - Executes programs on the behalf of user
- It acts as a
  - Control program
  - resource manager.
    - It controls I/O devices and co-ordinate resources among users.
- We can explain role of OS in detail from two viewpoints.
  - User view
    - Depends on how user interfaces with computer system
      - Single user / Multi-User / Both / without user intervention
  - System view

User View: in single user environment

IBM Personal Computer XT in 1988

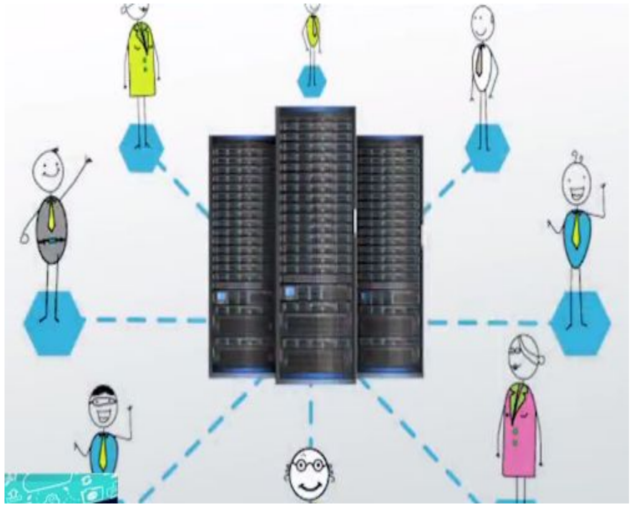
OS designed for

Individual usability
maximize performance
Protection & Security
Ease of use
Convenience

No Efficiency

Don't care about **resource utilization**

User View in multiuser environment Mainframe



Client-Server Architecture :Mainframe Systems

- Huge setup accessed by thousands of users.
- High processing power. (in MIPS)
- Handle bulky data.
- Used in NASA.

- Operating systems are designed for maximum resource utilization
- Used in multiuser environment.

Handheld Systems

- Handheld systems are designed for:
  - individual usability,
  - convenience,
  - ease of use
  - and maximize utilization.
  - Problems
    - low processing power. Need of recharging
    - Small i/o devices. Problem in accessing email, web browsing.
    - Low memory space. Need of flushing memory space, virtual memory.

Embedded Systems

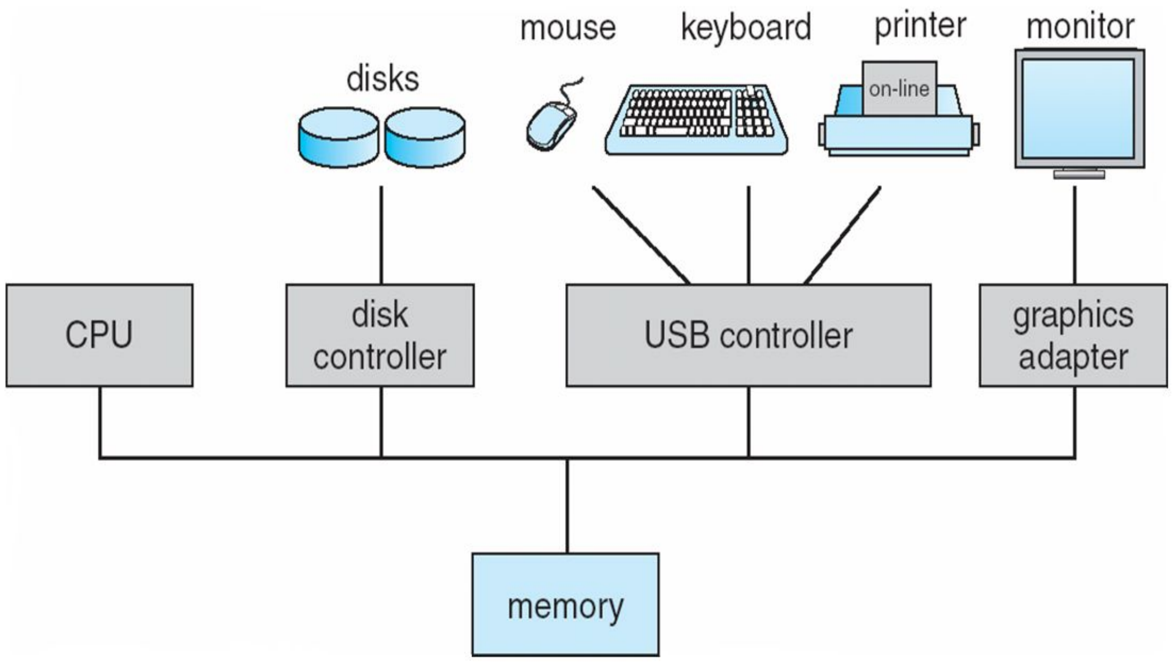
- Embedded Systems require :
  - Little or no user intervention

Computer System View

- OS is a program that act as a
  - OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Control devices,
  - user programs to prevent errors and improper use of the computer

  - Fundamental goal of OS is to execute user programs easily , conveniently & efficiently

Computer System Organization

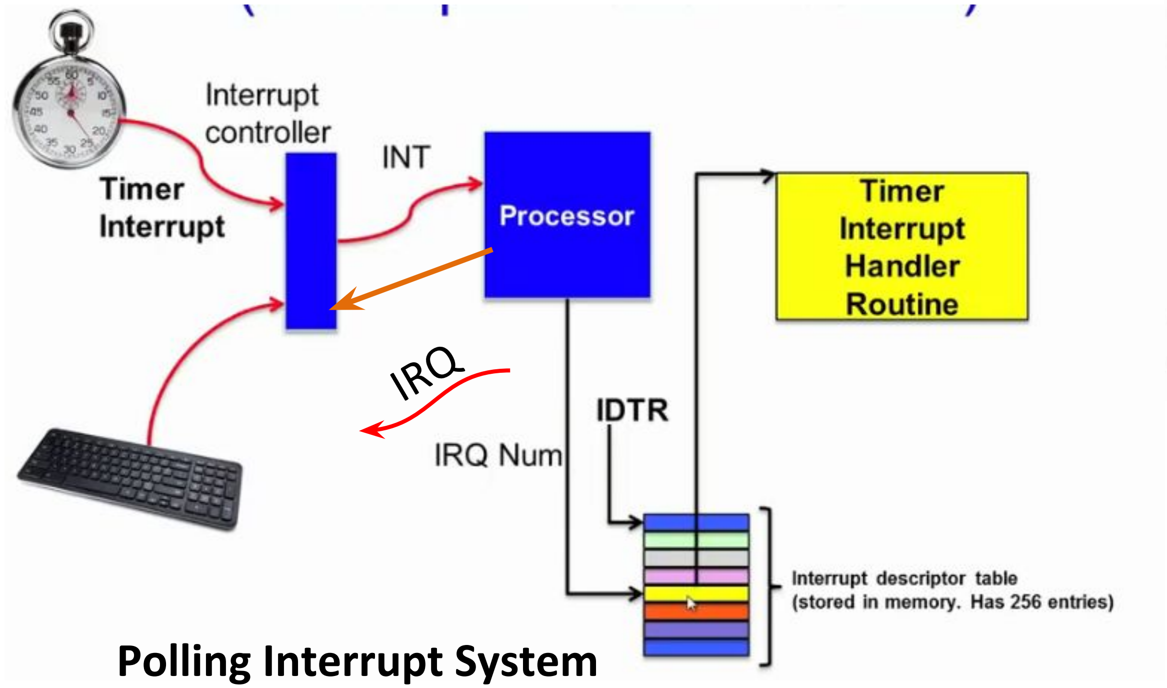


Modern General Purpose Computer System

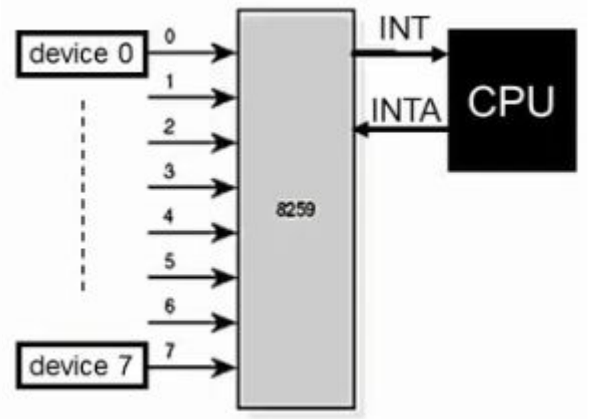
Computer Startup / Reboot

- **bootstrap program / firmware** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM
  - Initializes all contents of a system (CPU registers, memory, device controllers)
  - Loads operating system kernel
  - OS then executes first process (init) and waits for an event to occur
  - Occurrence of event is signaled by using an interrupt (hardware interrupt or software interrupt).
  - Hardware interrupt is handled by sending signal to CPU while software interrupt is handled sending trap to the OS caused either by executing system call / exception / illegal access of privileged instruction by user.

Windows,unix


Interrupt controller
INT
Processor
Timer Interrupt
Timer Interrupt Handler Routine
IRQ
IRQ Num
IDTR
Interrupt descriptor table (stored in memory. Has 256 entries)

Polling Interrupt System



8259 controller

***vectored* interrupt system**
Containing addresses
of all the service routines

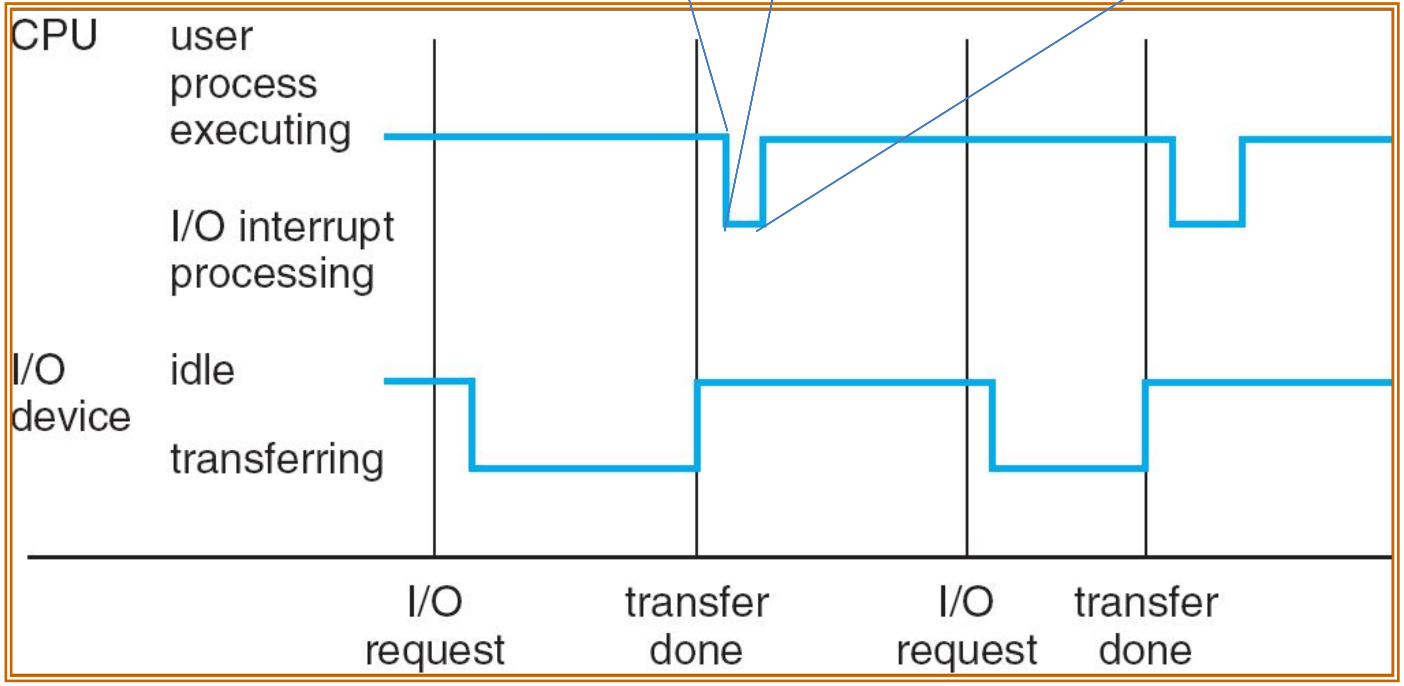
- The operating system preserves the state of the CPU by storing registers and the program counter.
- OS Determines which type of interrupt has occurred by *polling / vectored* interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt
- After interrupt is serviced, the saved return address is loaded into program counter.

Interrupt Timeline

I/O request,
preserves PCB of a process

Control passes
to a specific location

Control passes
To return address



Storage Structure

- Memory is basic computing resource used in computer system.
- Before execution program, data should be loaded in main memory.
- Required Instruction and data is sent to CPU registers.
- Results are stored back into main memory.

- Read Only Memory (ROM) : store static programs e.g. bootstrap program)
- EEPROM: Cannot be changed frequently e.g. modem.
- Main memory (RAM) – volatile, usually too small, CPU can access directly.
- Secondary storage – nonvolatile, extension of main memory , large capacity.
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material

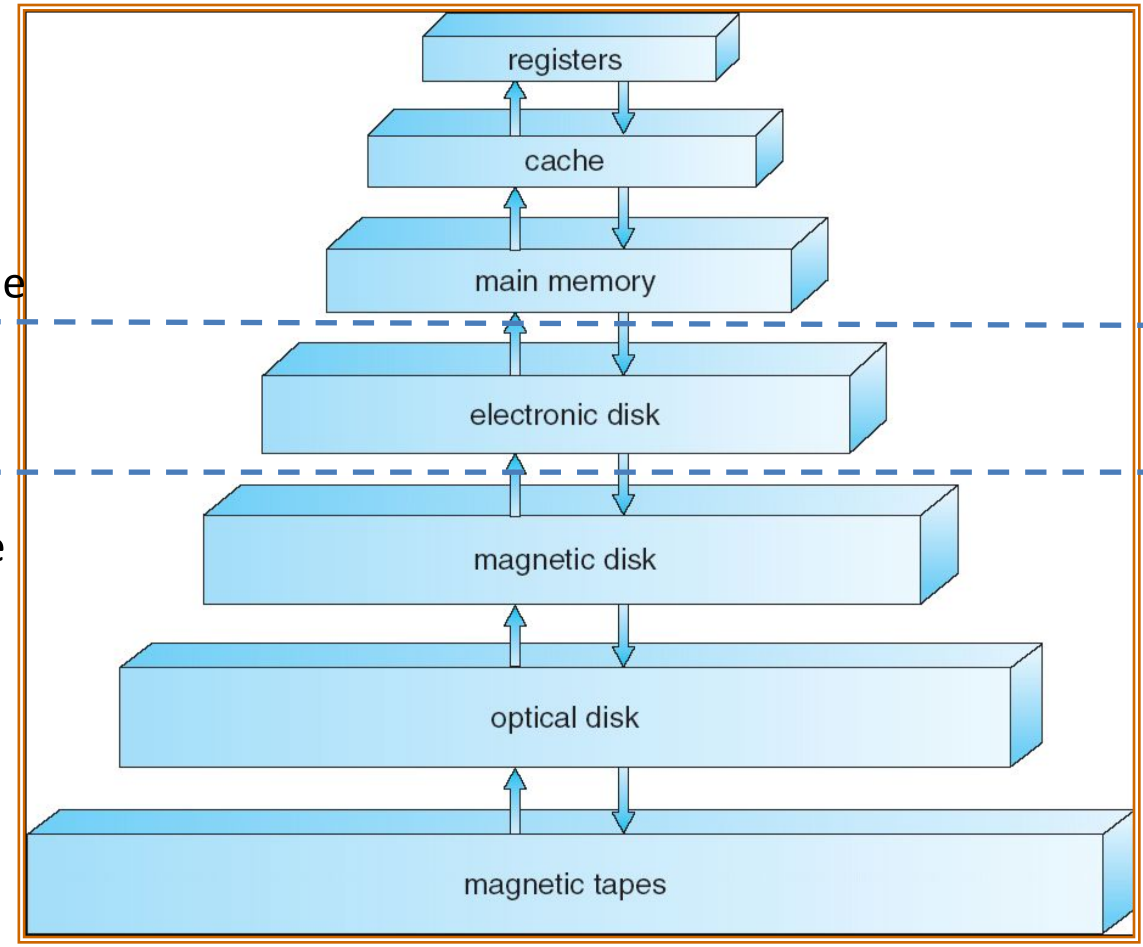
Storage-Device Hierarchy

Expensive,fast

Volatile

Non-Volatile

Cheap , slow



Difference in speed, cost, size and volatility

Storage Hierarchy

- Storage systems are organized in hierarchy.
- The main difference lies between
  - Speed
  - Cost
  - Size
  - Volatility
- *Caching* – copying information into faster storage system.

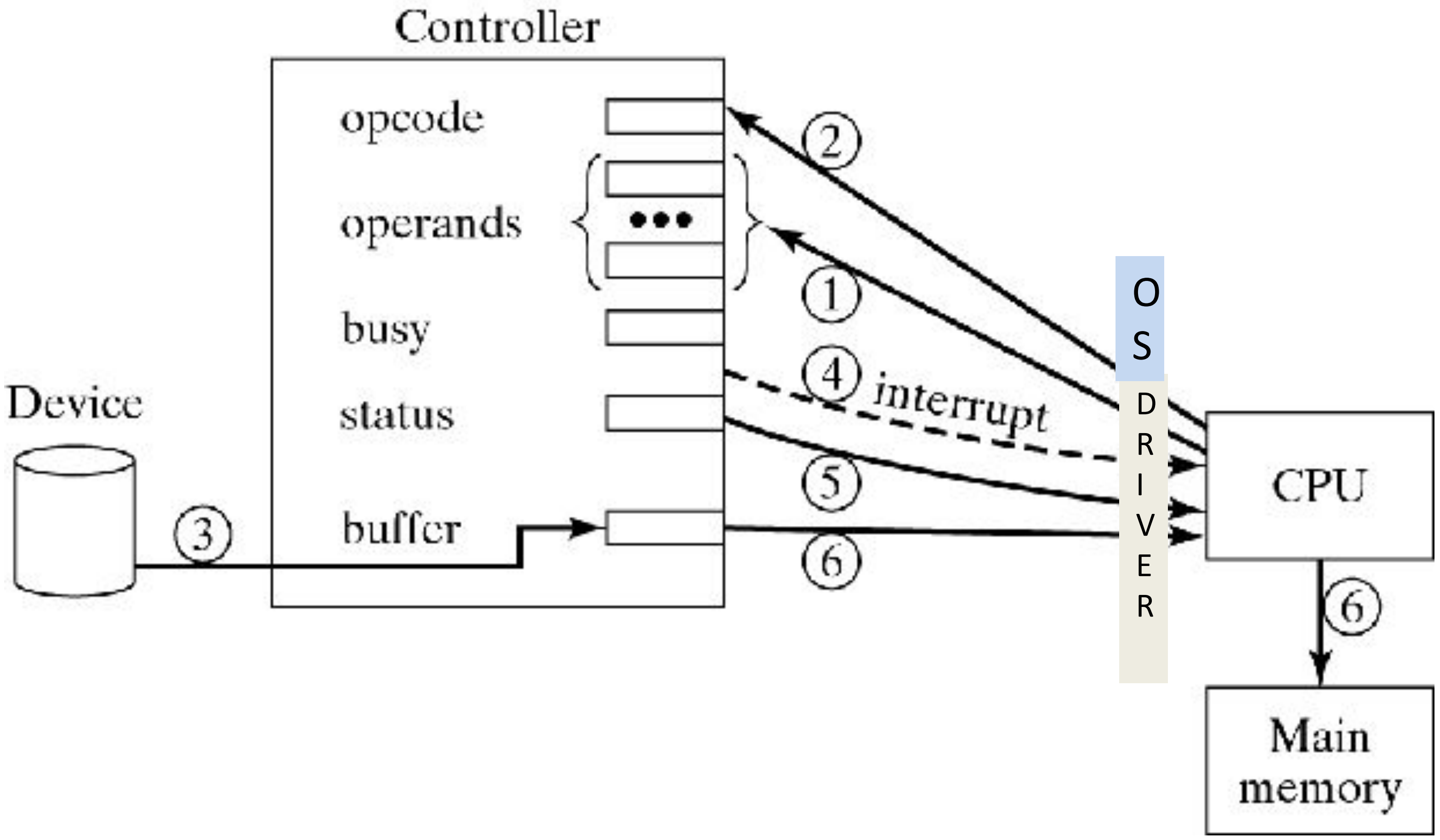
I/O structure

- Most of the times computer system remains busy in performing i/o operation.

- Modes of data transfer:
  - Programmed i/o
  - DMA (Direct Memory Access)

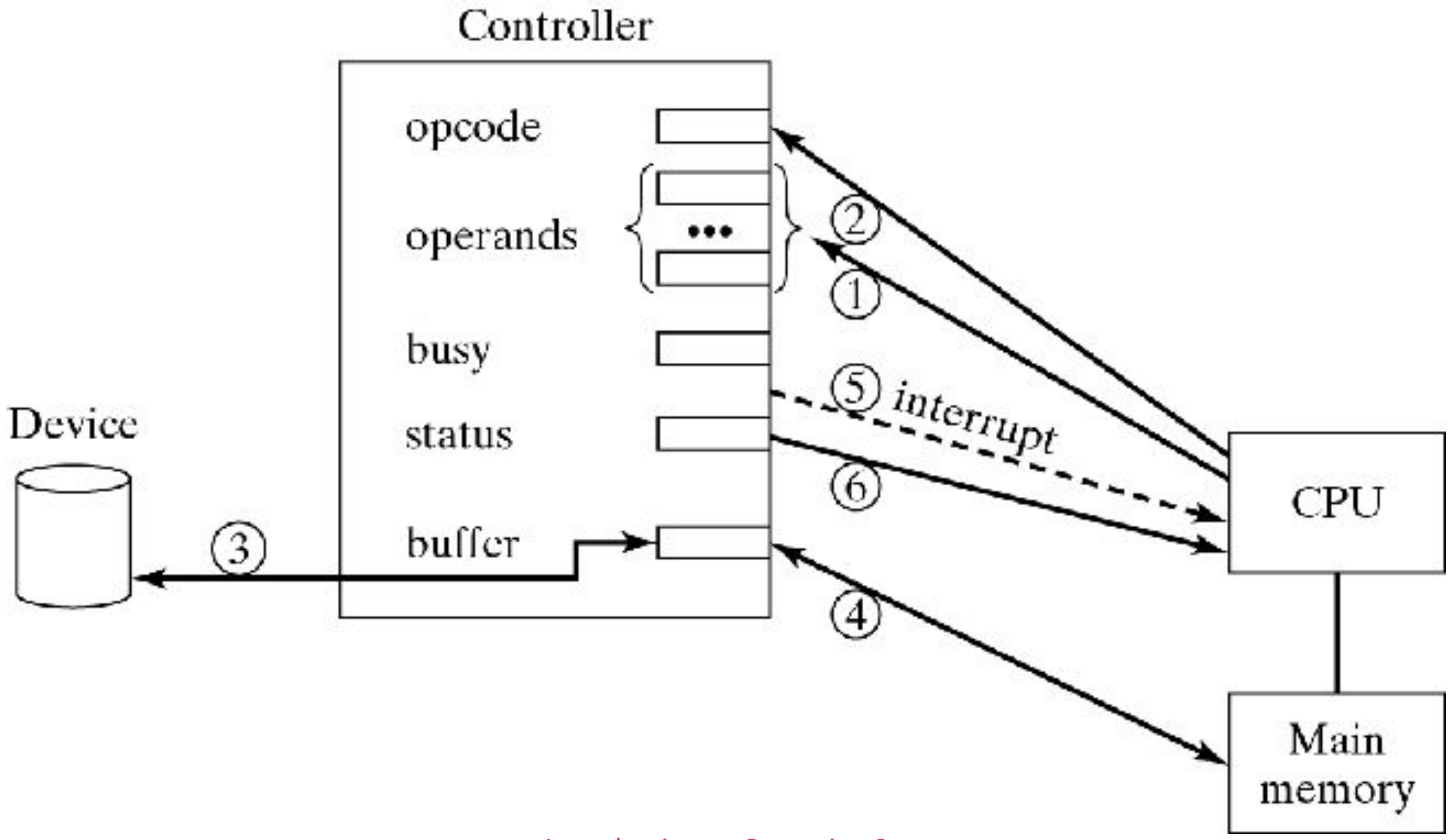
Programmed I/O with **Interrupts**

- Intervention of CPU while moving **data to memory**,
- Interrupt signal informs device driver when I/O operation **completes.** Device driver in turn return control to OS by returning data.
- **<u>Protocol : interrupt per byte</u>**



Direct Memory Access (DMA)

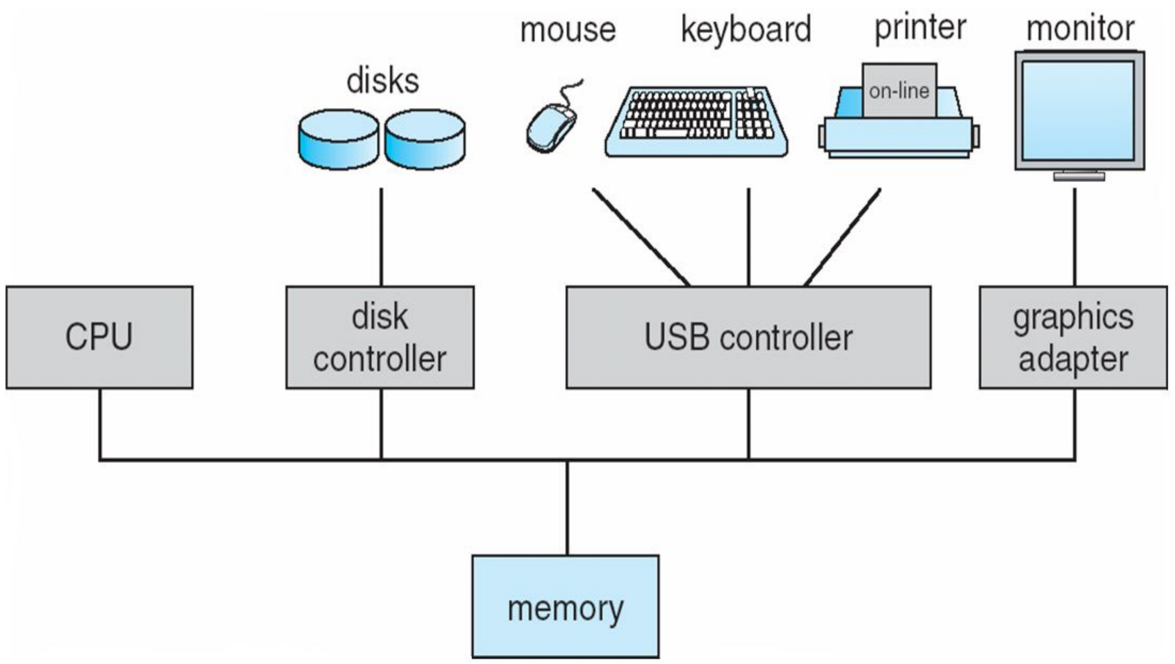
- CPU does not transfer data, only **initiates** operation
- **DMA controller transfers data** directly to/from main memory
- **Interrupts** when transfer completed
- Input protocol: one interrupt per block



Computer-System Architecture

- Single Processor Systems
- Multiprocessors systems
- Clustered Systems

Single Processor System



Computer-System Architecture
Single Processor Systems

- There is only **one main(general purpose) CPU capable of executing instruction** set of user processor.

- Almost all systems have other special-purpose processors as well. Thus relieve CPU from these tasks.
  - Keyboard processor to **convert the keystrokes** to be sent to cpu.
  - Disk microprocessor performs **disk scheduling algorithms**

- But OS cannot communicate with these processors. They work autonomously.

Computer-System Architecture Multi-Processors Systems

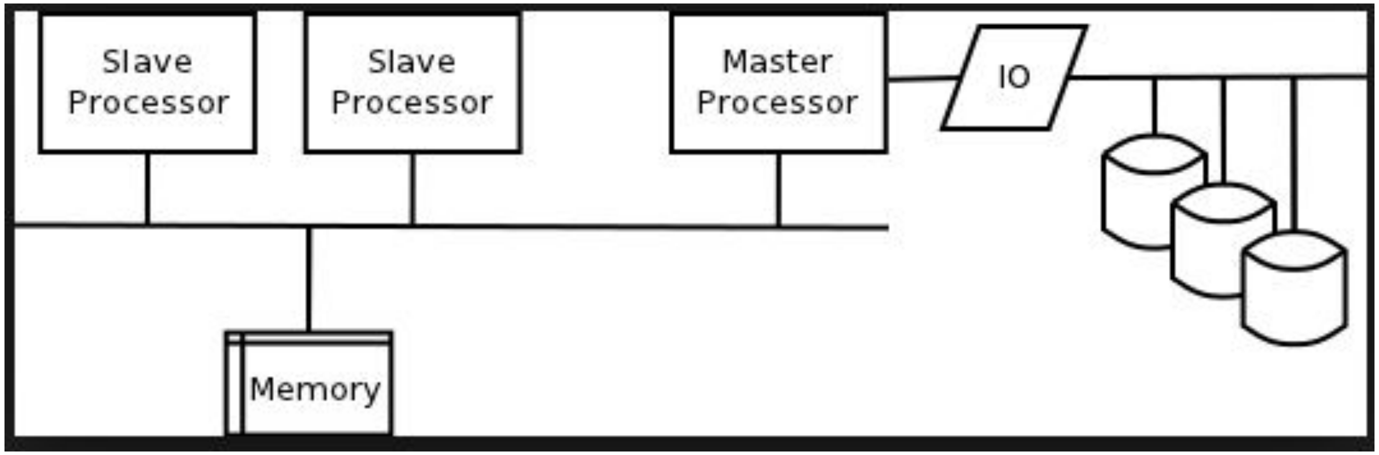
- Also known as **parallel systems**, **tightly-coupled systems(share storage)**

- **They share bus, clock, memory and peripheral devices.**

- Advantages include:
  1. **Increased throughput:** **speed up ratio with N processors is not N**
     1. **Overhead of synchronization, contention while accessing device.**
  2. **Economy of scale :** **share devices**
  3. **Increased reliability** –
     - Graceful degradation : continue to provide minimum services
     - Fault tolerance : can suffer a failure of single component still continue operation.

Multi-Processors Systems

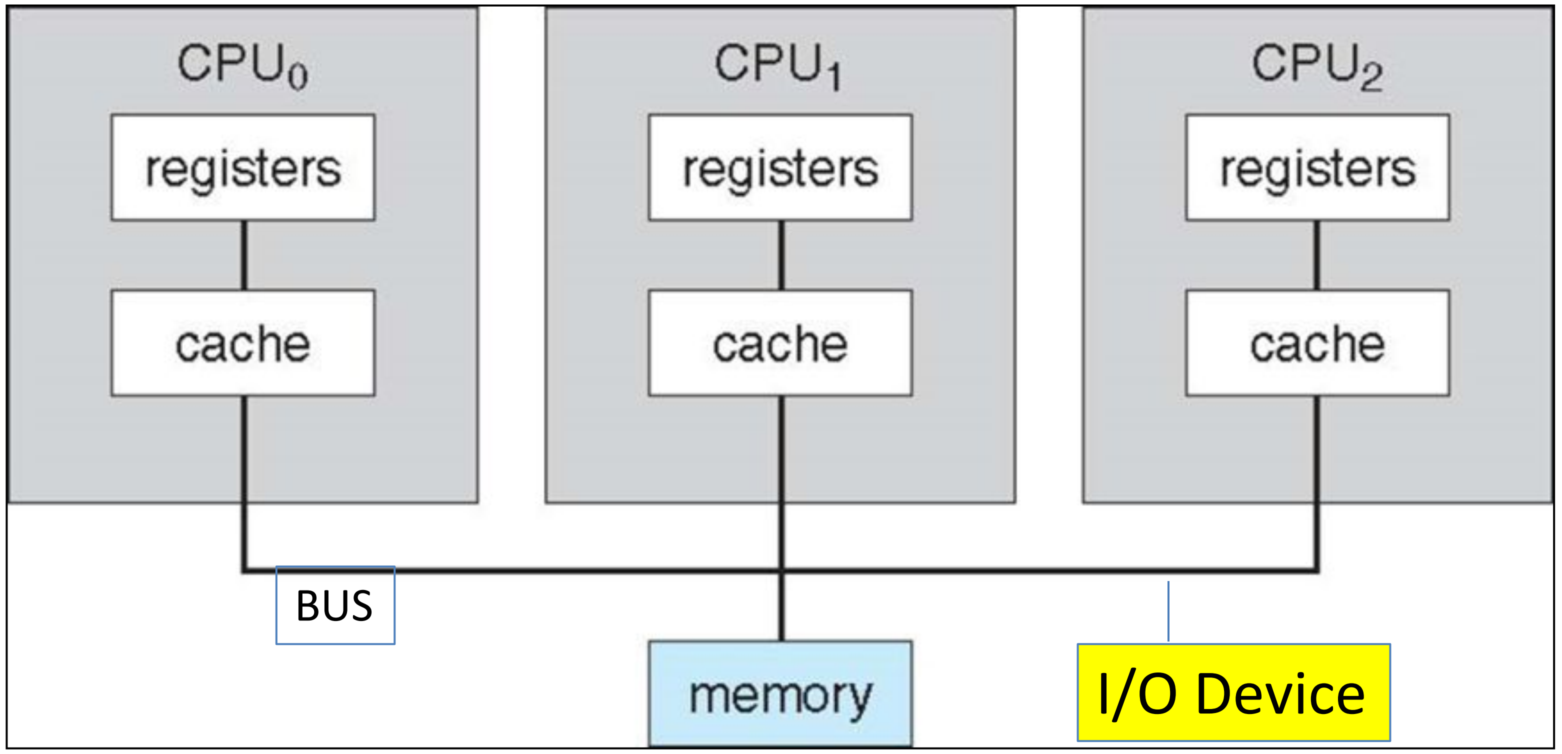
Asymmetric Multiprocessing (ASMP)–



Computer-System Architecture Multi-Processors Systems Types

1. **Asymmetric Multiprocessing (ASMP)**–
   - each processor is assigned a specific task.
   - Master – Slave relationship between the processors.
     - Master processor schedules and allocate work to slave processors.

Symmetric Multi-Processors Systems



Tightly Coupled Systems

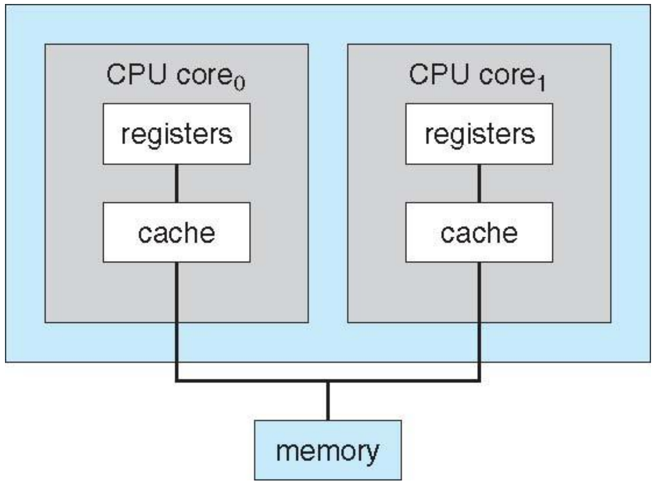
Multi-Processors Systems

Symmetric Multiprocessing (SMP)

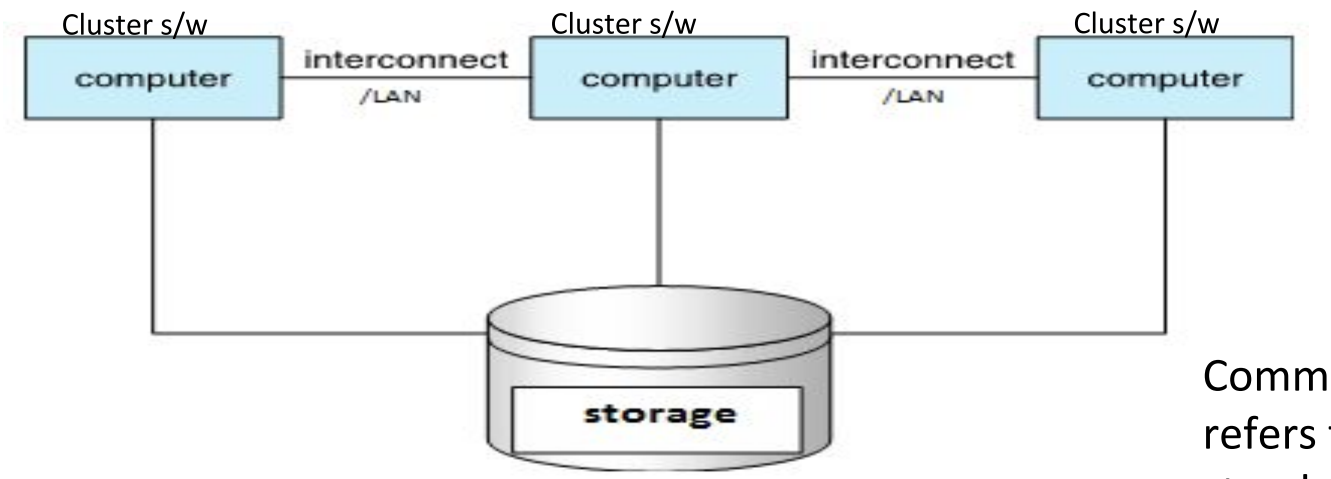
1. **No master-slave relationship**
2. Each processor performs all tasks within OS.
3. Each processor has local registers and cache.
4. N processes can be executed on N CPU's.
5. I/O control is required.
6. Inefficiency of any processor can be avoided by sharing data structures(memory).

Multi-Processors Systems
A Dual-Core Design

- Single chip with **multicore.**
- **Faster than single chip with single core.**
- Each core has its own register set and cache
- Multi-core CPU appear as a separate processors to the OS



Clustered Systems



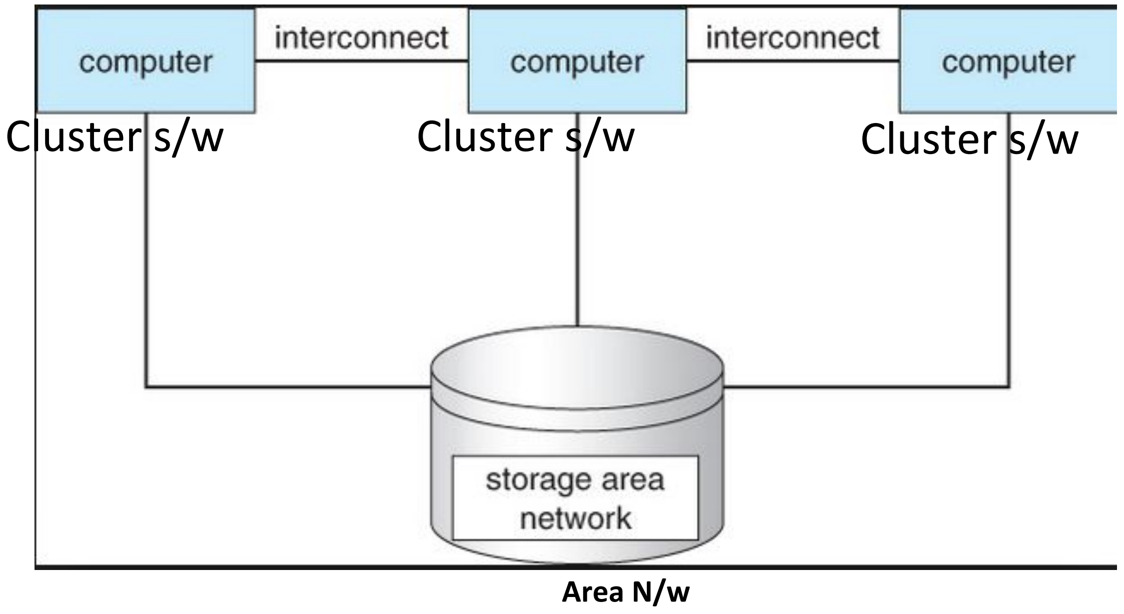
Commodity hardware refers to cheap, standardized servers that are easy to buy off the shelf from any vendor.

- They are collection of **multiple homogeneous individual systems.**
- Clusters are built using commodity hardware.
- Requirement :
  - cluster s/w on each m/s

- Features:
  - High availability :
    - Redundancy, Monitoring
  - Provide high-performance computing environment than multiprocessor system.
    - Parallelization : Dividing modules
- Parallel Cluster
  - Multiple hosts can access same data on shared storage
    - E.g. oracle real application cluster.

Clustered Systems

- Cluster supports dozens of systems separated by miles and share a pool of storage called **storage area network.**

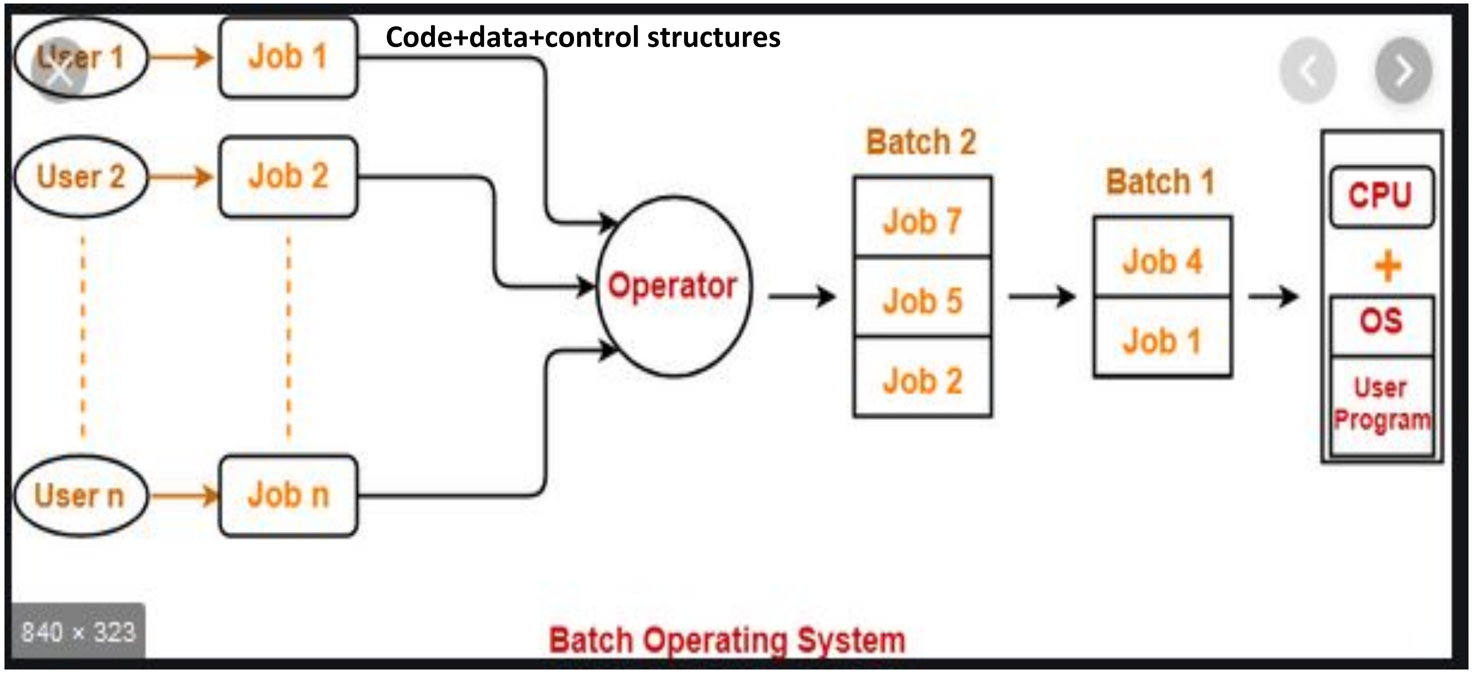


Pool of Storage

OS Structures

- Some common features are:
  - Multiprogramming
  - Time Sharing

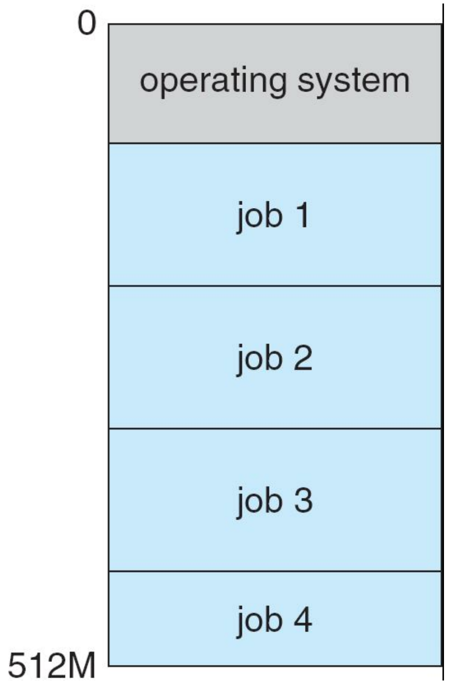
Batch Operating System



- Users give jobs to a console operator.
- Console operator group similar jobs to form a batch.
- Computer system execute one batch at a time.
- Drawback:
  - Wastage of resources

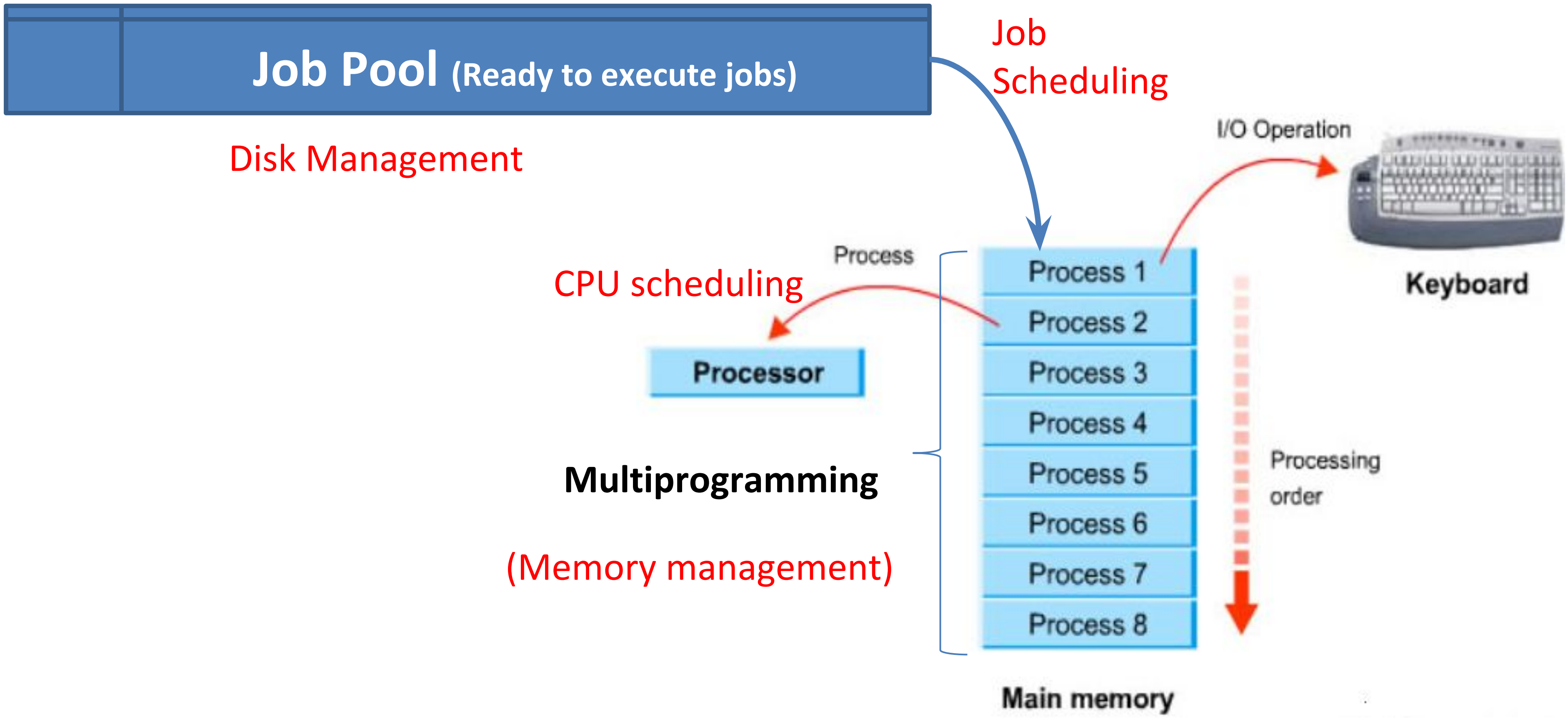
Why Multiprogramming?

- Single program cannot keep CPU and I/O devices busy at all times
- User can run multiple jobs simultaneously
- Multiprogramming organizes jobs (code and data) in memory so that CPU always has one to execute. (increases CPU utilization)



Memory Layout for multiprogramming system

Multiprogramming



- CPU is never idle.
- Effective resource utilization.

Features in Multiprogramming

Disk Management: job pool keeps ready to execute jobs on device.

job scheduling: Jobs selected from job pool and loaded in memory.

Memory management: manages jobs in memory via swapping and virtual memory.

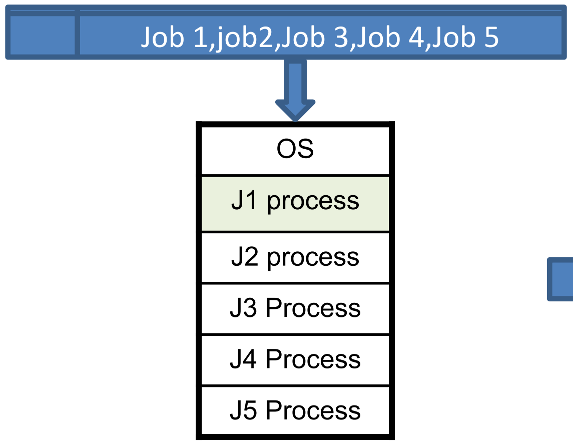
CPU scheduling: CPU Scheduling is **a process of determining which process will own CPU for execution while another process is on hold**

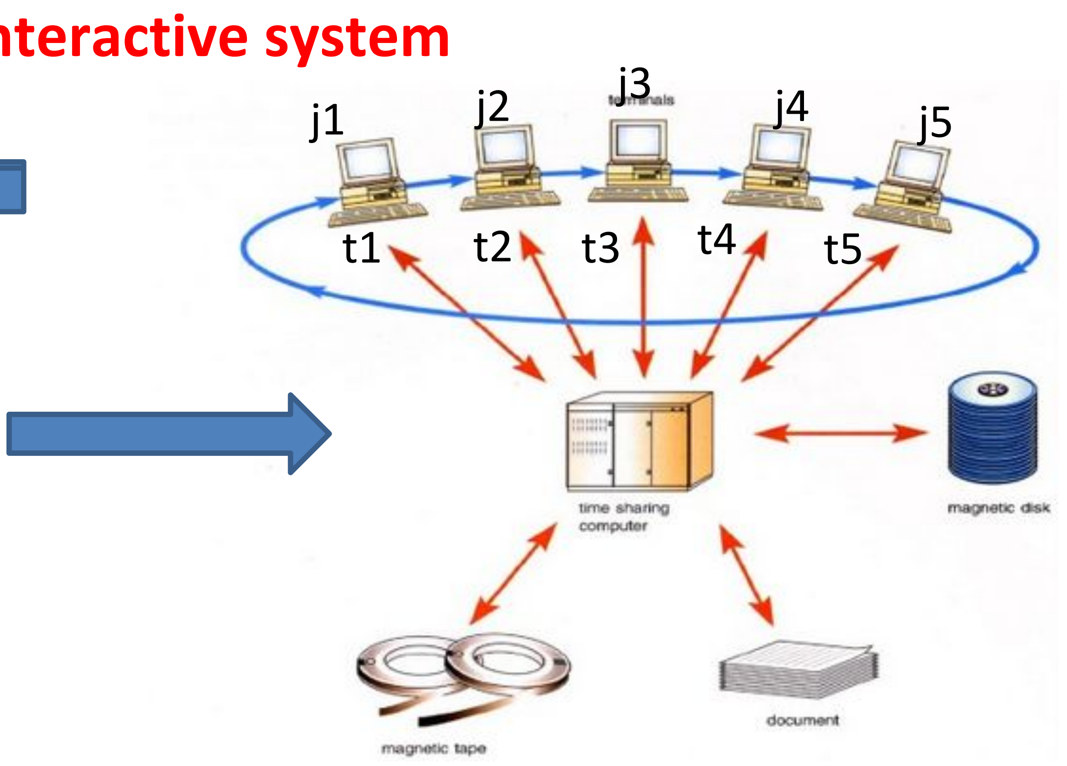
Disk scheduling: Disk scheduling is done by operating systems **to schedule I/O requests arriving for the disk**. Disk scheduling is also known as I/O scheduling. Disk scheduling is important because: Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller.

Multiprogramming:

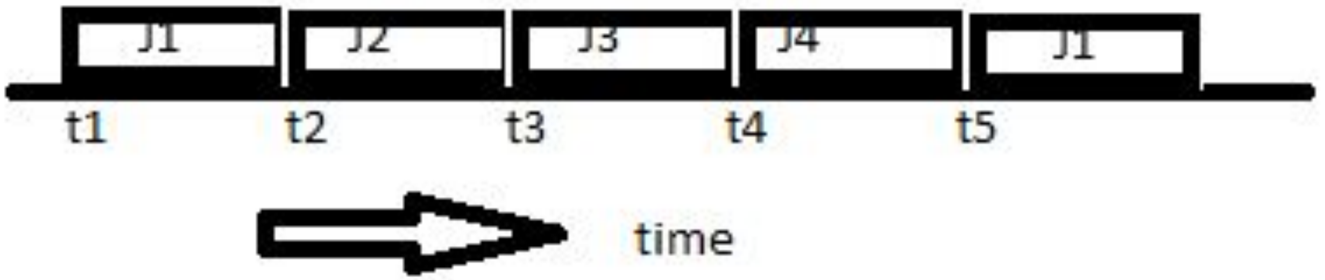
- **Advantages:**
  - Efficient utilization of CPU, memory, storage devices, and peripheral devices
  - Throughput increases (Throughput means **total number of programs executed over a fixed period of time**. In multiprogramming, CPU does not wait for I/O for the program it is executing, thus resulting in an increased throughput)
- **Disadvantages:**
  - Sometimes long time jobs have to wait long time.
  - Tracking of all processes sometimes difficult.
  - Requires CPU scheduling.
  - Requires efficient memory management.
  - No user interaction with any program during execution.

Time Sharing / Multitasking Systems:





Reasonable response time
Swapping/ virtual memory



Time Sharing **(Multitasking)**

- It is logical extension of multiprogramming .
- Multiple jobs are executed by switching the CPU between them.

- **User interaction:**
  - **Each user is given a time slice to interact with cpu.**
  - **Feeling of dedicated system**
  - Each user has at least one program executing in memory .
  - If several jobs ready to run at the same time CPU scheduling
  - Memory management by swapping.
  - Virtual memory allows execution of processes not completely in memory.

- **E.g. Unix**

Time Sharing / Multitasking Systems features

- Disk management
- Job scheduling
- Multiprogramming
- Memory Management
  - Virtual memory, Swapping
- CPU scheduling
- Disk scheduling
- **Job synchronization**
  - **Orderly execution of job**
- **Deadlock free**
  - **No waiting for other job**
- **Communication & security**

Timesharing operating systems Advantages/Disadvantages

- Advantages
  - User interaction.
  - Quick response.
  - Avoids duplication of software.
  - Reduces CPU idle time.
- Disadvantages
  - Question of **security and integrity** of user programs and data.
  - Problem of data **communication**.

OS Operations
Dual Mode Operation

- **OS is interrupt driven**
- When there is no work OS remains idle and waits for an event. Occurrence of an Event is signalled by h/w or s/w interrupt.
- h/w interrupt is signalled by sending signal to cpu while s/w interrupt (caused due to bug in a program, exception, use of unprivileged instruction, system call) is signalled by sending trap to OS.
- Since OS and user share h/w and s/w error , bug in one program may create problem to other programs or even to OS.
- So for proper execution of OS, to distinguish between OS code and user code most computer systems take hardware support.

Dual Mode Operation

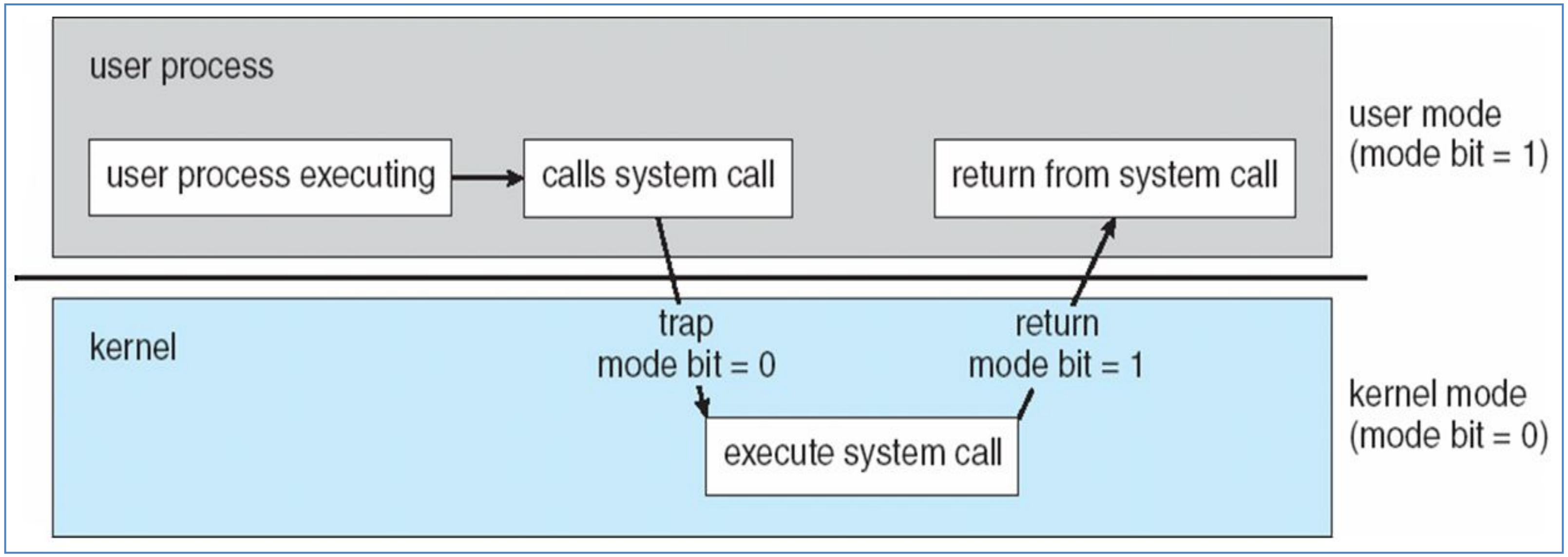
- Some systems uses dual mode operation (uses mode bit in h/w).
  - Such systems uses two modes 1. User mode 2. Kernel Mode (supervisor mode, privileged mode).
  - When system execute user process mode bit in ah/w set to 1.
  - When there is bug, exception, unprivileged instruction or use of system call in a running program , interrupt (trap) is sent to OS. Mode bit set to 0. after execution of service routine or handling error in a program, again mode bit set to 1 . Then control passes to user processes.
- MS DOS do not support dual mode. Bug in a program can wipe out OS program

Operating-System Operations (cont.) Dual Mode Operation

- **Some computer system uses** **Dual-mode** operation which allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Transition from User to Kernel Mode

- During booting, OSis in kernel mode and while executing application, it is in user mode.
- Interrupt by system call changes user mode (bit 1) to kernel mode (bit 0)
- Some instructions like I/o control,memory access are privileged instructions. Accessing privileged instructions in user mode is illegal and send trap to OS.



Caching

- When information is **first accessed** it is kept in main memory.

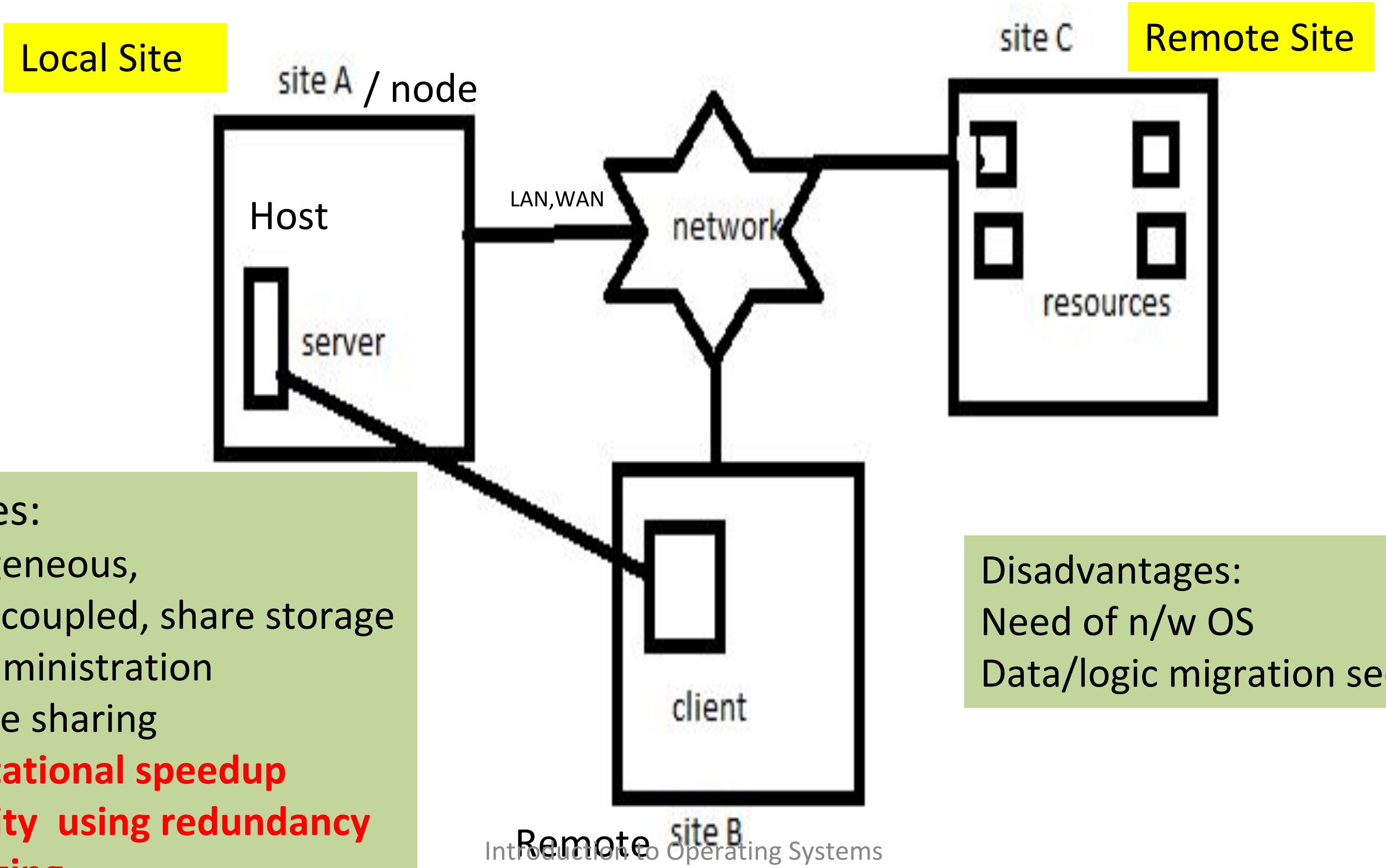
- After it is used, on temporary basis it is copied into faster storage called **cache memory**.

- When particular information is needed, it is **first searched in cache memory.** If it is available, it is directly accessed from cache. Else it is accessed from source.
- **<u>Cache coherency</u>**
  - In a multiprocessor system each process has local cache.
  - If all of them access same data item, say 'A', update to 'A' in one cache should be reflected in all other cache.
  - As cache have limited size, cache management is critical problem.
  - Proper cache size and replacement policy is needed.

Distributed Systems

- collection of physically separate, loosely coupled, possibly heterogeneous processors interconnected by communication network for sharing files and data.
- Functionality of distributed systems depends on networking (media,lan,wan,wireless communication).
- Each processor is called as site / node.
- From the point of view of a specific processor in distributed system, the rest of the processors and their respective resources are remote, while its own resources are local.
- Location indicates specific site while specific m/c on the site is called host.

General Distributed Systems Structure



Features:
Heterogeneous,
Loosely coupled, share storage
Local administration
Resource sharing
Computational speedup
Reliability using redundancy
Downsizing

Disadvantages:
Need of n/w OS
Data/logic migration security

Special purpose systems

- Real time Embedded systems
- Multimedia systems
- Handheld systems

Real Time Embedded Systems

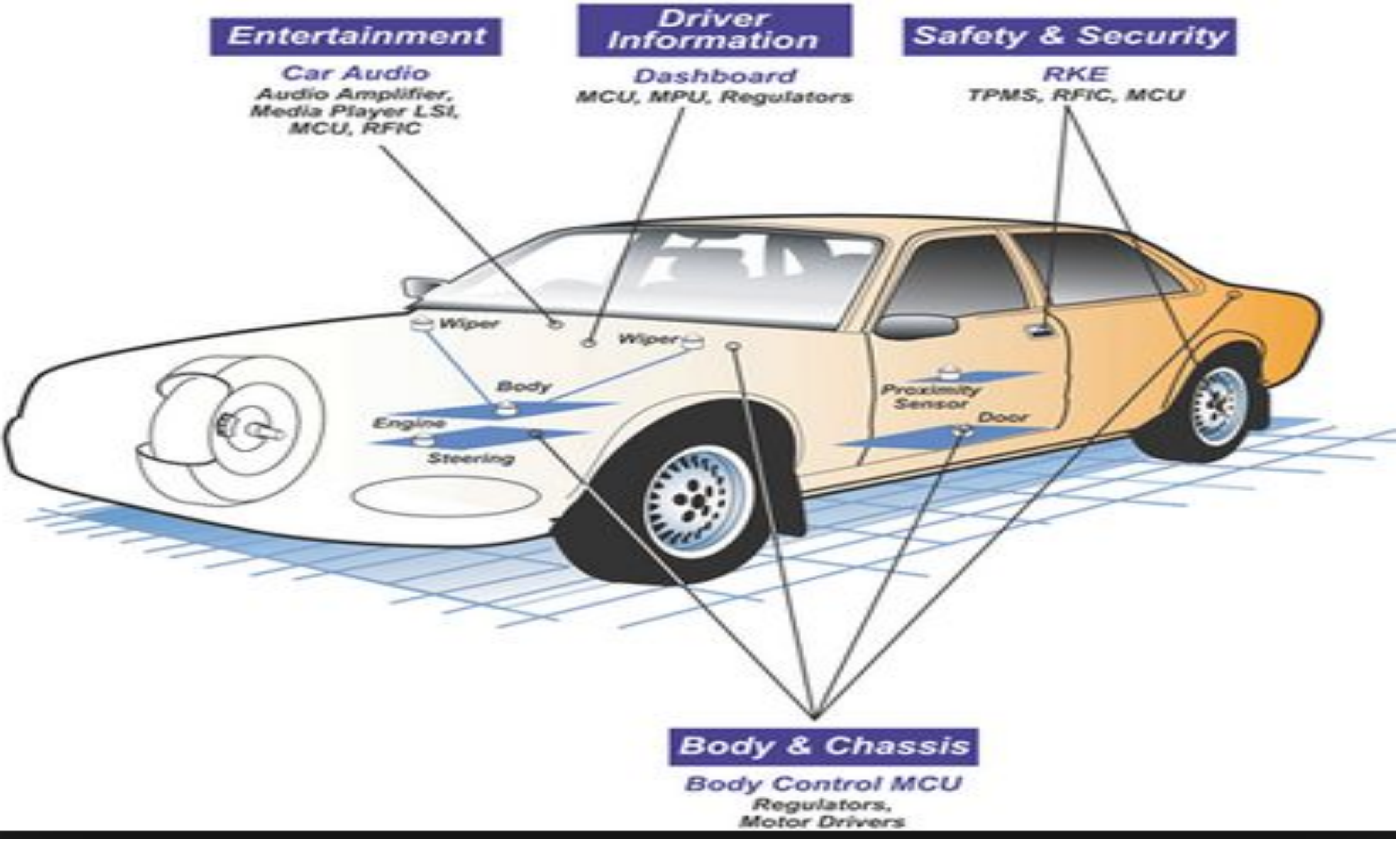
- Embedded systems are intended for specific task with little or no user interaction.
- Use **system-on-chip (SOC) strategy** instead of bus.
- Limited functionality
- Monitoring, controlling

- Variations:
  - Desktop systems running Applications for special-purpose using win/linux os.
  - Hardware devices with application specific integrated circuits.
  - Embedded systems run real time OS
- E.g. home appliances, automobiles, robotic arms, computer systems.

Special purpose systems
Real time embedded systems

- What action to be taken is decided at run time
- RTOS require rigid time requirements. So mainly used for controlling
  - Sensor bring data, computer analyzes data, may use controls to adjust data and devices.

- Hard / soft RTOS.


Entertainment
Car Audio
Audio Amplifier, Media Player LSI, MCU, RFIC
Driver Information
Dashboard
MCU, MPU, Regulators
Safety & Security
RKE
TPMS, RFIC, MCU
Wiper
Wiper
Body
Engine
Steering
Proximity Sensor
Door
Body & Chassis
Body Control MCU
Regulators, Motor Drivers

Multimedia Systems

- Most OS handle conventional data like text documents, word processing files, spreadsheets etc.
- Recent trend is use multimedia data that include **audio / video / images** along with **conventional data**.
- Multimedia data must be delivered as per **time constraints**. E.g. for video 30 frames per second.
- Applications : video conferencing, online lectures, movies.
- Can be seen on small devices.
- Data delivery is through
  - Local playblack
    - watching dvd on local m/c
  - Streaming
    - Data delivery from remote site (online lectures)

Multimedia: Text

Multimedia data

Stone

Papyrus

Parchment

Multimedia systems

- Characteristics:
  - Use very large files. So need of compression.
  - Continuous media require high data transfer rate
  - Must be delivered as per time constraints.
    - E.g. video frames must be delivered at specific rate. (for video 30 frames / s in mpeg format)

Handheld systems

- E.g cellular phone, mobile, tablet

- Features
  - Small size,
  - less storage,
  - low speed,
  - less memory,
  - small I/O

Handheld systems

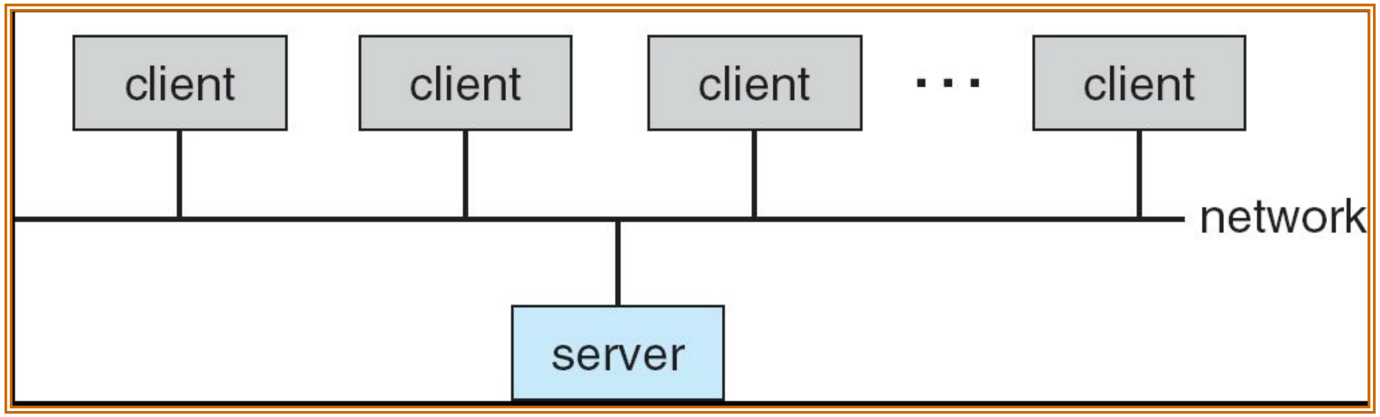
- Microprocessor speed:
  - Most systems are with low speed
  - But systems with high speed require more power, large batteries, continuous charging
- Memory management:
- Small i/o
  - Small keyboard, and screen
  - Problem in browsing
  - We can use web clipping
- Features added
  - Wifi, bluetooth for data transfer
- Advantages:
  - Convenience, portable

Computing Environments

- Traditional computing
  - Use of mainframe,multasking machines
- Client-Server computing
  - Computing server,file server
- Peer-to-peer computing
  - No client-server distinction
- Web computing

Computing Environments

- Client-Server Computing
  - Special form of distributed system.
  - High speed, computational power, less cost machines as server.
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e. database)
    - **File-server** provides interface for clients to store and retrieve files



Peer-to-Peer Computing

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers (of same class)
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via *discovery protocol*

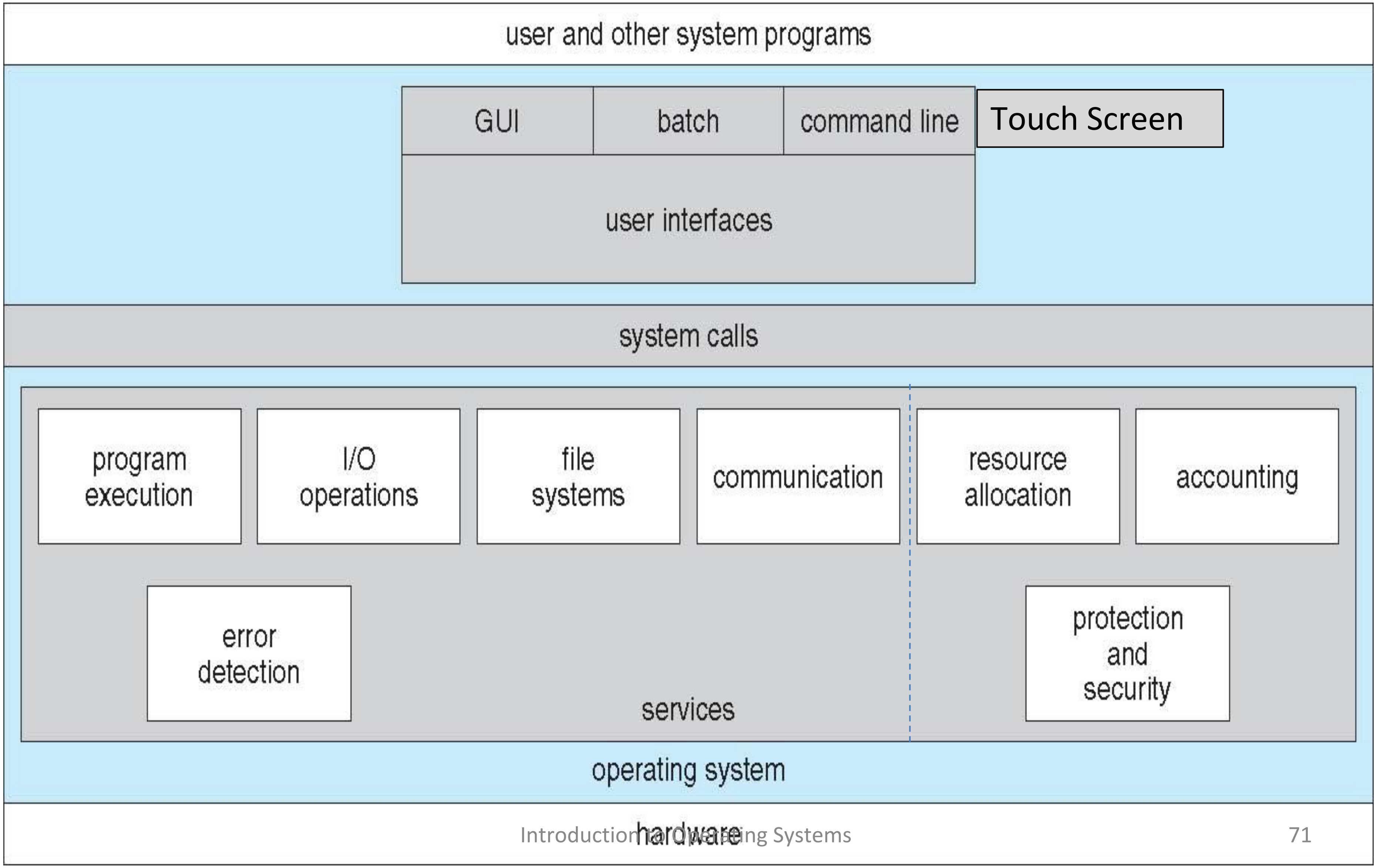
Web-Based Computing

- Web has become common everywhere.
- Devices use wired / wireless n/w to access web.
- New category of devices called **load balancers are used** to manage web traffic among similar servers.
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers

Open Source OS

- Free s/w **Available in source code format** rather than binary format.
- Student can modify OS, compile and execute to try out the changes.
- Security is more as many people are viewing the code. (Linux)

OS Services


user and other system programs
GUI
batch
command line
Touch Screen
user interfaces
system calls
program execution
I/O operations
file systems
communication
resource allocation
accounting
error detection
protection and security
services
operating system
hardware

OS Services

- OS is like a government. It provides an environment to execute a program.

- **<u>Services helpful to users</u>**
  - Provides user Interface
  - Helps in program execution
  - Control I/O operation
  - File system manipulation
  - Communications
  - Error detection

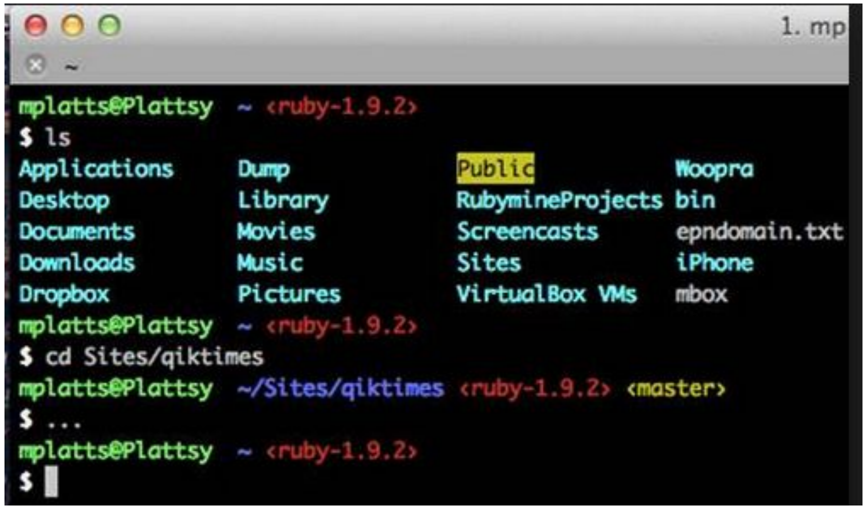
- **<u>Services for increasing efficiency of the system by sharng resources in Multiuser Environment</u>**
  - Resource allocation
  - Accounting

User interface:

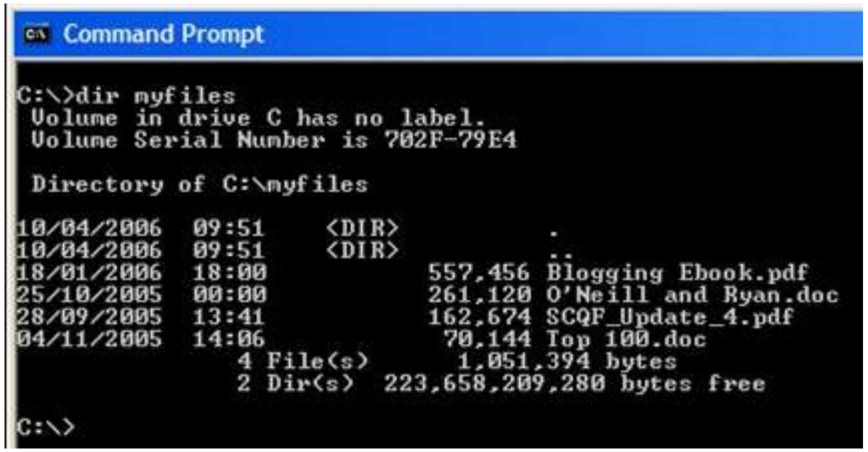
Type of Interfaces

User interface:

CLI, batch file,GUI directing to I/O operation, Touch screen



```
mplatts@Plattsy ~ <ruby-1.9.2>
$ ls
Applications    Dump        Public             Woopra
Desktop         Library     RubymineProjects   bin
Documents       Movies      Screencasts        epndomain.txt
Downloads       Music       Sites              iPhone
Dropbox         Pictures    VirtualBox VMs     mbox
mplatts@Plattsy ~ <ruby-1.9.2>
$ cd Sites/qiktimes
mplatts@Plattsy ~/Sites/qiktimes <ruby-1.9.2> <master>
$ ...
mplatts@Plattsy ~ <ruby-1.9.2>
$
```

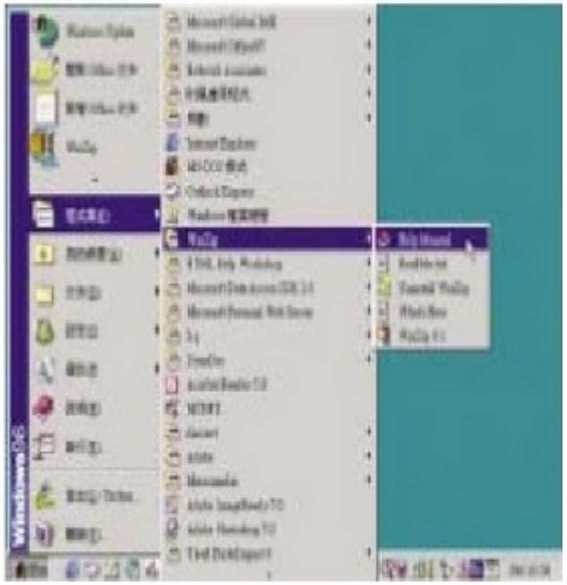


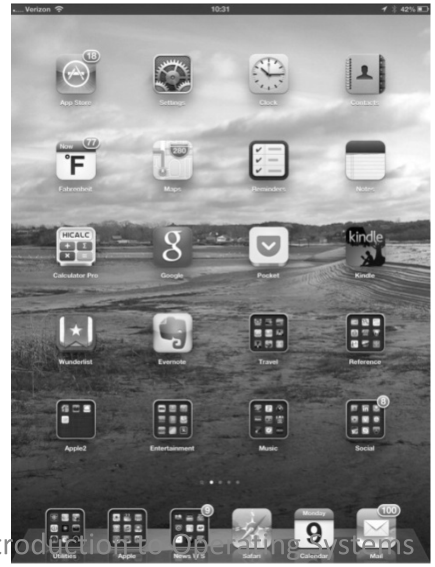
```
C:\>dir myfiles
 Volume in drive C has no label.
 Volume Serial Number is 702F-79E4

 Directory of C:\myfiles

10/04/2006  09:51    <DIR>          .
10/04/2006  09:51    <DIR>          ..
18/01/2006  18:00           557,456 Blogging Ebook.pdf
25/10/2005  00:00           261,120 O'Neill and Ryan.doc
28/09/2005  13:41           162,674 SCQF_Update_4.pdf
04/11/2005  14:06            70,144 Top 100.doc
               4 File(s)      1,051,394 bytes
               2 Dir(s)  223,658,209,280 bytes free

C:\>
```

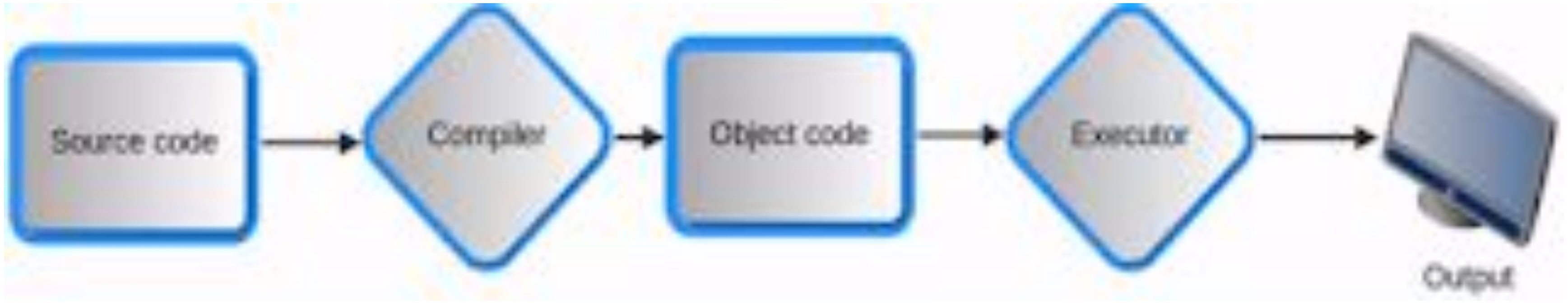




User interface:

- This is one of the most important services that is provided by the operating system.
- The user interface is something that allows the user to interact with the operating system or to interact with the computer itself almost all operating systems have a user interface.
- This user interface can take several forms like the command-line interface also known as CLI or the graphical user interface known as the GUI now examples of the command-line interface are like your command prompt that you have in Microsoft Windows or the terminal that you have in your open two based systems.
- So in this command interface you can provide text-based commands using your keyboard in order to perform certain tasks by the operating system and then the most commonly used user interface is the graphical user interface or the GUI.

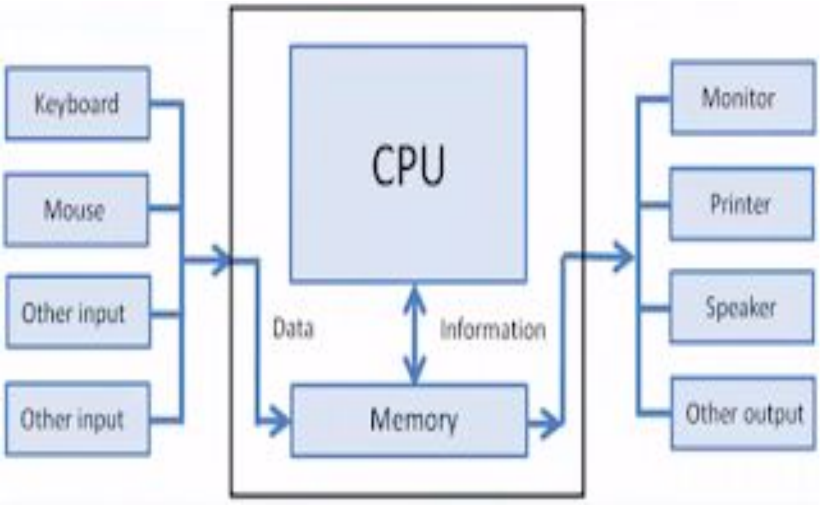
Program execution:



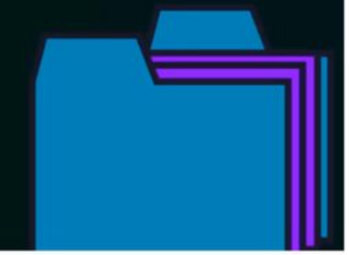
- We know what our programs and our software's, so the operating system must provide for the execution of the programs.
- You should be able to run or execute your programs and software's that you have, the system must be able to load the program into memory and it should be able to run that program.

- So this is also another very imp service that should be provided by an operating system.

Control I/O operations:

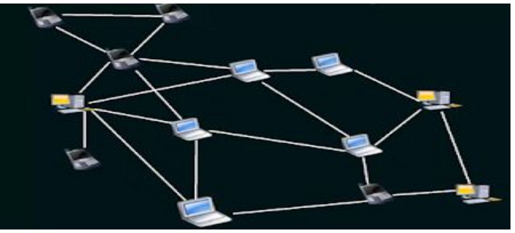


- A running program may require IO which may involve file or input/output devices and these input/output operations should be provided by the operating system.
- And a user cannot directly control the input and output devices.
- when you are using your keyboard and mouse you feel like you are controlling it by yourself directly but there is the operating system in between you and your system that actually controls the usage of the i/o devices.
  - For **efficiency or security** issue OS control i/o operations in a program that require a file in a device



File-system manipulation:

- This is also an important service and file system manipulation or file system management involves files.
- There are many files and directories in our system that has to be used.
- The operating system must control how these files are manipulated or how these files are managed so sometimes we may have to create files.
- Sometimes we may have to delete files, modify files, search for a given file.
- So all this is controlled by the operating system and also it also controls the permission that is given to certain programs or users for the access of certain files so all the files that we have cannot be allowed to be used by every program or by every user there is an access restriction.
- These restrictions of access is also controlled by the operating system.
  - Program may need to Create/delete /search files/directories, allow/deny permissions to files / directory.



Communication:

- There are many circumstances in which one process needs to exchange information with another process.
- Communications may occur between processes that may be executing on the same computer or between processes that are executing on different computers tied together by a computer network.
- So the communication between these processes between the same computers or even between the different computers is controlled by the operating system.

–OS helps in Inter-process communication via message passing or shared memory .

Error detection & correction:

- The operating system needs to be constantly aware of possible errors

- So to ensure correct and consistent computing, the operating system

Resource allocation:

- Resource allocation means allocating resources to different processes or different users.
- Resources can be of different types it may be the CPU it may be the files it may be the input/output devices. It may be the main memory and so on. There are so many resources that you have and there are many processes running in your system and all these processes require certain resources at a certain point of time. The operating system must help in a resource allocation means it should allocate the required resource to the processes which are waiting or which are asking for those resources and it must allocate them in an efficient way. Such that all the processes get the resources that they need and no process keeps waiting for the resource and never gets it.

  – For optimum use of resources
  – CPU cycles taking into account speed & no. Of jobs, main memory, storage,printer etc.

Accounting

- To keep track of which users use how much and what kind of computer resources, OS is required.
- This record-keeping may be used for accounting or simply for accumulating usage statistics.
- So by keeping an account of this or by having statistics of this usage it can be a valuable tool for researchers who wish to reconfigure the system or to improve the computing services.

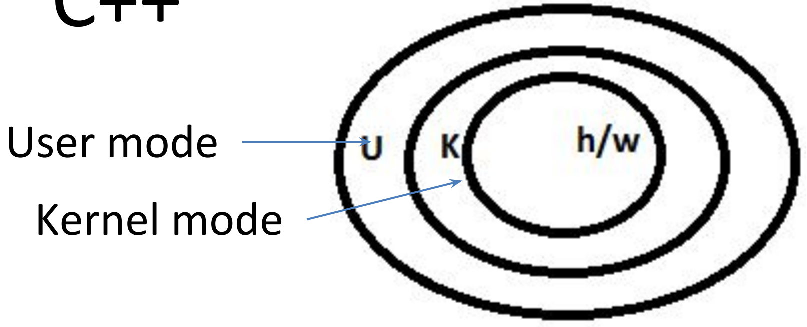
- So if you want to improve your computing services you need to know how the resources are used and how it is actually working so for this purpose accounting is an important service.

Protection & Security:

In multi-user environment protecting information , controlled access to system resources, synchronization between process, prohibiting interference of processes.
Security from outsiders using authentication, authorization

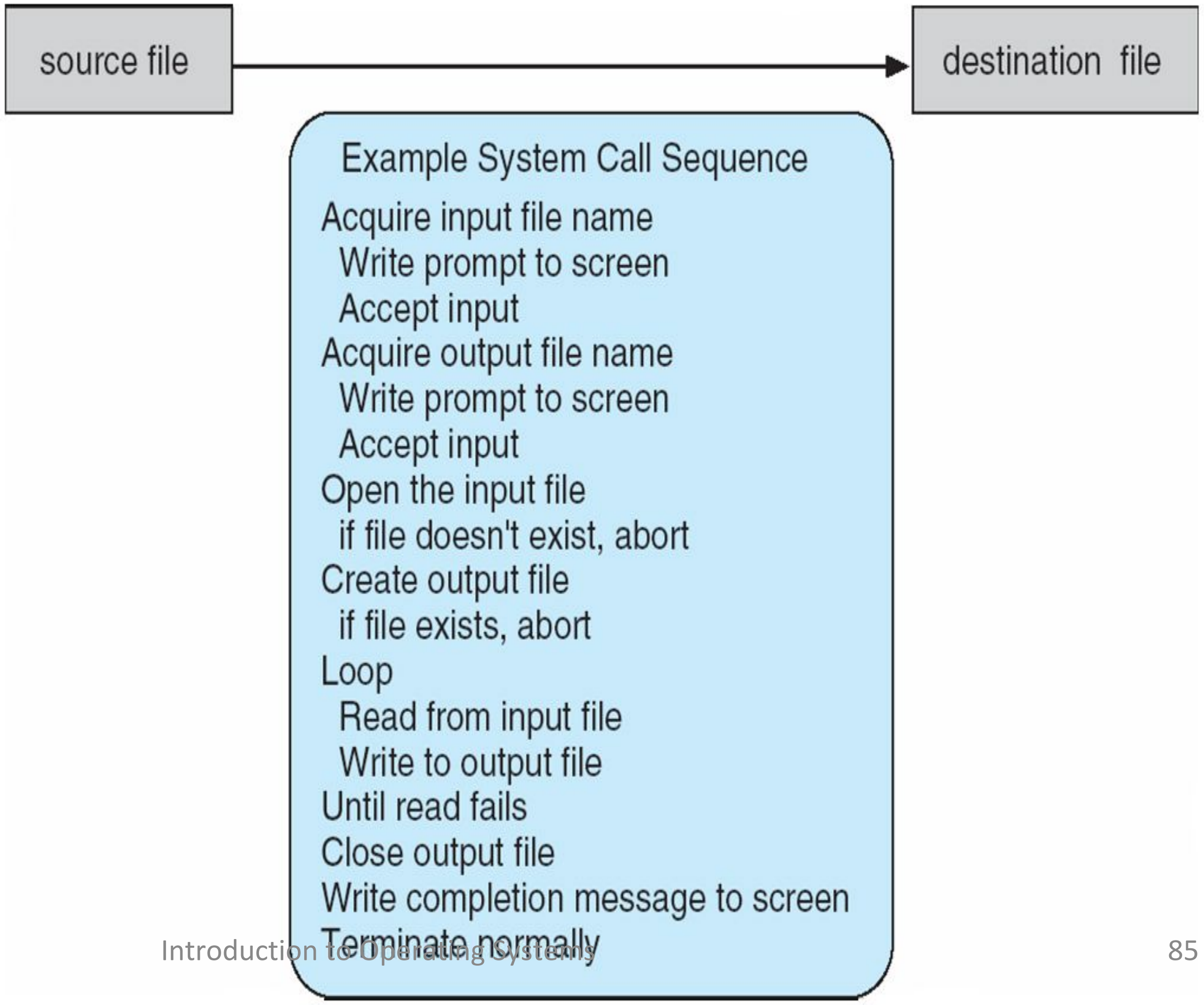
System Call

- System call is an interface to OS services.
- System calls are used in programs to access resources or to user services via kernel.
- System calls are available as routines in C or C++

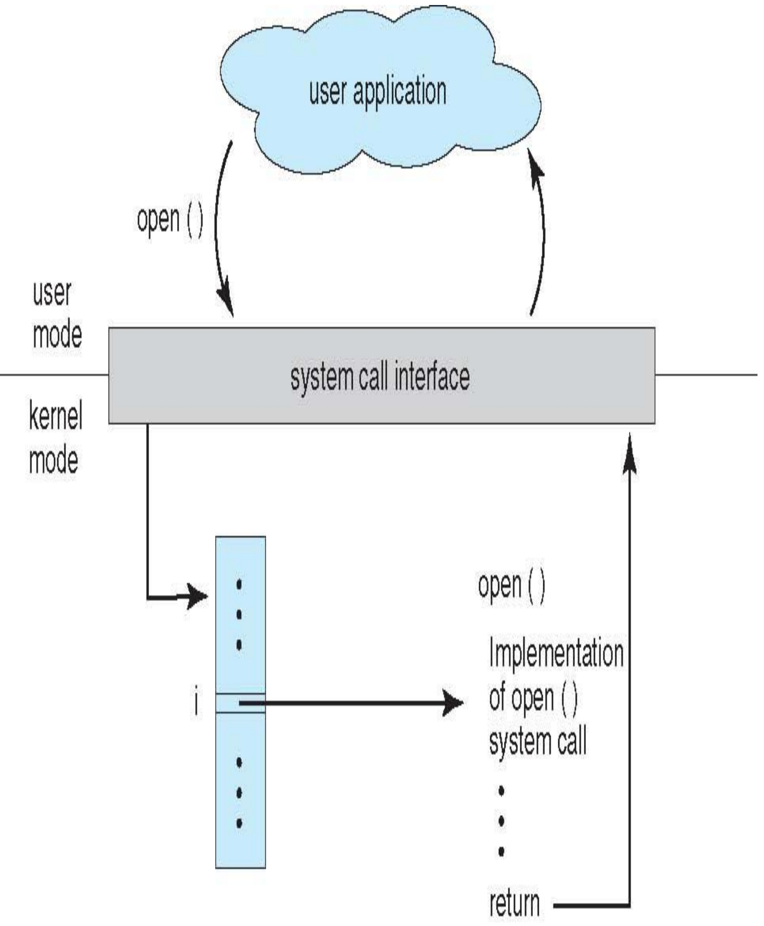


System Calls

- These calls are generally available as routines for doing low-level tasks like accessing hardware.
- System call sequence to copy the contents of one file to another file

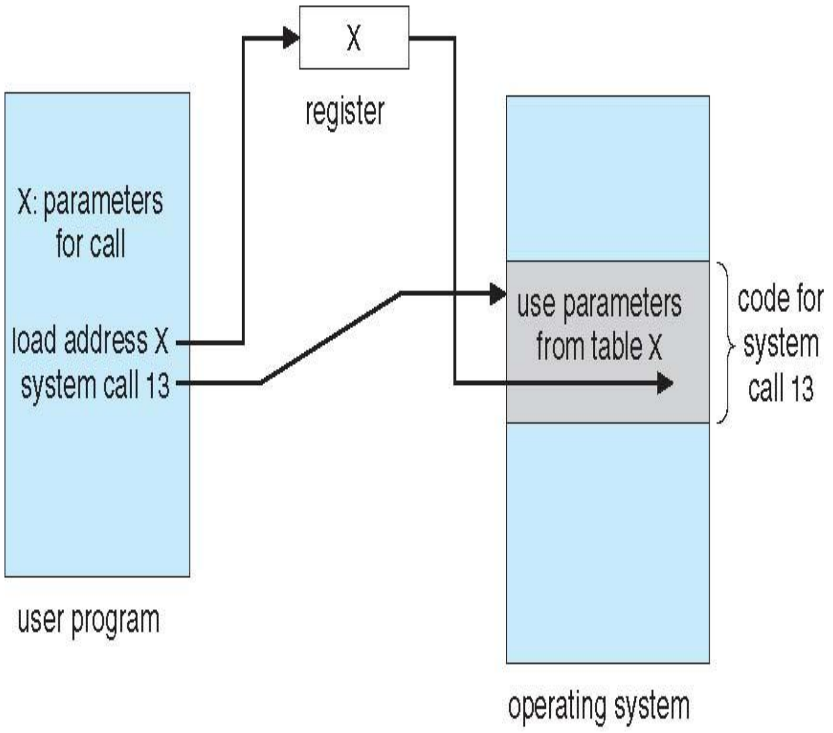


API for System call



- Mostly accessed by programs via a high-level **Application Programming Interface** (**API**) rather than direct system call use.
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- E.g. WIN32 API contain ReadFile() function.
- Runtime support : library function

Parameter Passing via Table



Three general methods exist for passing parameters to the OS:
Parameters can be passed in registers.
When there are more parameters than registers, parameters can be stored in a block and the block address can be passed as a parameter to a register.
Parameters can also be pushed on or popped off the stack by the operating system.

System Calls Types

- System calls are available for
  - Process management
  - Device manipulation
  - File manipulation
  - Communication
  - protection

Types of System Calls

- Process control
- This system calls perform the task of process creation, process termination, etc.
  - create process, terminate process
  - end, abort
  - load, execute
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
  - Dump memory if error
  - **Debugger** for determining **bugs, single step** execution
  - **Locks** for managing access to shared data between processes
- File management
- File management system calls handle file manipulation jobs like creating a file, reading, and writing, etc
  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes

Types of System Calls

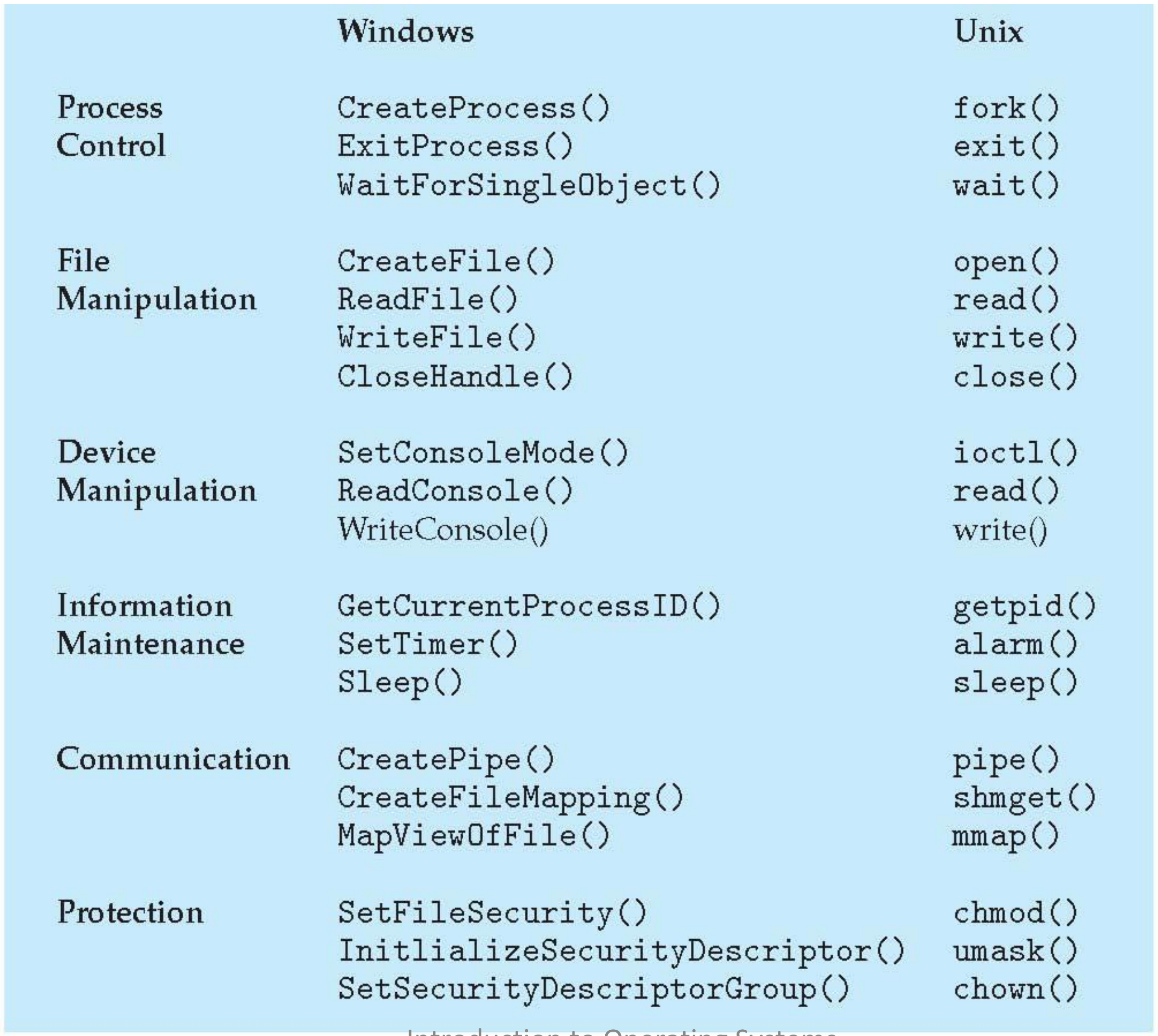
- Device management
- Device management does the job of device manipulation like reading from device buffers, writing into device buffers, etc.
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices

- Information maintenance
- It handles information and its transfer between the OS and the user program.
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes

Types of System Calls (Cont.)

- Communications
- These types of system calls are specially used for interprocess communications.
  - create, delete communication connection
  - send, receive messages if **message passing model** to **host name** or **process name**
    - From **client** to **server**
  - **Shared-memory model** create and gain access to memory regions
  - transfer status information
  - attach and detach remote devices
- Protection
  - Control access to resources
  - Get and set permissions
  - Allow and deny user access

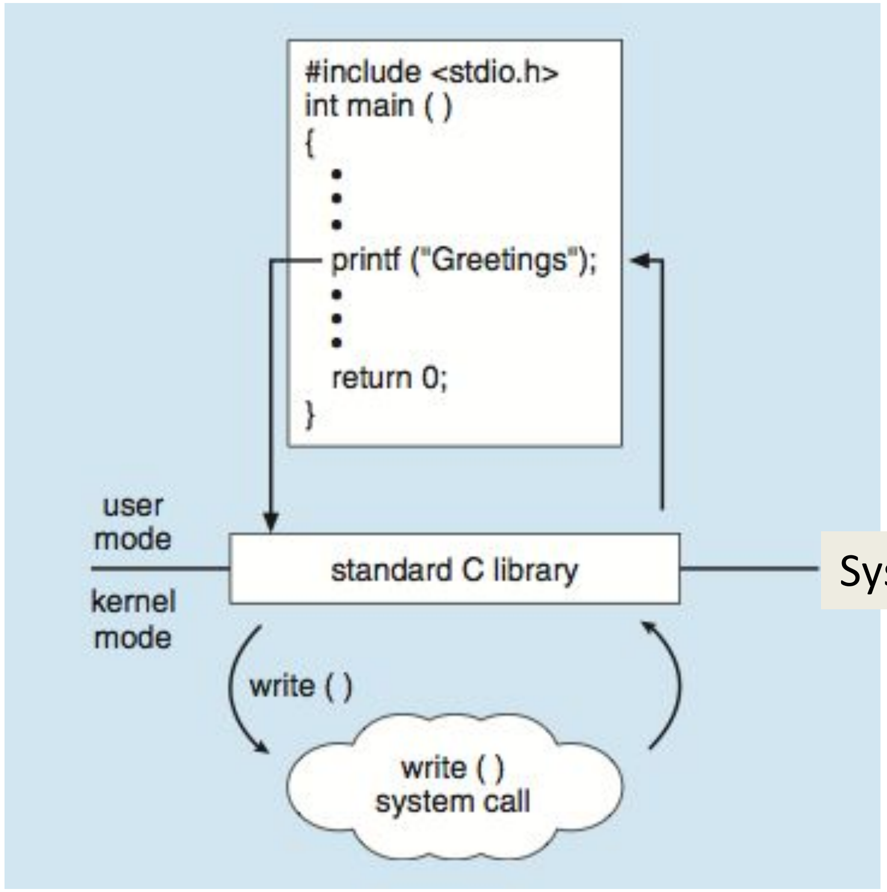
Examples of Windows and Unix System Calls



	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()

System Calls

- The system call interface (library functions) invokes the intended system call in OS kernel and returns status of the system call and any return values

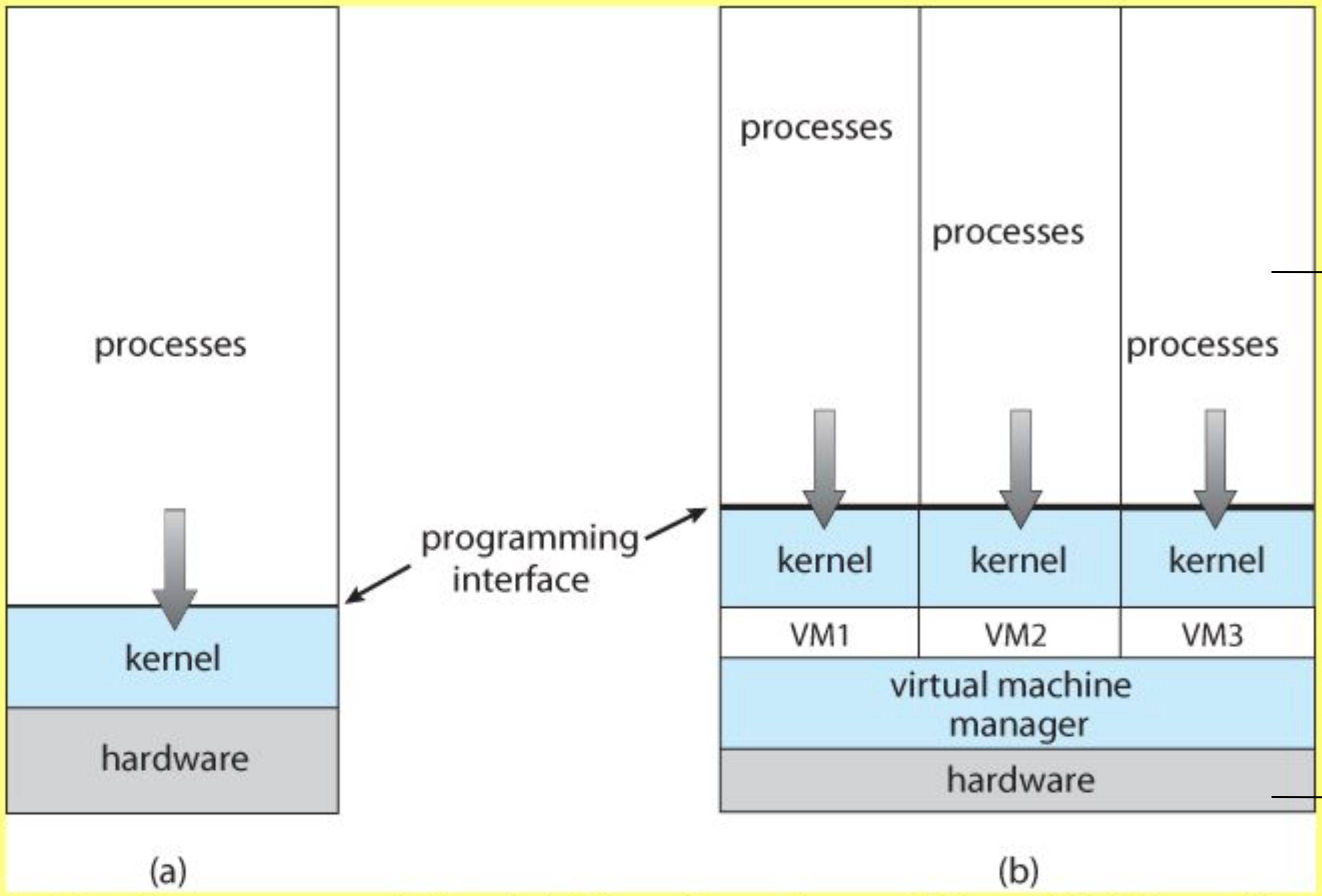


```
#include <stdio.h>
int main ( )
{
   •
   •
   •
   printf ("Greetings");
   •
   •
   •
   return 0;
}
```

System Call Interface

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

Virtual Machines



Guest m/c

Host m/c

Figure 16.1 - System models. (a) Nonvirtual machine. (b)Virtual machine.

Virtual M/c

- Available h/w of a m/c can be abstracted to use into several execution environments thereby creating illusion that each execution environment is running on own private computer.
  - **Advantages**
  - We can run different os concurrently
  - Can share memory,
  - communicate,
  - protection.

  - **Disadvantages**
  - Running more than one virtual machine can lead to an unstable output.
  - It is less efficient and slow running than a physical machine.

File System

- Introduction
- Directory system
- Access methods
- Allocation methods
- Disk and drum scheduling.

File Concept

- Computers can store information on various storage media, such as magnetic disks, magnetic tapes, and optical disks. So that the computer system will be convenient to use, the operating system provides a uniform logical view of information storage.
- The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, the *file.*
- Files are mapped by the operating system onto physical devices.
- These storage devices are usually non volatile, so the contents are persistent through power failures and system reboots.

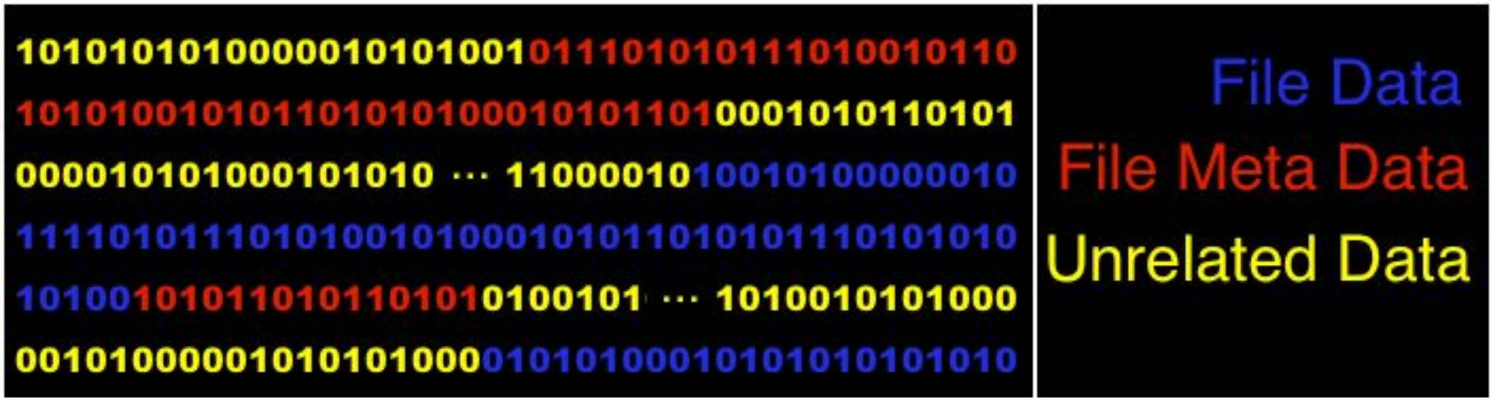
File Concept

- Logical storage unit.
- **It is a named collection of related information recorded on secondary storage.**
- File can be a program file or data file.
- Data files may be numeric, alphanumeric, or binary
- Program file may be a text file or formatted file.
- Since main memory is usually too small to accommodate data and programs, secondary storage like hard-disk is used as back up to main memory and store data permanently.

- File System provides the mechanism to **store and access data and programs and related information from storage device.**

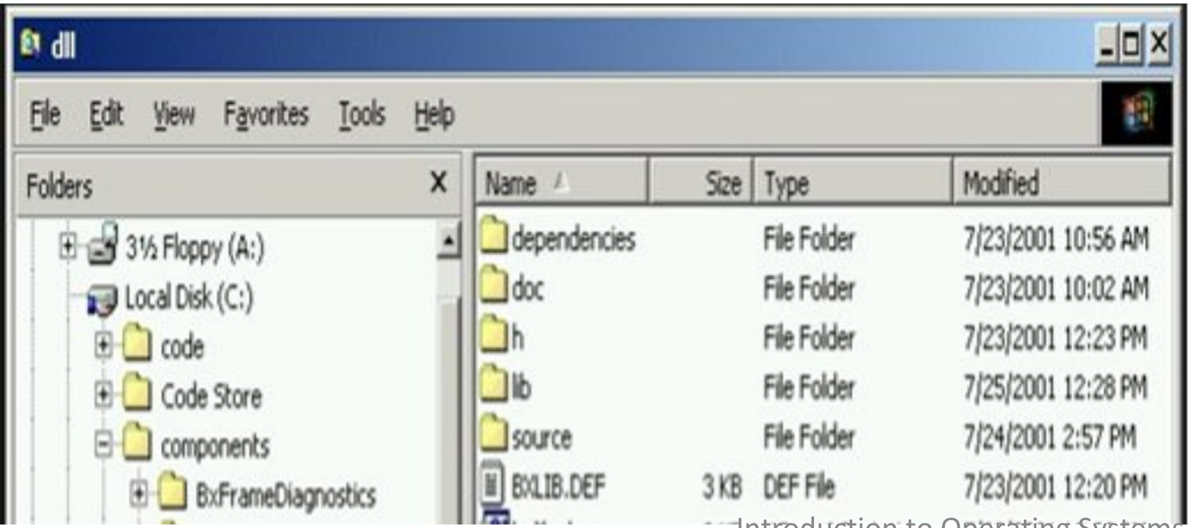
- **File system consist of two parts:**
  - **Directories**
  - **files**

FS Introduction



Hard disk

File system: bridges the gap



Name	Size	Type	Modified
dependencies		File Folder	7/23/2001 10:56 AM
doc		File Folder	7/23/2001 10:02 AM
h		File Folder	7/23/2001 12:23 PM
lib		File Folder	7/25/2001 12:28 PM
source		File Folder	7/24/2001 2:57 PM
BXLIB.DEF	3 KB	DEF File	7/23/2001 12:20 PM

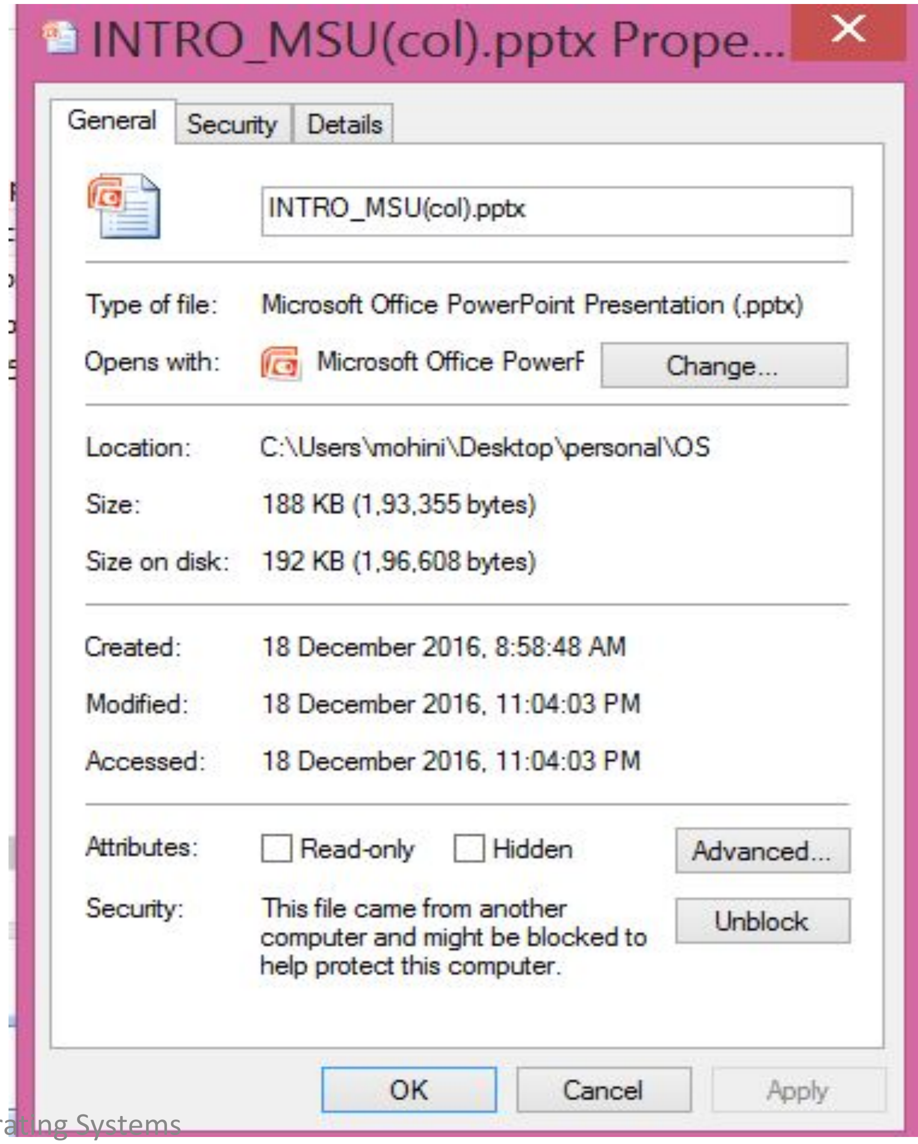
Nested folder abstraction
Of hard disk data

File Attributes

- *A* file is named, for the convenience of its human users, and is referred to by its name.
- *A* name is usually a string of characters, such as *example.c.* Some systems differentiate between uppercase and lowercase characters in names, whereas other systems do not.
- When a file is named, it becomes independent of the process, the user, and even the system that created it.
- For instance, one user might create the file *example.c,* and another user might edit that file by specifying its name.

File Attributes

- Name
- Identifier
- Type
- Location
- Size
- Date, time and user identification
- protection



INTRO_MSU(col).pptx

Type of file: Microsoft Office PowerPoint Presentation (.pptx)

Opens with: Microsoft Office PowerF

Location: C:\Users\mohini\Desktop\personal\OS

Size: 188 KB (1,93,355 bytes)

Size on disk: 192 KB (1,96,608 bytes)

Created: 18 December 2016, 8:58:48 AM

Modified: 18 December 2016, 11:04:03 PM

Accessed: 18 December 2016, 11:04:03 PM

Attributes: ☐ Read-only ☐ Hidden

Security: This file came from another computer and might be blocked to help protect this computer.

File Attributes

- *A* file's attributes vary from one operating system to another but typically consist of these:
- 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.
- Type: This information is needed for systems that support different types of files.
- Location: This information is a pointer to a device and to the location of the file on that device.
- Size: The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.
- Protection: Access-control information determines who can do reading, writing, executing, and so on.
- Time, date, and user identification: This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

File Operations

- Creating a file: Two steps are necessary to create a file. First, space in the file system must be found for the file. Second, an entry for the new file must be made in the directory.
- Writing a file: To write a file, we make a system call specifying both the name of the file and the information to be written to the file. Given the name of the file, the system searches the directory to find the file's location. The system must keep a *write* pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
- Reading a file: To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a *read* pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated. Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process . Both the read and write operations use this same pointer, saving space and reducing system complexity.

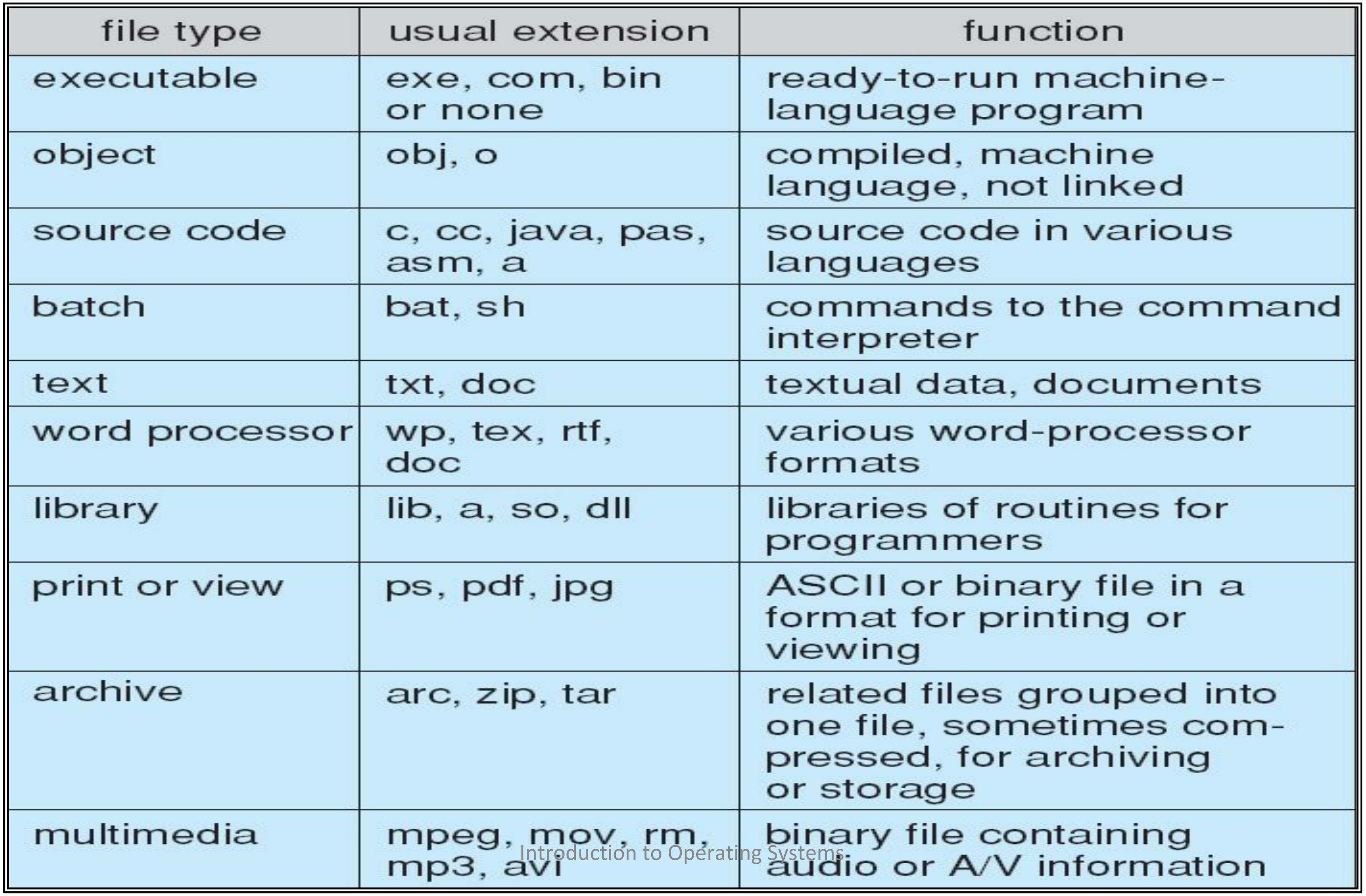
File Operations

- Repositioning within a file: The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/0. This file operation is also known as a file seek.
- Deleting a file: To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
- Truncating a file: The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged-except for file length-but lets the file be reset to length zero and its file space released.

File types

- A common technique for implementing file types is to include the type as part of the file name.
- The name is split into two parts-a name and an extension, usually separated by a period character. In this way, the user and the operating system can tell from the name alone what the type of a file is.
- For example, most operating systems allow users to specify a file name as a sequence of characters followed by a period and terminated by an extension of additional characters.
- File name examples include resume.doc, Server.java, and ReaderThread. C. The system uses the extension to indicate the type of the file and the type of operations that can be done on that file.
- The .com and .exe files are two forms of binary executable files, whereas a .bat file is a containing, in ASCII format, commands to the operating system.

Common file types

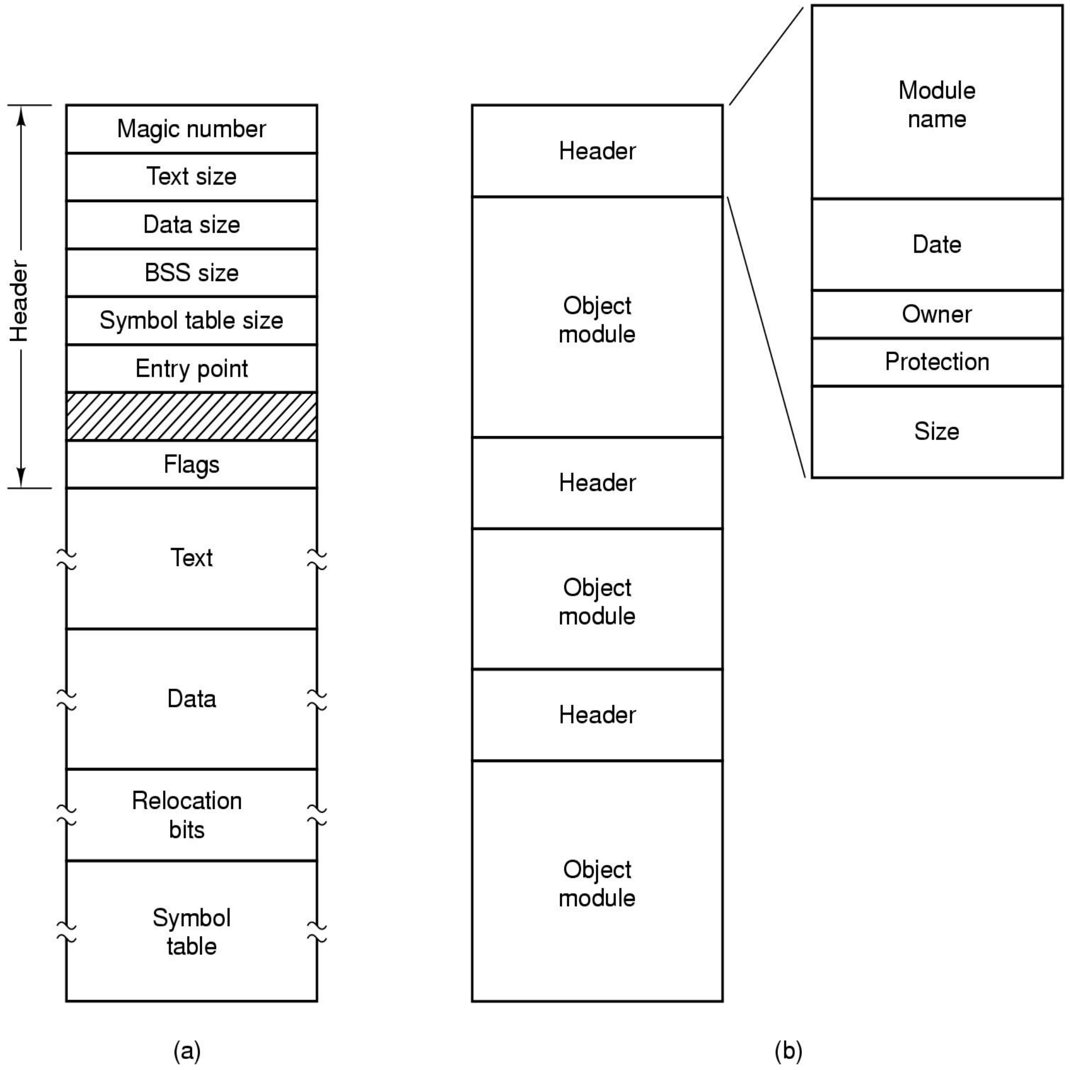


file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual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

File Structure

- File System consist of two components
  - Files
  - Directory structure
    - Organizes and provides information about files.

- File structure depends on its type.
  - A text file is a sequence of characters organized into lines.
  - A source file is a sequence of procedures and functions.
  - An object file is a sequence of bytes organized into blocks that are understandable by the machine.

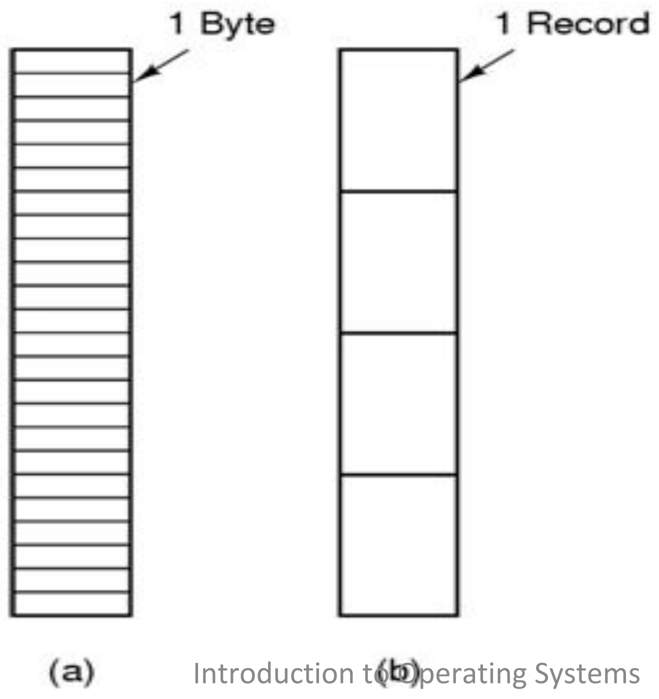


Exe file

Archive File

File Structure

- OS Type
  - Unix considers each file to be a stream of bytes.
  - DOS considers each file as a stream of records.
    - Problem of fragmentation



Internal File Structure

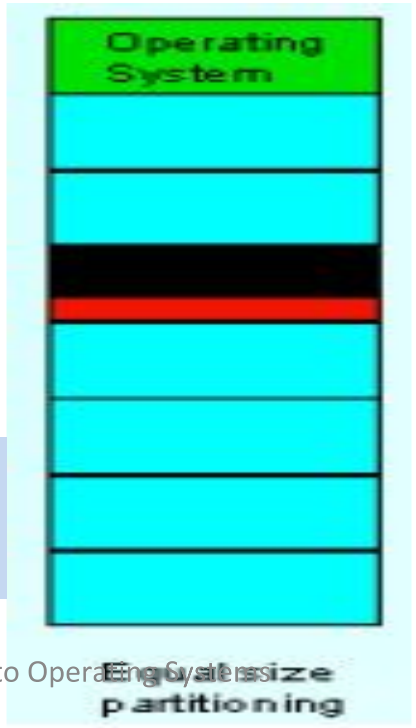
- Internally, locating an offset within a file can be complicated for the OS.
- Disk systems typically have a well-defined block size determined by the size of a sector.
- All disk I/O is performed in units of one block (physical rercod), and all blocks are the same size.
- It is unlikely that the physical record size will exactly match the length of the desired logical record.
- Packing a number of logical records into physical blocks is a common solution to this problem.
- For example, the UNIX OS defines all files to be simply streams of bytes. Each byte is individually addressable by its offset from the beginning (or end) of the file.
- In this case, the logical record size is 1 byte.

Internal File Structure

- The file system automatically packs and unpacks bytes into physical disk blocks -say, 512 bytes per block- as necessary.
- The file may be considered to be a sequence of blocks.
- All the basic I/O functions operate in terms of blocks. Because disk space is always allocated in blocks, some portion of the last block of each file is generally wasted. If each block were 512 bytes, for example, then a file of 1,949 bytes would be allocated four blocks (2,048 bytes); the last 99 bytes would be wasted.
- The waste incurred to keep everything in units of blocks (instead of bytes) is internal fragmentation.
- All file systems suffer from internal fragmentation; the larger the block size, the greater the internal fragmentation.

Internal Fragmentation

- i/o is performed in terms of blocks.
- The wastage of space due to equal size blocks. fixed allocated blocks.
- Unequal block size can remove the problem.
- E.g wastage of 99 bytes if file is 1949 bytes (for 512 bytes/ block)
-



1949/512=3.80
512*4=2048 bytes

413

512

512

512

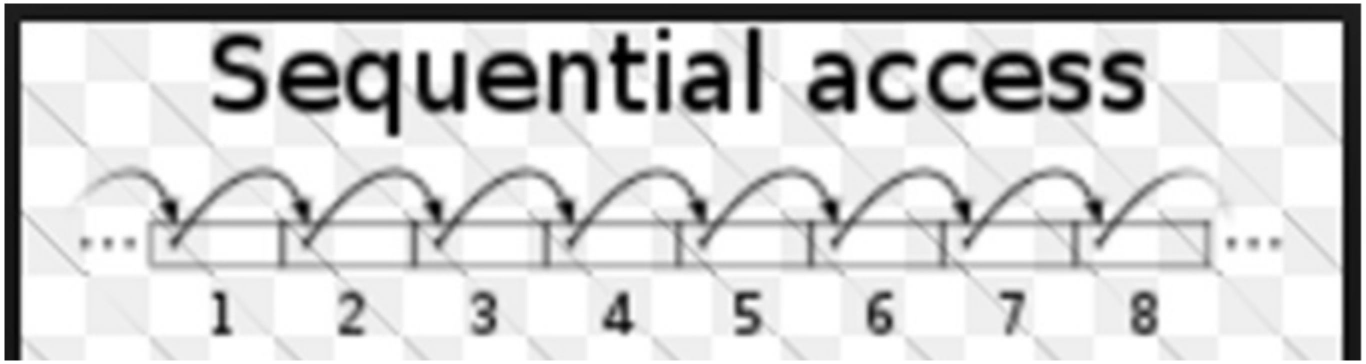
Wastage of space within a block
Because of fixed size block partitioning.

Access Methods

- Information in the file can be accessed using various methods.
- Access method depends on device and file organization method.
- Access Methods:
  - Sequential Access
  - Direct Access
  - Index Sequential Access
- Some OS employ more than one access method.

Sequential Access

- Most simple and common method of data access.
- read all bytes/records from the beginning
- cannot jump around, could rewind or back up



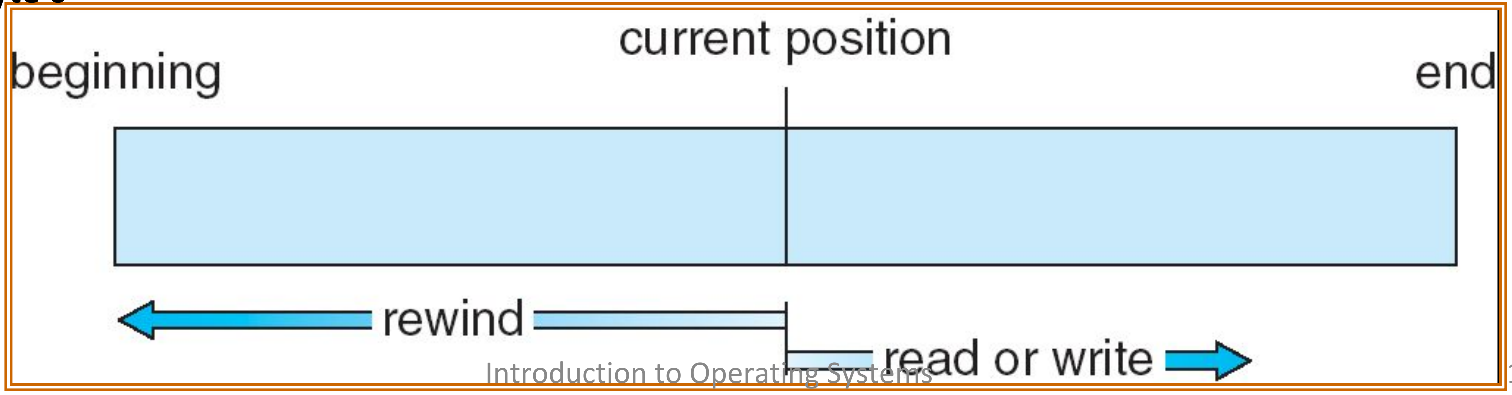
- Device can be magnetic tape or Disk Drive.
- Followed by editors and compilers

Sequential Access

- **Operations**
  - read next: read next portion of the file and advances the pointer
  - write next: append and modifies EOF pointer.
  - reset: rewind and reset pointer to the beginning
  - program can be written to skip n records forward or backward.

Record / Byte 0

Record / Byte n



Sequential Access

- The OS read the file word by word.
- A pointer is maintained which initially points to the base address of the file.
- If the user wants to read first word of the file then the pointer provides that word to the user and increases its value by 1 word.
- This process continues till the end of the file.
- Most of the files such as text files, audio files, video files, etc need to be sequentially accessed.

Sequential Access

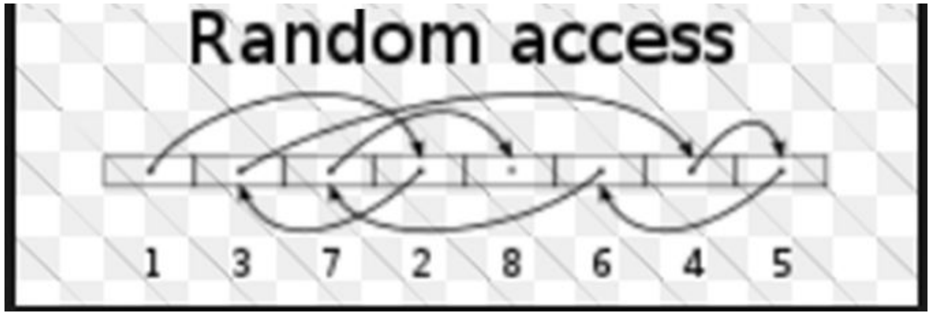
- **Advantages of Sequential Access Method**
- It is one of the simplest method of file access and easy to implement.
- There is no need for any storage space identification.
- It uses disk and memory efficiently.
- It also allows data to be stored on many types of media, in a device independent manner.
- Errors in the files remain localized.
- It is also economical and easier to organize and maintain.
- **Disadvantages of Sequential access method**
- Searching a record is a time consuming affair as it allows only sequential access. To search a record at nth location entire file needs to be processed.
- New records can only be added to the end of a file.
- It requires the transactions to be sorted in a particular sequence before processing.
- There is high data redundancy.
- It is not possible to handle the random enquiries.

Direct Access / Relative Access

- File is viewed as numbered sequence of fixed length blocks or records
- There are no restrictions on the order of reading or writing direct-access file.
- Programs can read file in any order. Eg. Block 14 then block 53 and then block 7.
- E.g. database application like flight reservation system
- file operations uses relative record number or block number as parameter.

– Two approaches

- **Read n**
- **Write n**



Advantages –

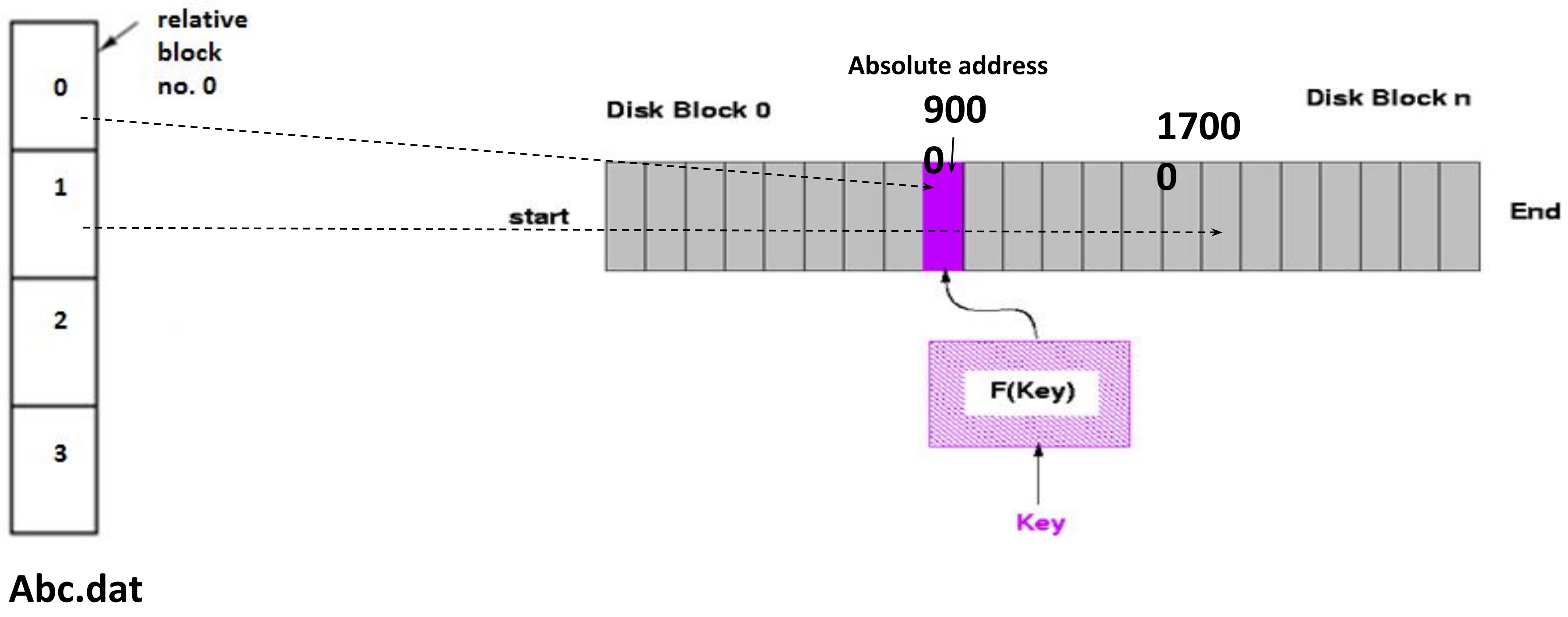
- The direct access method solves the problem of sequential access and helps to access the information in the file immediately.
- The best example of direct file access is the database system.
- When the database system receives queries, it calculates which block has the desired information and then returns the information to the user.
- We can simulate the sequential access method with the direct access method but vice versa can be inefficient and complex.
- It results in faster access of desired record. As a result retrieval process is not so time consuming as in direct access.
- It does not require records to be sorted before processing.
- It allows faster updating of several files.
- It also allows the random record deletion and insertion.
- It is best suited for online transaction processing system like online reservation system.

Disadvantages-

- This method requires backup facility as records are directly updated.
- It requires expensive direct access storage devices such as hard disk to store records.
- Less efficient as compared to sequential file organization in terms of usage of storage space.
- Data may be accidentally erased or overwritten unless special precautions are taken.
- It is not possible to access such a file sequentially.

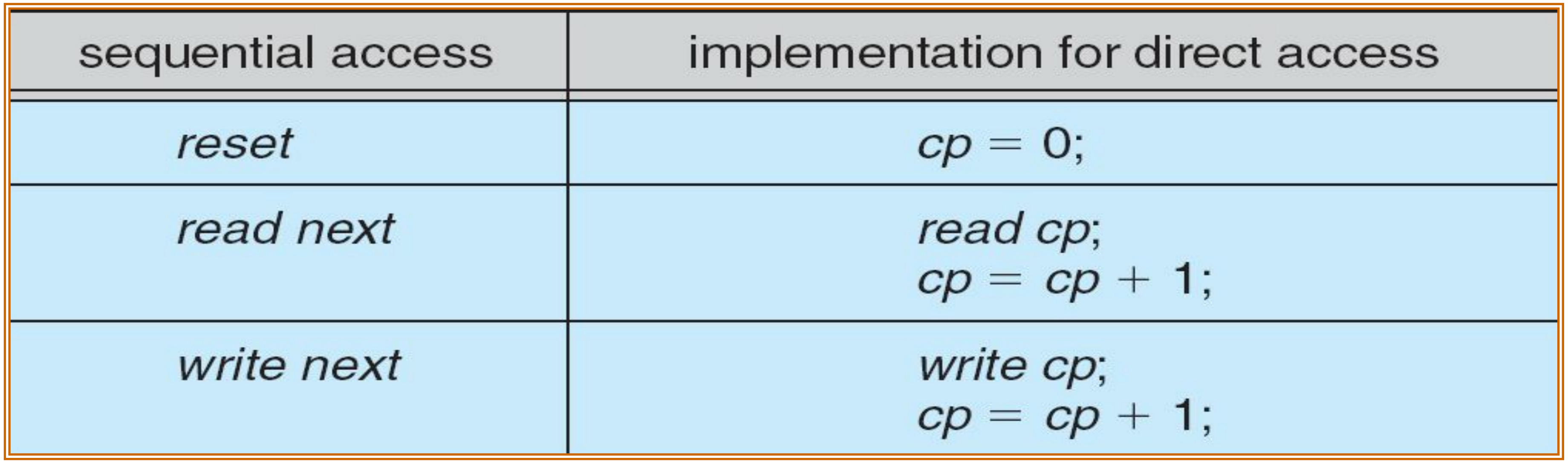
Absolute vs Relative Block Address

- File is viewed as numbered sequence of blocks/ records
- File access using Relative address.



Access Methods contd........
Simulation of sequential access on direct address

- Assume cp is current position



sequential access	implementation for direct access
reset	$cp = 0;$
read next	*read cp*; $cp = cp + 1;$
write next	*write cp*; $cp = cp + 1;$

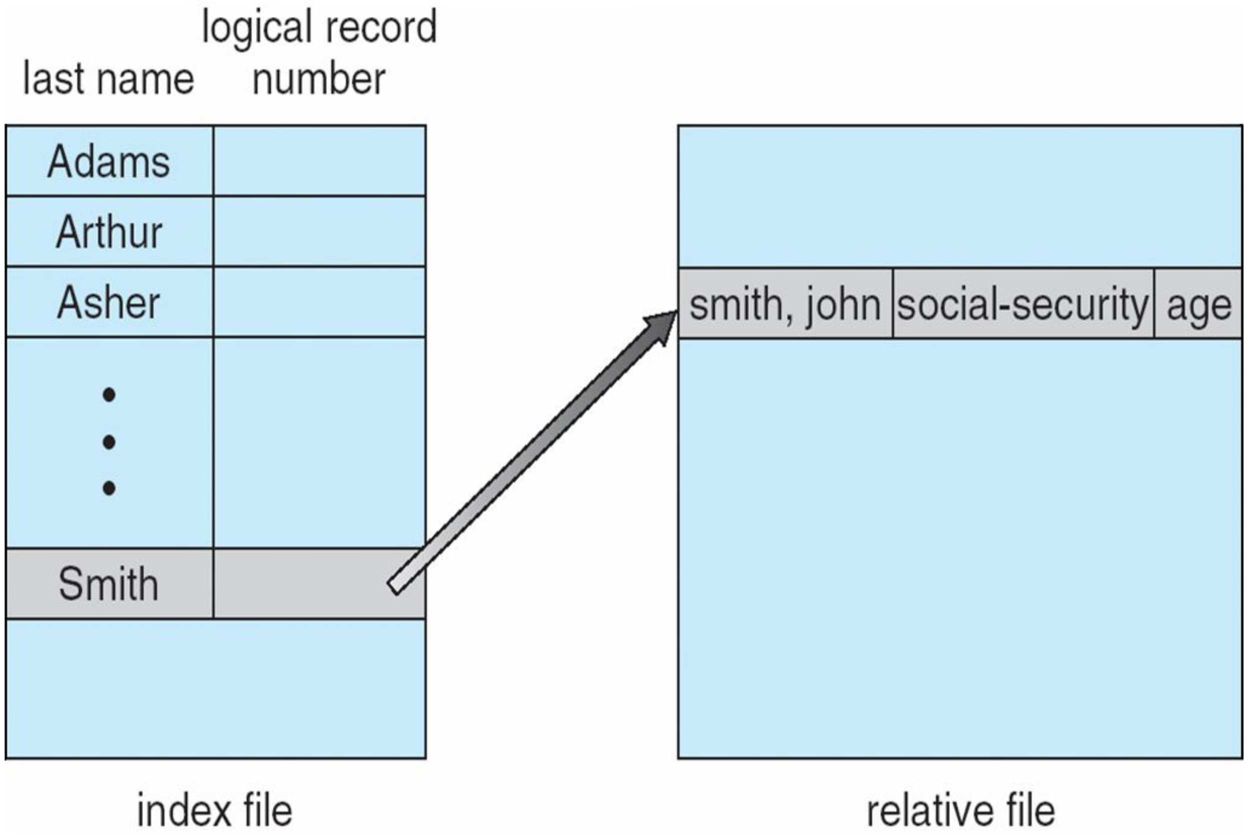
Indexed Access Method

- Index::
  - Fast retrieval of data
  - Contain key value and address of record
  - File kept sorted on a defined key
  - If index is too large use multilevel indexing.
  - Small master index in memory , points to disk blocks of secondary index

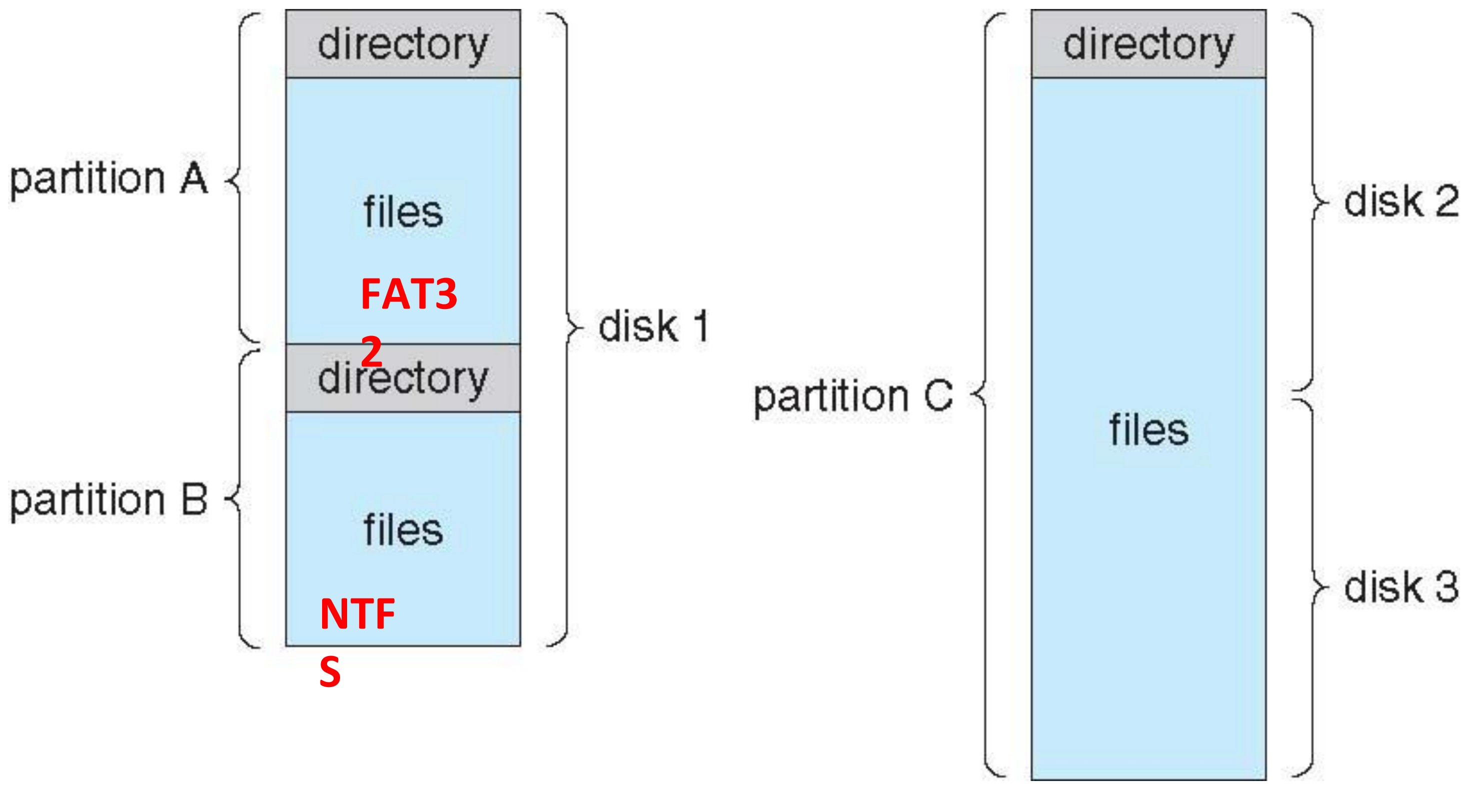
- **Advantage of indexed access method**
- This method allows records to be processed both sequentially and randomly in a efficient manner.
- Accessing of records is faster and less time consuming but requires index table to be properly organized.

- **Disadvantage of indexed access method**
- It requires a lot of storage space because of the presence of index.
- As compared to other access method, indexed access is less efficient in terms of usage of storage space
- The indexed files have to be reorganized from time to time to get rid of deleted records and improve performance that gets gradually decreased with addition of new records.
- It is also expensive as it requires special software and extra storage for index.

Example of Index and Relative Files in VMS OS



A Typical File-system Organization
Disk Structure



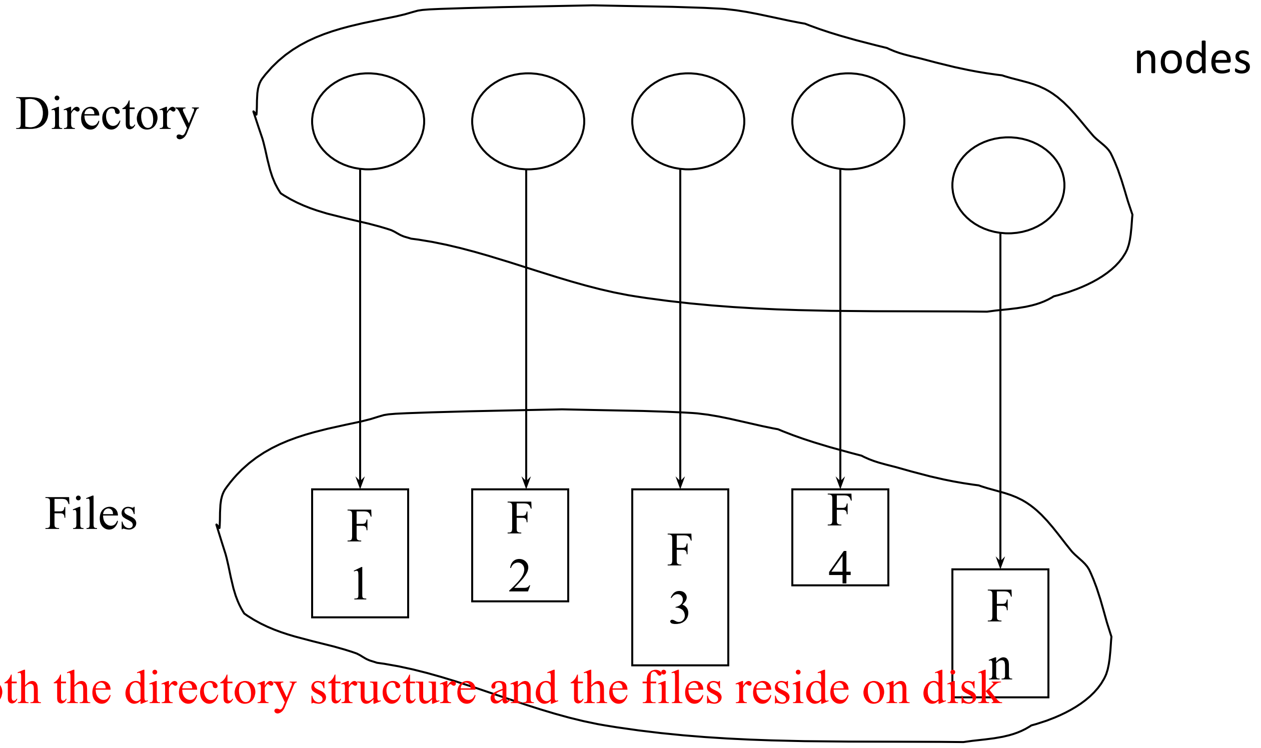
raw space– unformatted without a file system

Swap space

Disks or partitions can be **RAID** protected against failure

Directory Structure: Introduction

- Advantages of Partitioning:
  - limit the individual file system ,
  - putting multiple file systems on same
- Each volume containing file system also tracks that file system's info in **device directory** or **volume table of contents**
- Create, list, searching file, delete, rename operations on directory.



Both the directory structure and the files reside on disk

Directory Organization

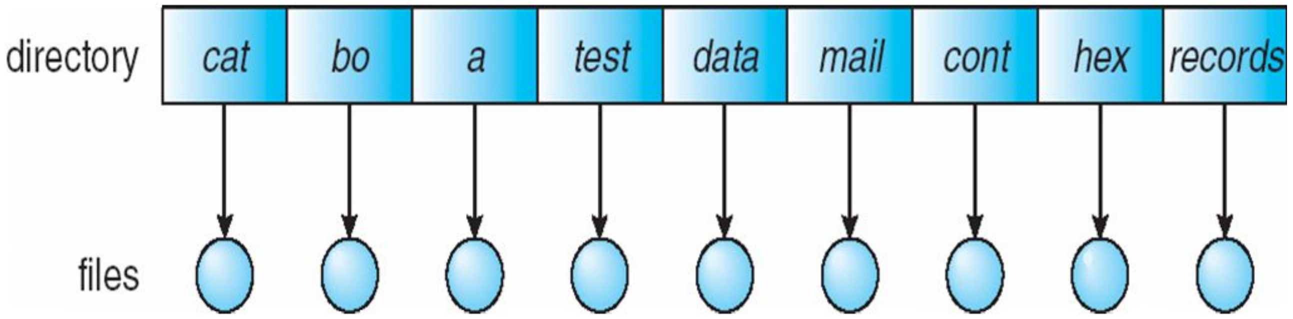
Advantages:

- Grouping – by properties
- Efficiency – locating a file quickly
- Naming – convenient to users

- **Schemes for defining logical structure of directory**
- Single-Level Directory
- Two-Level Directory
- Tree-Structured Directory
- Acyclic Graph Directory
- General Graph Directory

Single-Level Directory

- A single directory for all users. Simplest directory structure
- All files are contained in the same directory, which is easy to support and understand
- It has significant limitations, however, when the number of files increases or when the system has more than one user. Since all files are in the same directory, they must have unique names.
- If two users call their data file *test,* then the unique-name rule is violated.
- For example, in one programming class, 23 students called the program for their second assignment *prog2;* another 11 called it *assign2.* Although file names are generally selected to reflect the content of the file, they are often limited in length, complicating the task of making file names unique.
- The MS-DOS operating system allows only 11-character file names; UNIX, in contrast, allows 255 characters.
- Even a single user on a single-level directory may find it difficult to remember the names of all the files as the number of files increases.
- It is not uncommon for a user to have hundreds of files on one computer system and an equal number of additional files on another system.



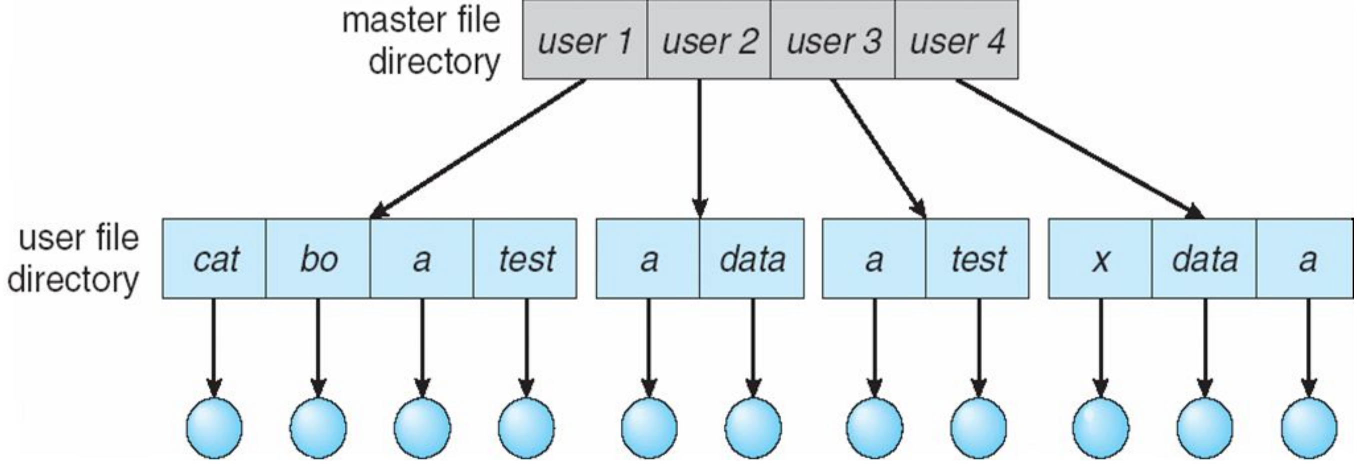
Single-Level Directory

- Advantages:
  - Simplest data structure
- Disadvantages:
  - Problem of name-collision.
  - Security
  - Difficult to remember or manage if number files increases

- Simplest data structure to implement but difficult to manage

Two-Level Directory

- A single-level directory often leads to confusion of file names among different users.
- The standard solution is to create a *separate* directory for each user.
- In the two-level directory structure, each user has his own User file directory.
- The UFDs have similar structures, but each lists only the files of a single user.
- When a user job starts or a user logs in, the system's Master file Directory is searched.
- The MFD is indexed by user name or account number, and each entry points to the UFD for that user. When a user refers to a particular file, only his own UFD is searched.
- Thus, different users may have files with the same name, as long as all the file names within each UFD are unique.
- To create a file for a user, the operating system searches only that user's UFD to ascertain whether another file of that name exists.

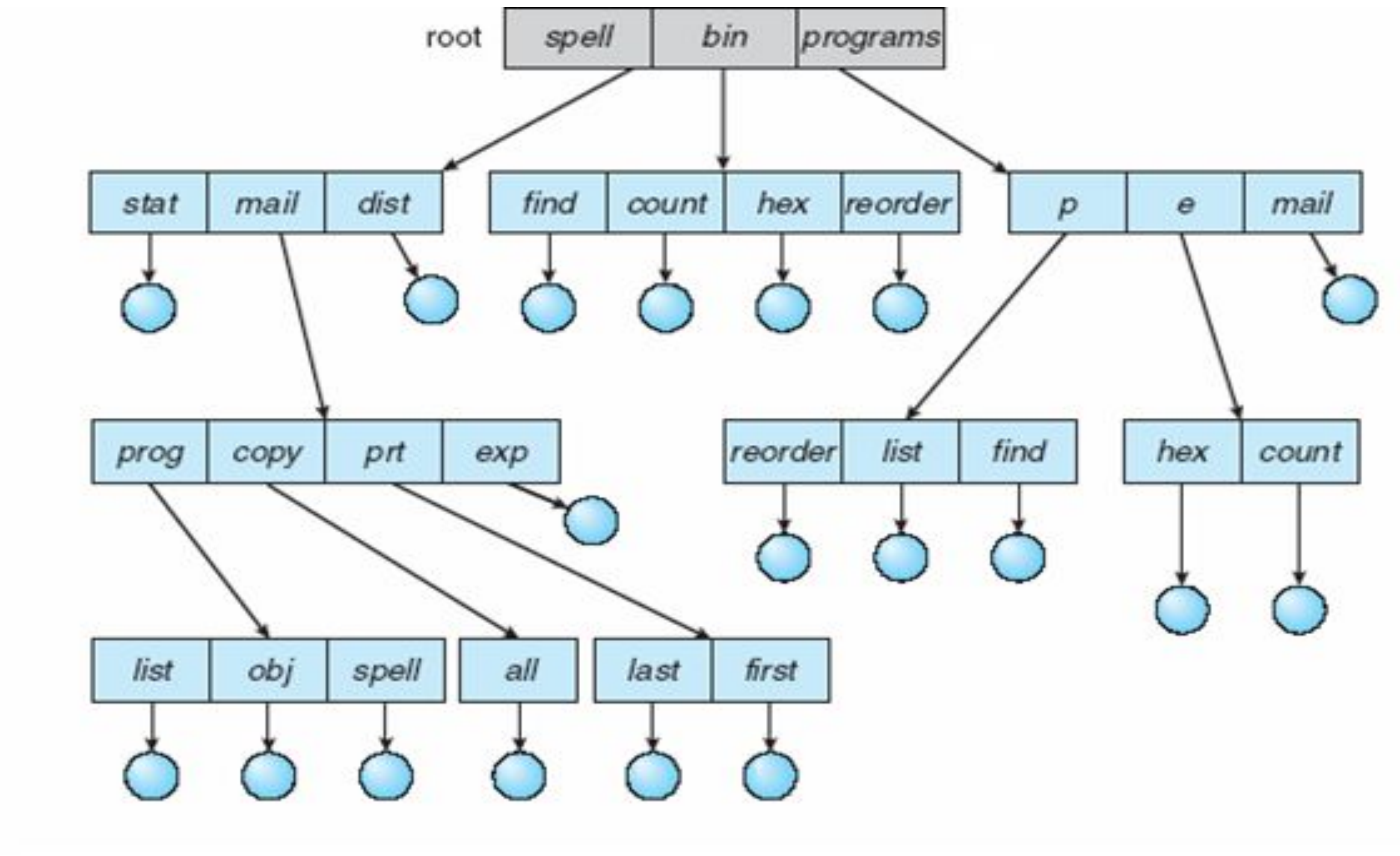


- To delete a file, the operating system confines its search to the local UFD; thus, it cannot accidentally delete another user's file that has the same name.
- Although the two-level directory structure solves the name-collision problem, it still has disadvantages.
- This structure effectively isolates one user from another. Isolation is an advantage when the users are completely independent but is a disadvantage when the users *want* to cooperate on some task and to access one another's files.
- Some systems simply do not allow local user files to be accessed by other users.
- For example, if user A wishes to access her own test file named *test,* she can simply refer to *test.* To access the file named *test* of user B (with directory-entry name *userb),* however, she might have to refer to */userb/test.* Every system has its own syntax for naming files in directories other than the user's own.

Two-Level Directory

- Advantages:
  - Reduce search space
  - Efficient searching
  - No name collision
  - Files can be shared by giving user name and file name, sometimes volume label
- **Disadvantages:**
  - User **can’t create subdirectories.**
  - Problem of accessing **system files**

Tree-Structured Directories



- To view a two-level directory as a two-level tree, the natural generalization is to extend the directory structure to a tree of arbitrary height.
- This generalization allows users to create their own subdirectories and to organize their files accordingly.
- A tree is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name.

Tree-Structured Directories (Cont.)

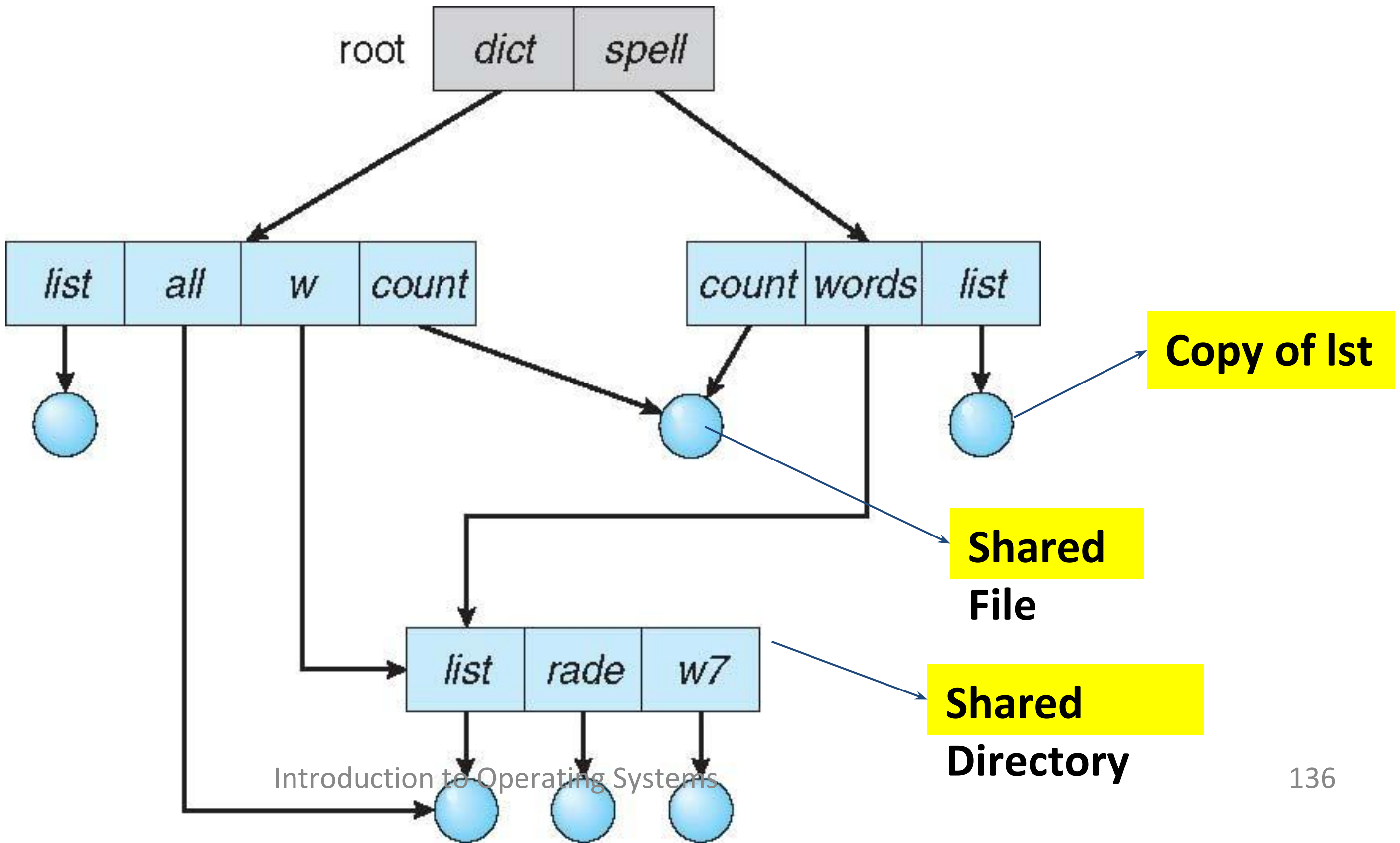
- Concepts to be used:
  - Current directory
  - Relative path
  - Absolute path
  - Operations on directory
- E.g. MS-DOS

•Generalization of two Level Directory
•User can create subdirectory
•Searching : absolute/relative path /
current directory
•Delete:
MS-DOS : Empty directory
UNIX: rm command
•Sharing : Not possible

- **Advantages:**
- Very general, since full pathname can be given.
- Very scalable, the probability of name collision is less.
- Searching becomes very easy, we can use both absolute paths as well as relative.
- **Disadvantages:**
- Every file does not fit into the hierarchical model, files may be saved into multiple directories.
- We can not share files.
- It is inefficient, because accessing a file may go under multiple directories.

Acyclic-Graph Directories

- Graph without cycle
- E.g. users working on joint project require common files
- Common files or subdirectories can be **shared**.
  - Changes made by one user visible to other users.
- E.g.Unix



Acyclic-Graph Directories implementation of shared files and subdirectories

- Duplicate entries in both files. (consistency problem)
- Create a separate directory. Marked it as link. Link include absolute path of a file.

Advantages & disadvantages

- Advantage:
  - More flexible than tree structured directory.
- Disadvantage:
  - Complex approach
    - Multiple paths can be used to access a file.
  - Aliasing problem :
  - Backup / Searching:
  - shared files should be traversed once. E.g. backup file system
- File deletion:
  - Removal of a file keeps dangling pointers. (Unix)
  - **Or keep reference file / counter.**
      - » **Delete file when reference file is empty or counter is zero.**

Acyclic-Graph Directories

- An acyclic graph is a graph with no cycle and allows us to share subdirectories and files.
- The same file or subdirectories may be in two different directories.
- It is a natural generalization of the tree-structured directory.
- It is used in the situation like when two programmers are working on a joint project and they need to access files.
- The associated files are stored in a subdirectory, separating them from other projects and files of other programmers since they are working on a joint project so they want the subdirectories to be into their own directories.
- The common subdirectories should be shared. So here we use Acyclic directories.
- It is the point to note that the shared file is not the same as the copy file.
- If any programmer makes some changes in the subdirectory it will reflect in both subdirectories.

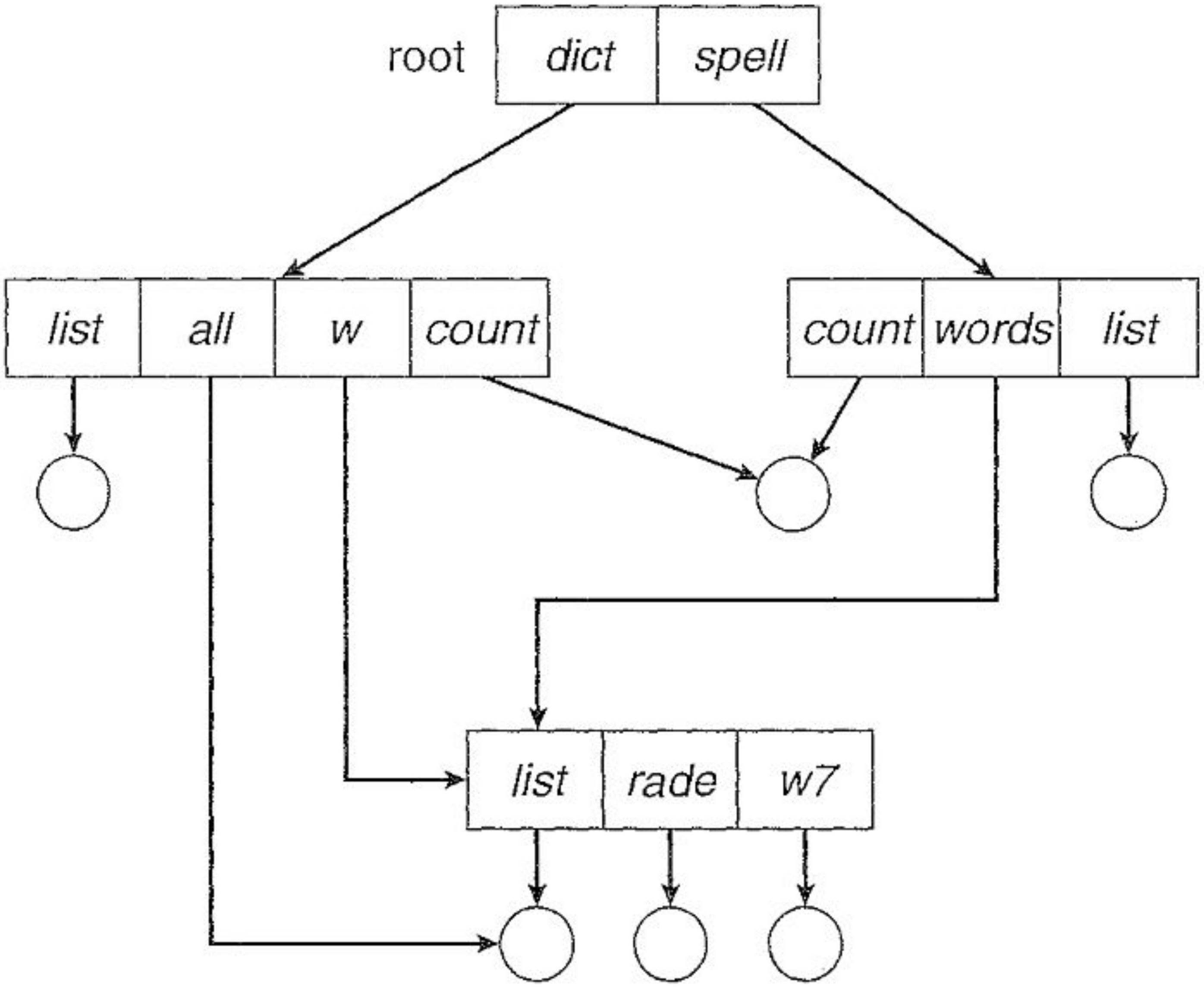
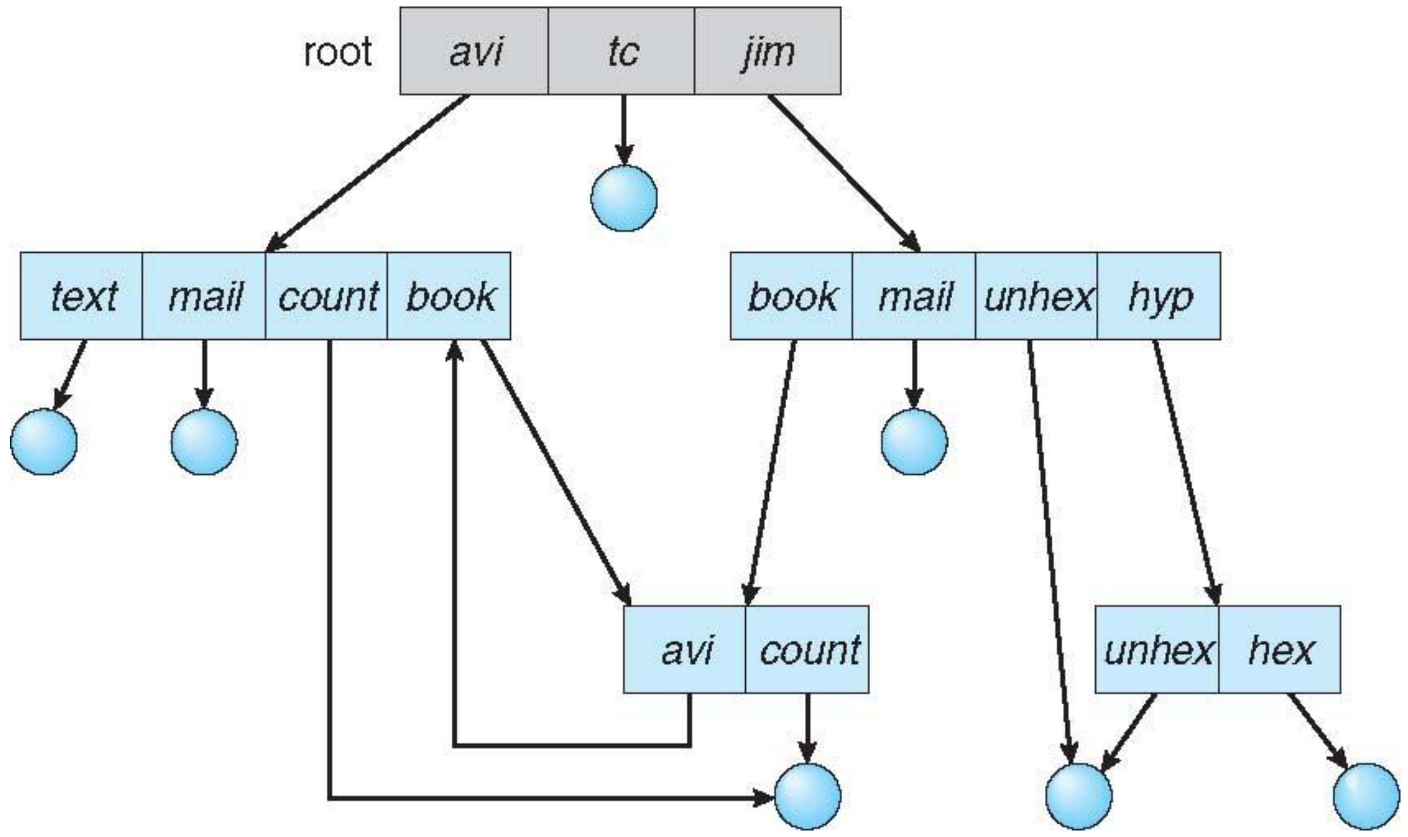


Figure 10.11 Acyclic-graph directory structure.

- **Advantages:**
- We can share files.
- Searching is easy due to different-different paths.
- **Disadvantages:**
- We share the files via linking, in case deleting it may create the problem,
- If the link is a soft link then after deleting the file we left with a dangling pointer.
- In the case of a hard link, to delete a file we have to delete all the references associated with it.

General Graph Directory



Self reference creates cycle

General Graph Directory problems

- In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory. The main problem with this kind of directory structure is to calculate the total size or space that has been taken by the files and directories.
- Searching:
  - Poorly designed algorithm might result in infinite loop, continually searching through cycle and never terminating.
- Garbage collection is used if file is deleted

General Graph Directory problems

- Deletion:
  - reference count may not be zero even it is not possible to refer a file.
    - Use garbage collection scheme that maintain reference list and determine when last reference has been deleted and space can be reallocated
- Link Addition:
  - Need of cycle detection algorithm which is expensive

- **Advantages:**
- It allows cycles.
- It is more flexible than other directories structure.
- **Disadvantages:**
- It is more costly than others.
- It needs garbage collection.

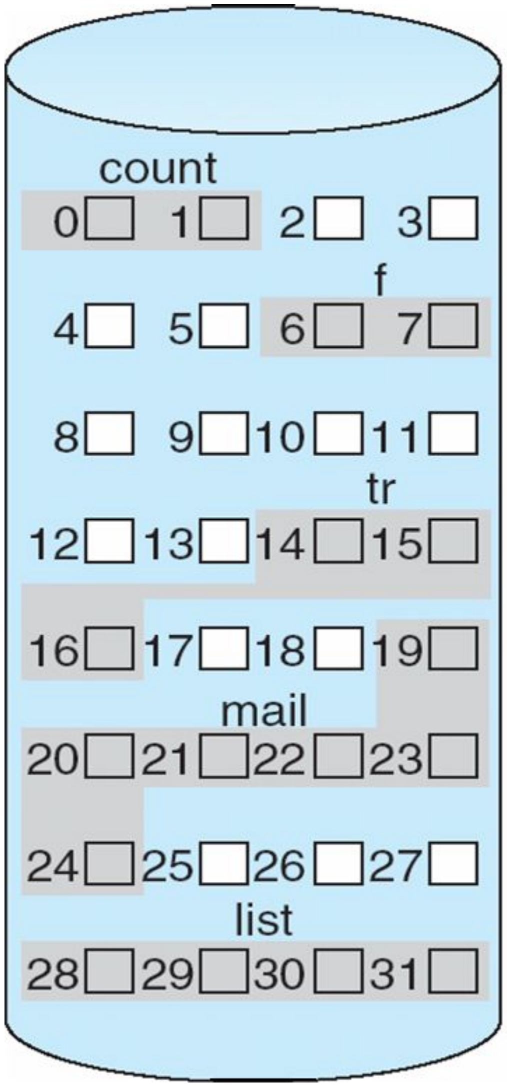
Mass-Storage Management

- 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
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling

Allocation Methods Introduction

- Files are stored on storage device.
- Main issue is how to allocate space to a file so that there is **effective space utilization**
- Three space allocation methods exists:
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation

Contiguous Allocation of Disk Space



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Contiguous Allocation

- each file occupies set of contiguous blocks, one after other
- Uses continuous address space on disk
  - Blocks are allocated as b, b+1, b+2,…….
- **Advantages**
  - **Can be used for sequential and direct access**
  - **Simple to implement** –
    - only starting location (block #) and length (number of blocks) of a file is in a directory
  - **Read performance is great.**
    - Minimum disk head movement
    - Accessing file is easy
    - Only need one seek to locate the first block in the file. The rest is easy.

Contiguous Allocation Problems

- Finding space for a new file
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.
  - **Dynamic storage-allocation problem**
    - How to satisfy the request of size n from the list of free holes for a file
      - » First fit strategy: Faster
      - » Best fit strategy: minimum space wastage
      - » Worst fit strategy : time & space complexity is more
  - **External fragmentation**
      - » When files are deleted, free space is broken into chunks.
      - » It becomes problem when largest contiguous chunk is insufficient to handle the request.

Contiguous Allocation Problems

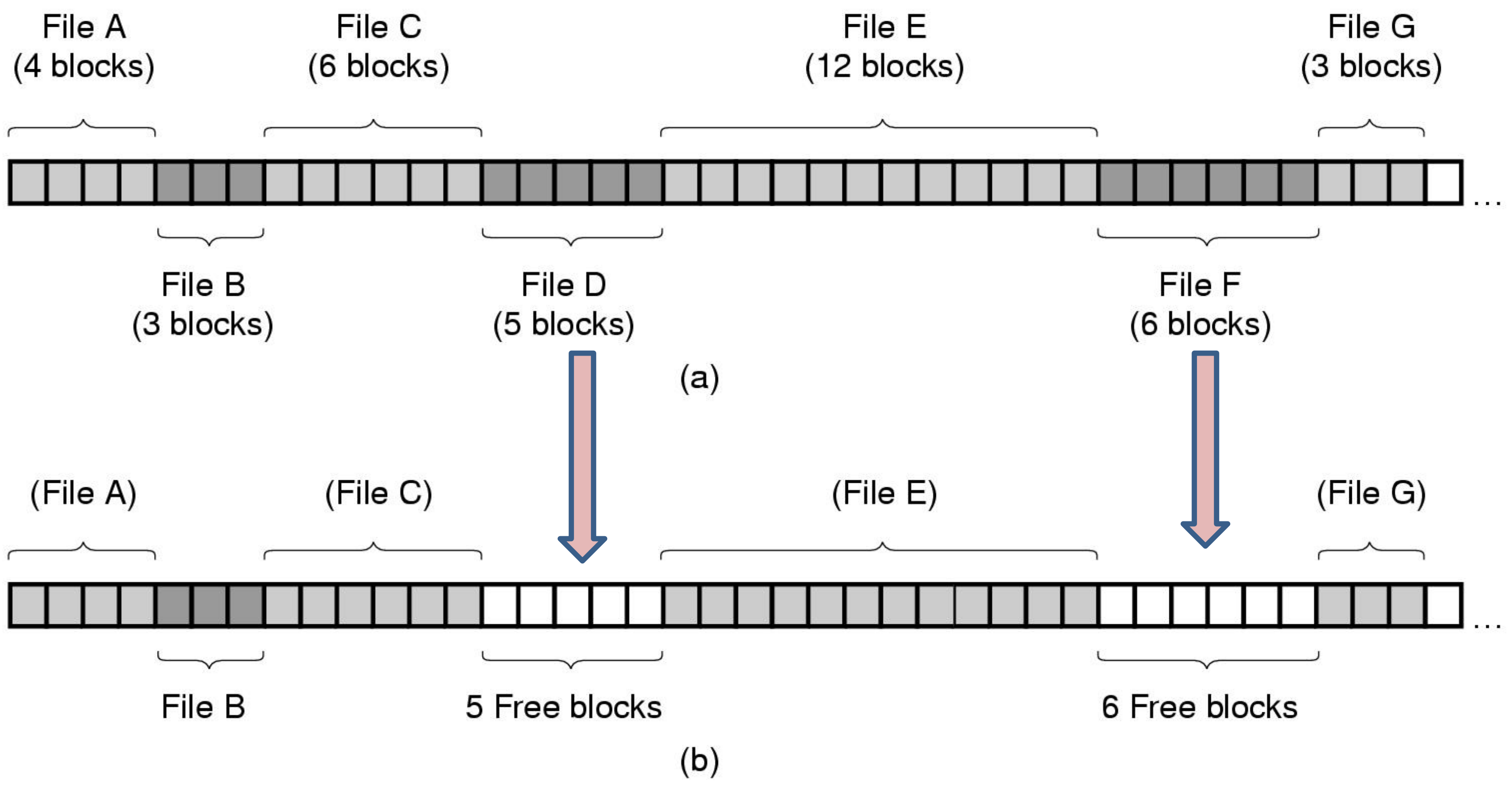
- **Strategies to avoid space loss due to fragmentation**
  - Need for **compaction routine**
    - Initially copy entire file system on other disk, and then copy the files back contiguously
    - **Can be done during off-hours**
    - **Problem of downtime (not efficient for production systems)**

    - or can use **on-line defragmentation (degrades performance)**

Contiguous Allocation Problems

- Determining space requirement
  - Overestimate
    - Internal fragmentation in slowly growing file (space wastage)
    - Even though file size is known advance, pre- allocation is inefficient, if file grows slowly.
  - Underestimate
    - Terminate and restart
    - Find & Copy it in a larger hole
  - Allocate new contiguous space in the form of **Extent.**
    - Initial, next extent allocation,
    - Maintain location & pointer to extent

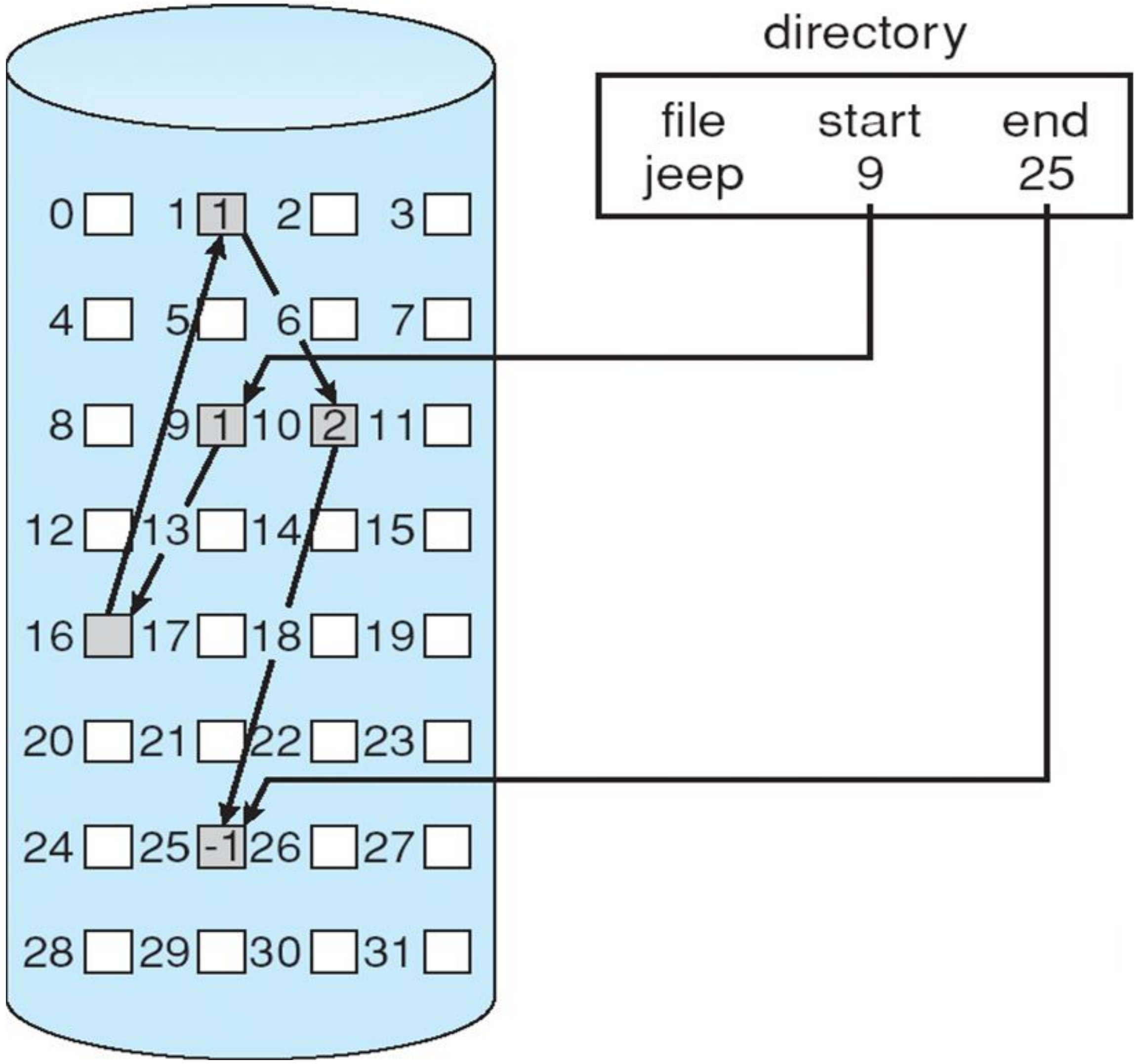
Contiguous Allocation: external fragmentation problem



(a) Contiguous allocation of disk space for 7 files.

(b) The state of the disk after files D and F have been removed.

Linked Allocation



Each file is linked list of disk blocks

Linked Allocation

- It is non contiguous.
- file is divided in blocks in physical memory but non contiguously.
- Each file is a linked list of disk blocks
- Directory contains pointer to first and last block.
- Each file block contain pointer to next block

- File Operations
- File creation : Free blocks are arranged from the free list.
- File read: Each block contain pointer to next block.
  - Follow pointers

Linked Allocation

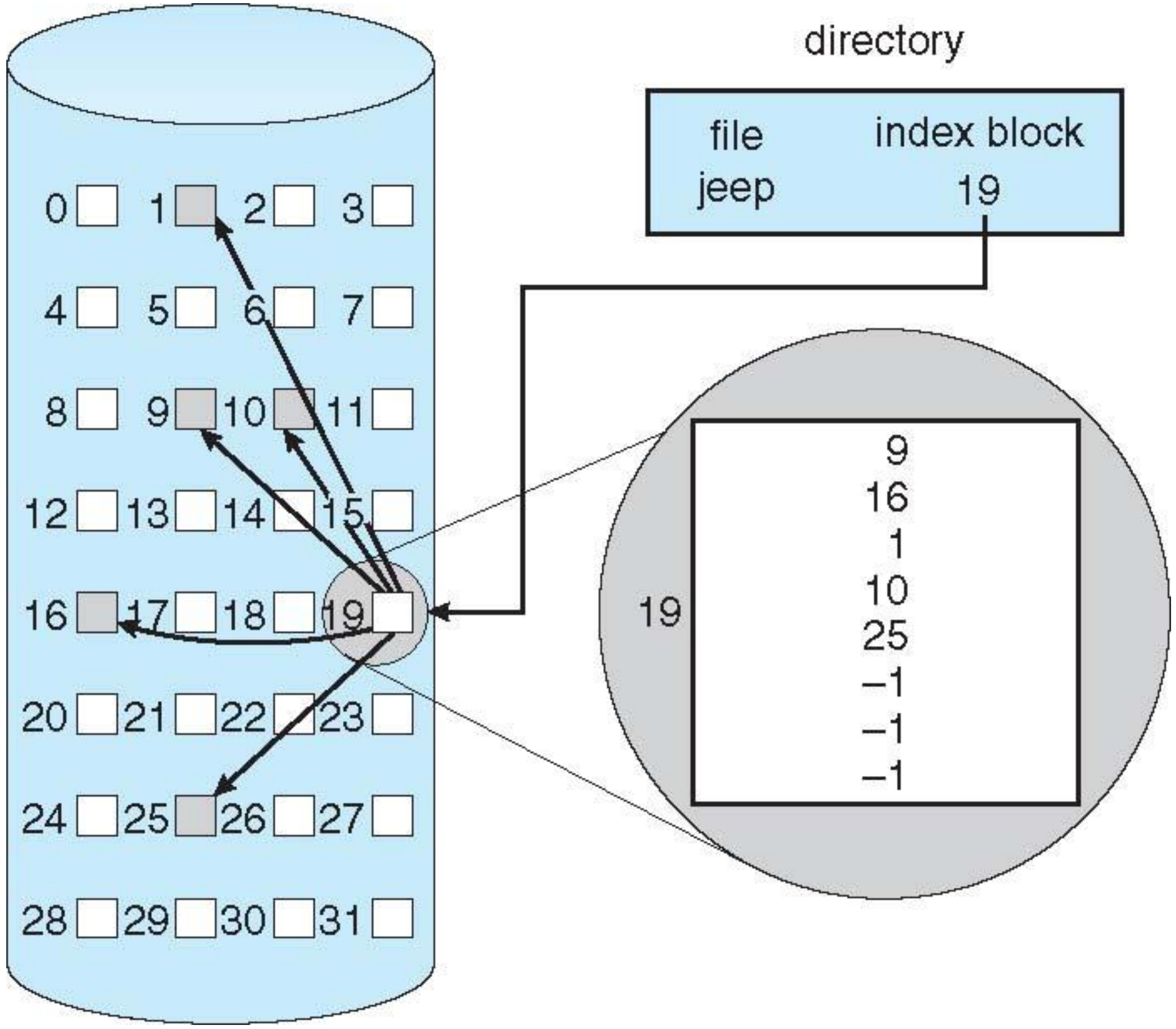
Advantages

- Dynamic storage allocation
- no External fragmentation
- File size can increase

Disadvantages

- large seek time
- Effective only for sequential access
- random access or direct access is difficult
- Space wastage for pointers /overhead of pointers
  - If block size 512KB, pointer 4KB, Effective block size 508KB
  - Solution: Use cluster of blocks that improves disk access time and decreases space need for pointers
  - Clusters may create problem of internal fragmentation
- Reliability
- Lost/damaged pointer
  - Solution use of doubly linked list

Example of Indexed Allocation



Allocation Methods - Indexed

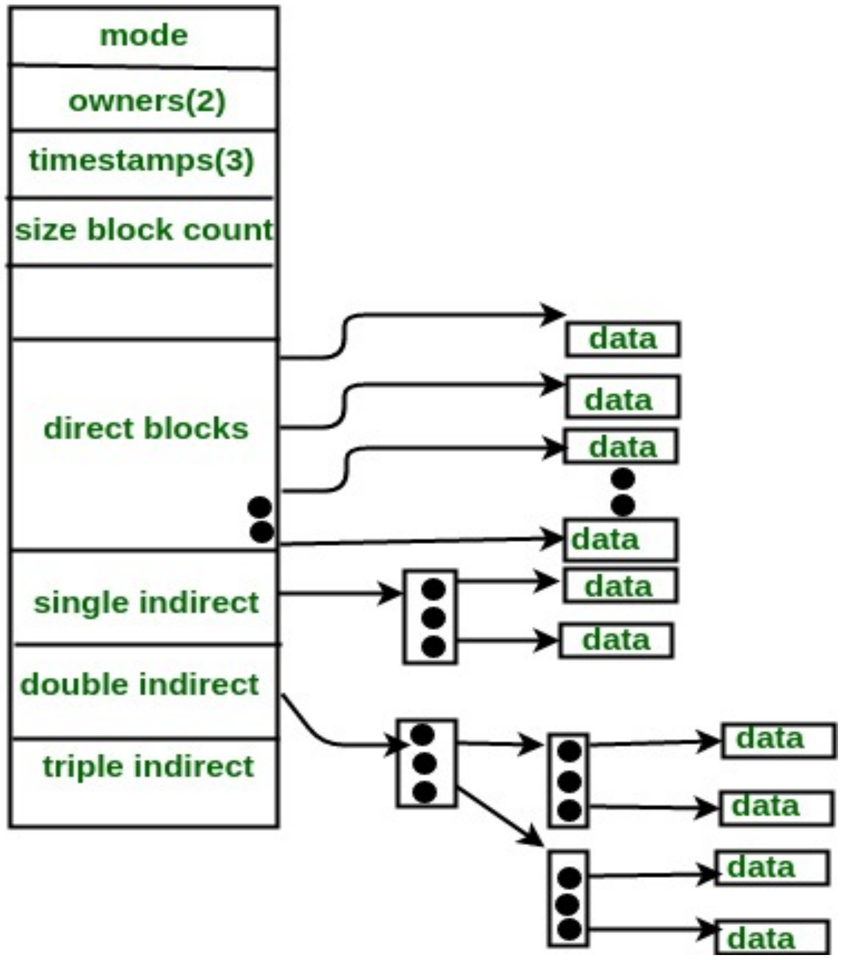
- In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block .

- **Advantages**
  - Efficient random access
  - No external fragmentation

- **Disadvantage**
  - Suitable for only direct access
  - Pointer overhead more than pointer overhead of linked allocation
  - Space overhead to maintain index block

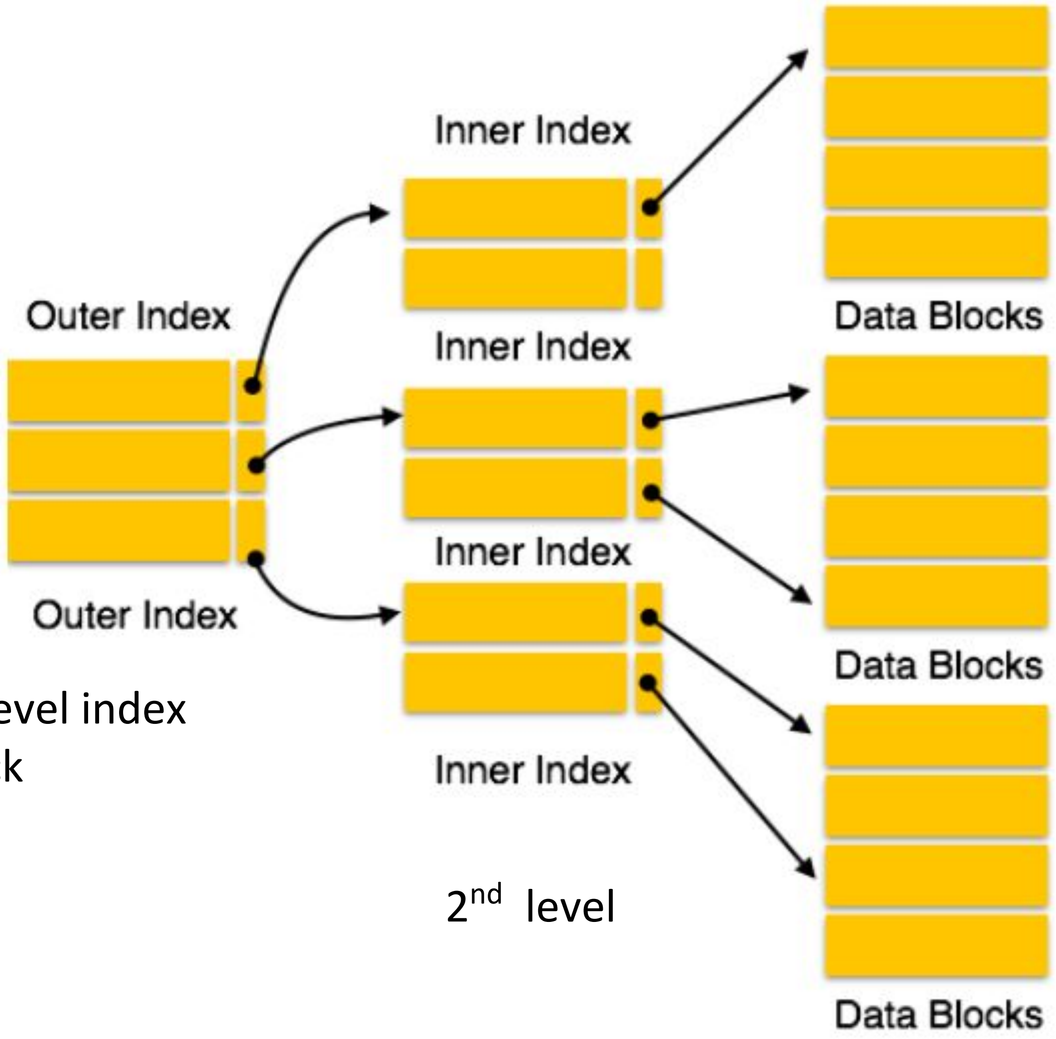
Allocation Methods - Indexed

- For files that are very large, single index block may not be able to hold all the pointers.
- Following mechanisms can be used to resolve this:
- Linked scheme: This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block.
- Multilevel index: In this policy, a first level index block is used to point to the second level index blocks which inturn points to the disk blocks occupied by the file. This can be extended to 3 or more levels depending on the maximum file size.
- Combined Scheme: In this scheme, a special block called the Inode (information Node) contains all the information about the file such as the name, size, authority, etc and the remaining space of Inode is used to store the Disk Block addresses which contain the actual file as shown in the image below. The first few of these pointers in Inode point to the direct blocks i.e the pointers contain the addresses of the disk blocks that contain data of the file. The next few pointers point to indirect blocks. Indirect blocks may be single indirect, double indirect or triple indirect. Single Indirect block is the disk block that does not contain the file data



Combined Scheme:

Multilevel Index



1st level index block

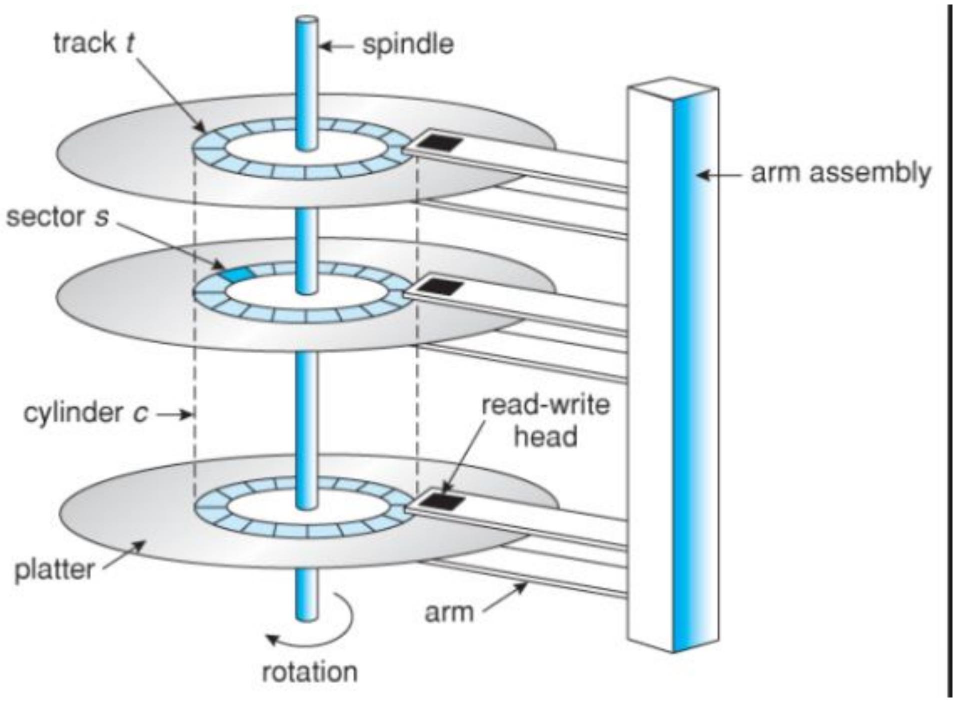
2nd level

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.
- Disk scheduling is important because:
- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller.
- Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.
- Seek Time:Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- Rotational Latency: Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- Transfer Time: Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- Disk Access Time: Disk Access Time is:
- Disk Access Time = Seek Time +
- Rotational Latency +
- Transfer Time

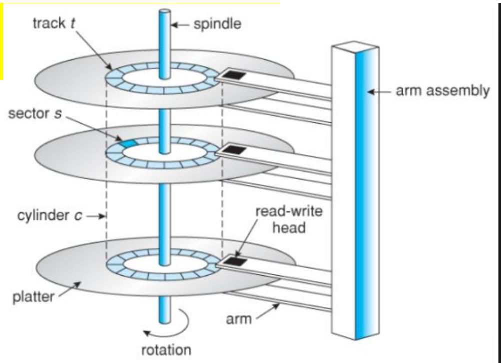
Disk Scheduling

- Disk Response Time: Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of the all requests. Variance Response Time is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.



Disk Scheduling

- **Goal of OS:** Efficient use of resources is the main job of operating system.
- For i/o operation, these goals can be achieved by **improving access time and disk bandwidth**
- For i/o operation system must know type of operation (input / output), disk address, memory address, number of block transfer
- Whenever process request for i/o operation whether to service immediately or it can be pending request depends on availability of i/o controller and disk drive.
- How to service depends on various disk scheduling algorithm.
- The process of picking ready job from disk queue is scheduling.



Disk scheduling algorithms

- First-Come, First-Served(FCFS)
- Shortest-Seek-Time-First (SSTF)
- SCAN Scheduling
- C-Scan Scheduling
- Look Scheduling

First-Come, First-Served(FCFS)

- FCFS stands for First-Come-First-Serve.
- It is the simplest form of disk scheduling. It is a very easy CPU scheduling algorithm.
- (FCFS) algorithm is intrinsically fair, but it generally does not provide the fastest service.
- It is an OS disk scheduling algorithm that runs the queued requests and processes in the way that they arrive in the disk queue.
- In this scheduling algorithm, the process which requests the processor first receives the processor allocation first.
- It is managed with a FIFO queue.
- In FCFS, each process eventually has a chance to execute, therefore there is no starvation.
- **Disadvantages:**
- It is not very efficient because of its simplicity.
- Its average waiting time is high.
- It is a Non-Preemptive CPU Scheduling Algorithm, which implies that once a process has been assigned to a CPU, it would never release the CPU until the process has completed executing.

Problem

- Suppose that a disk drive has 200 cylinders, numbered from 0 to 199. The drive is currently serving a request at cylinder 53. the queue of pending requests, in FIFO order, is 98, 183, 37, 122, 14, 124, 65, 67

  Starting from current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

  a. FCFS
  b. SSTF
  c. SCAN
  d. LOOK
  e. C-SCAN
  f. C-LOOK

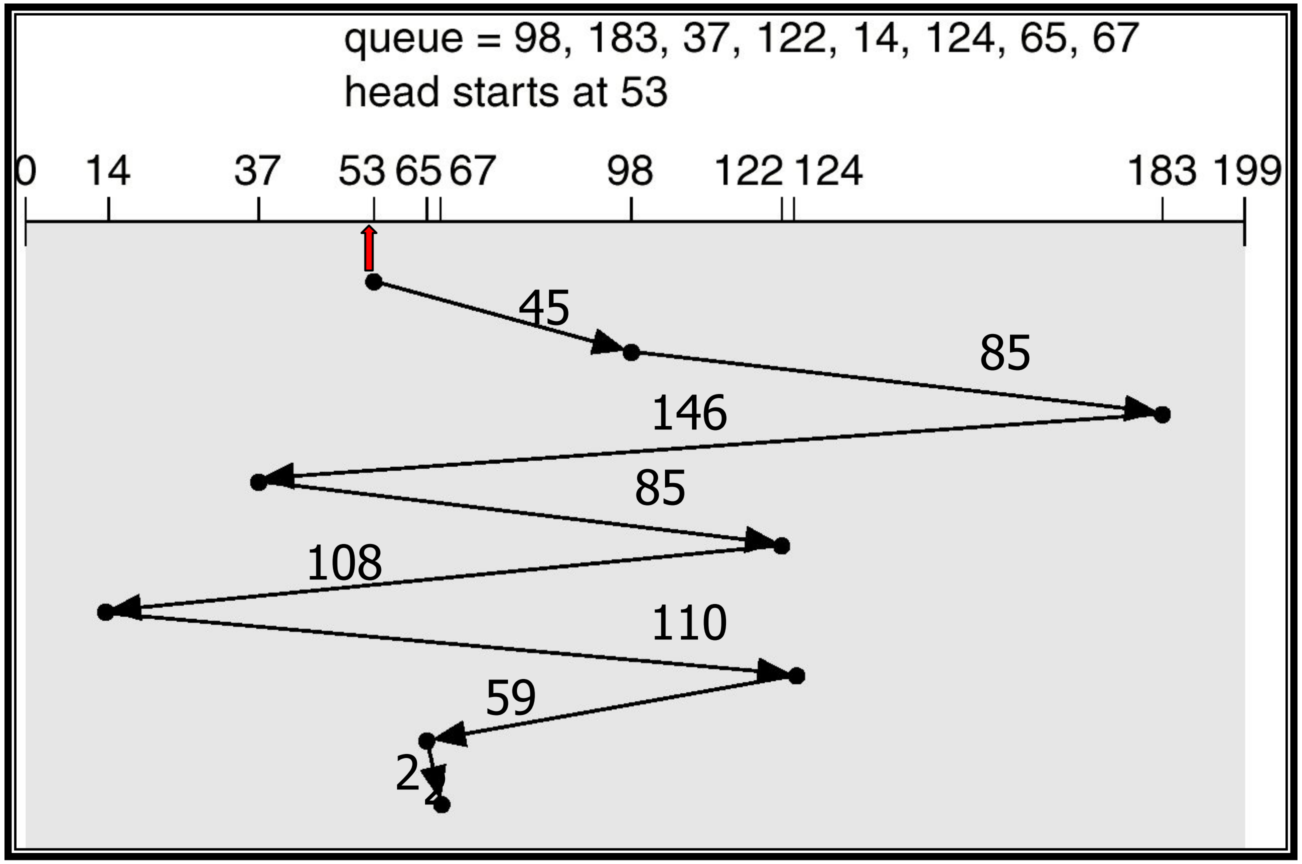
FCFS

- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

  98, 183, 37, 122, 14, 124, 65, 67

  Head pointer 53
- Consider, for example, a disk queue with requests for I/0 to blocks on cylinders in that order.
- If the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183, 37, 122, 14, 124, 65, and finally to 67,
- for a total head movement of 640 cylinders.
- This schedule is diagrammed in Figure 12.4.

FCFS



(98-53)+(183-98)+(183-37)+(122-37)+(122-14)+(124-14)+(124-65)+(67-65)

Total head movement=45+85+146+85+108+110+59+2= 640 cylinders

Shortest-Seek-Time-First(SSTF)

- It is one form SJF scheduling.
- It is reasonable to service all requests close to current head position than moving the head far away to service other requests.
- This assumption is used in SSTF.
- Efficiency of I/O can be increased, if request requiring minimum seek time is serviced first.
- **SSTF chooses the pending request closest to current head position (least seek time).**

Problem

- Suppose that a disk drive has 200 cylinders, numbered from 0 to 199. The drive is currently serving a request at cylinder 53. the queue of pending requests, in FIFO order, is 98, 183, 37, 122, 14, 124, 65, 67

  Starting from current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

  a. FCFS
  b. SSTF
  c. SCAN
  d. LOOK
  e. C-SCAN
  f. C-LOOK

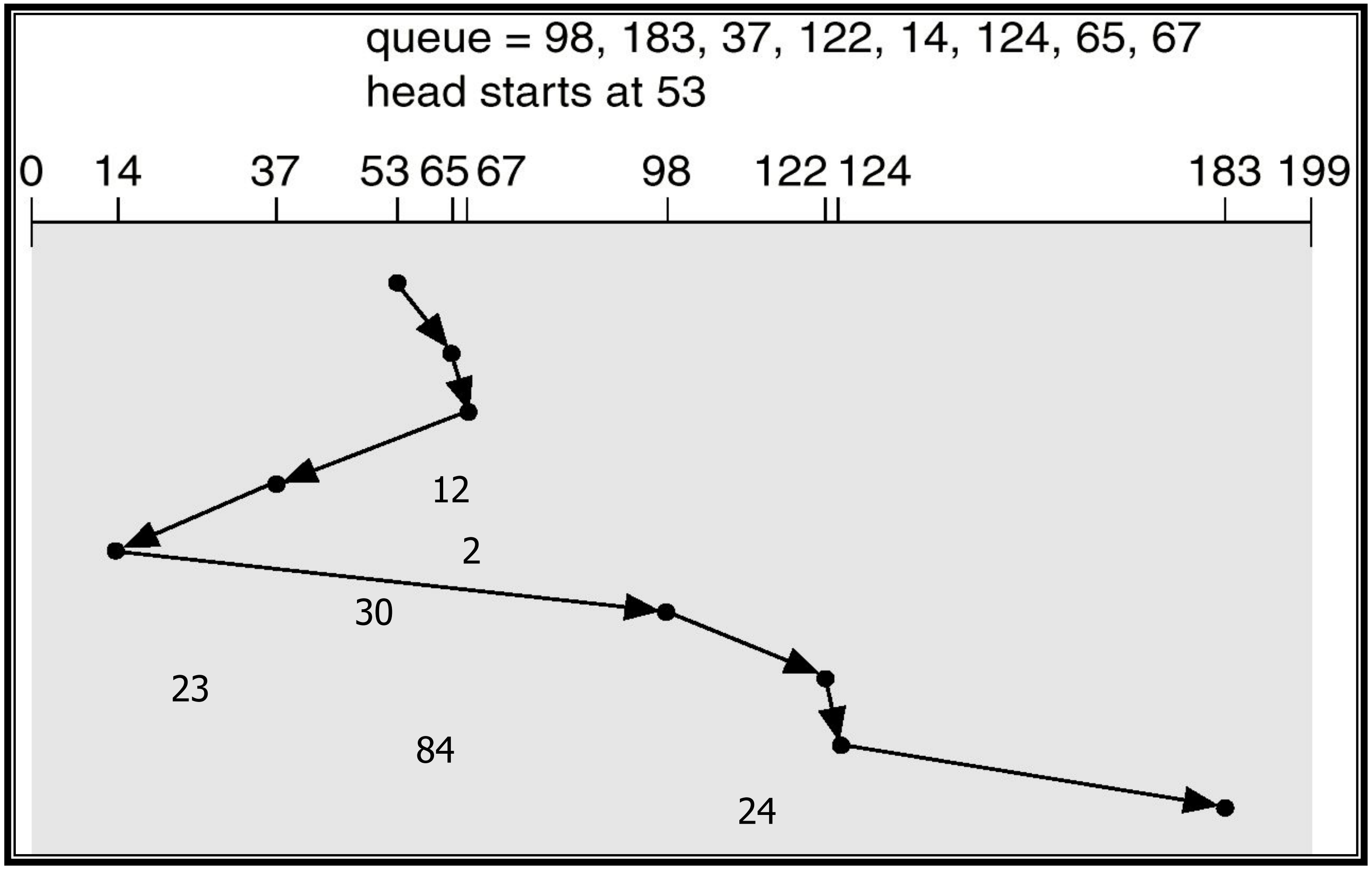
SSTF

- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

  98, 183, 37, 122, 14, 124, 65, 67

- Head pointer 53
- For our example request queue, the closest request to the initial head position (53) is at cylinder 65.
- Once we are at cylinder 65, the next closest request is at cylinder 67.
- From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next.
- Continuing, we service the request at cylinder 14,then 98, 122, 124, and finally 183.
- This scheduling method results in a total head movement of only 236 cylinders-little more than one-third of the distance needed for FCFS scheduling of this request queue.
- Clearly, this algorithm gives a substantial improvement in performance.

SSTF (Cont.)



2

(65-53)+(67-65)+(67-37)+(37-14)+(98-14)+(122-98)+(124-122)+(183-124)

Total head movement=12+2+30+23+84+24+2+59=236 cylinders

SSTF scheduling

- Advantages:
  - Substantial improvement over FCFS.
  - Increase efficiency of the system.
  - It improves and increase the throughput.
  - SSTF's total seek time is lower than the FCFS.
  - It has less response time and average waiting time.

- Disadvantages
  - **Starvation**
  - Continual stream of requests near one another could cause to wait for far pending request .
  - Although the SSTF algorithm is a substantial improvement over the FCFS Algorithm, it is not optimal.
  - In the SSTF disk scheduling algorithm, the high variance is available in waiting time and response time.
  - The algorithm is slowed by frequent changes in the head's direction.

Comparison between the FCFS and SSTF Disk Scheduling Algorithm

FCFS Disk Scheduling Algorithm	SSTF Disk Scheduling Algorithm
FCFS stands for First Come First Serve.	SSTF stands for Shortest Seek Time First.
It is not effective in seek movements.	It is very effective in seek movements.
It increases the total seek time than the SSTF.	It reduces the total seek time than the FCFS.
It gives more response time and average waiting time.	It gives less response time and average waiting time.
It doesn't cause starvation to any request.	The request which is far from the head will suffer starvation in the SSTF algorithm.
Its head direction doesn't matter, and the head moves in both forward or reversed directions.	Its head direction plays an important role in breaking a tie between requests.

SCAN / Elevator Algorithm

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, direction of the head movement is reversed.
- It is also called ‘Elevator Algorithm’ , since disk arm behaves as elevator in the building.
- In addition to current head position we should know direction of movement.
- Advantages:
- High throughput
- Low variance of response time
- Average response time

- Disadvantages:
- Long waiting time for requests for locations just visited by disk arm

Problem

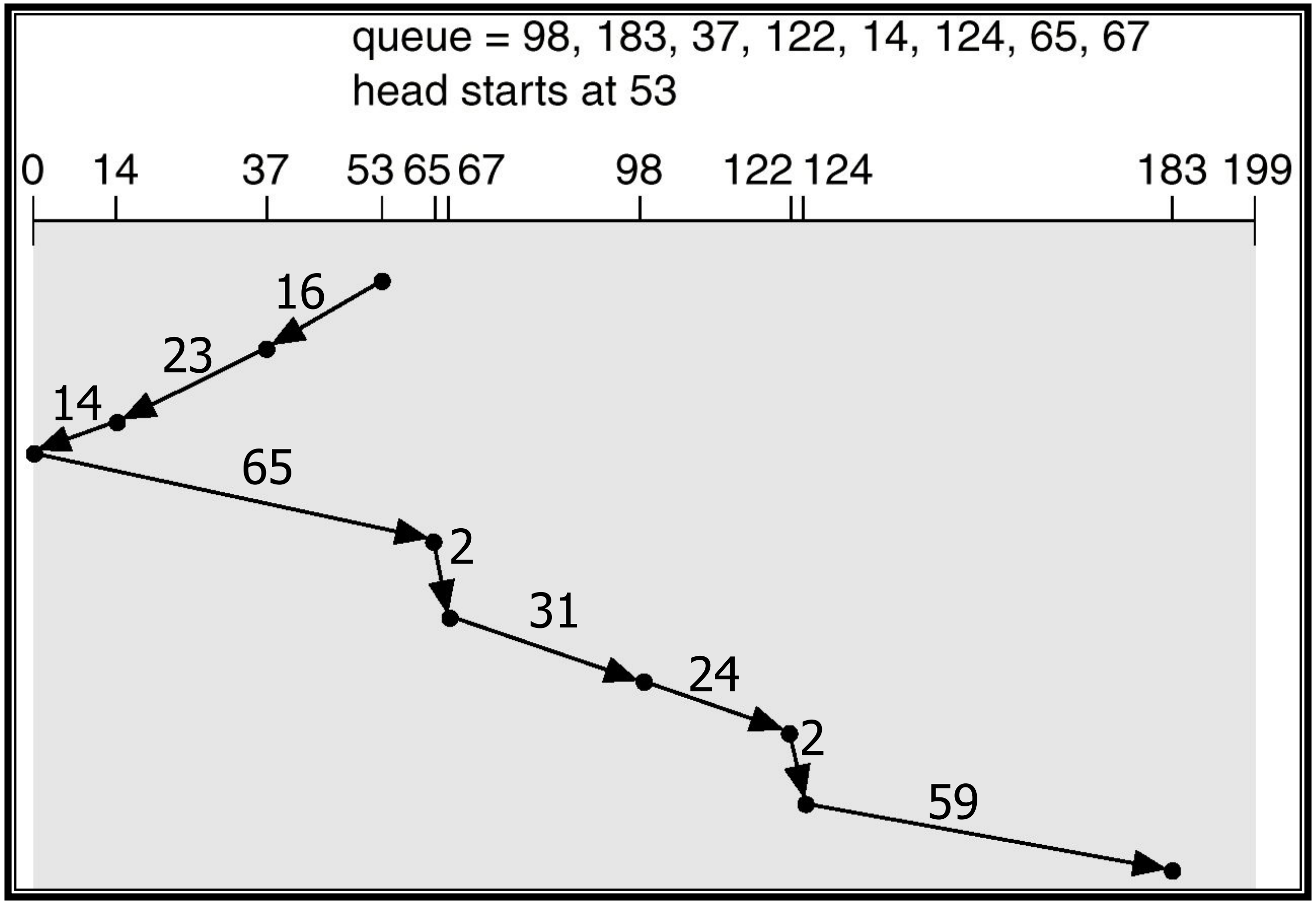
- Suppose that a disk drive has 200 cylinders, numbered from 0 to 199. The drive is currently serving a request at cylinder 53. the queue of pending requests, in FIFO order, is 98, 183, 37, 122, 14, 124, 65, 67

  Starting from current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

  a. FCFS
  b. SSTF
  c. SCAN
  d. LOOK
  e. C-SCAN
  f. C-LOOK

SCAN (Cont.)

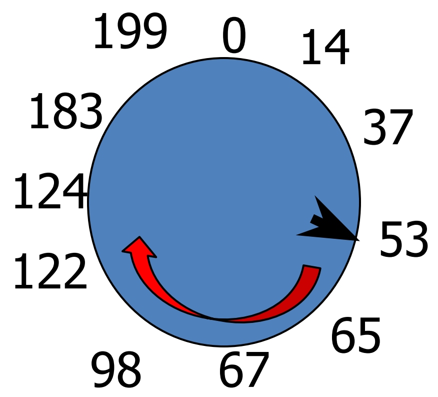
Assume head movement is towards zero.



Total head movement=16+23+14+65+2+31+24+2+59=236 cylinders

C-SCAN

- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.
- Advantages:
- Provides more uniform wait time compared to SCAN



Problem

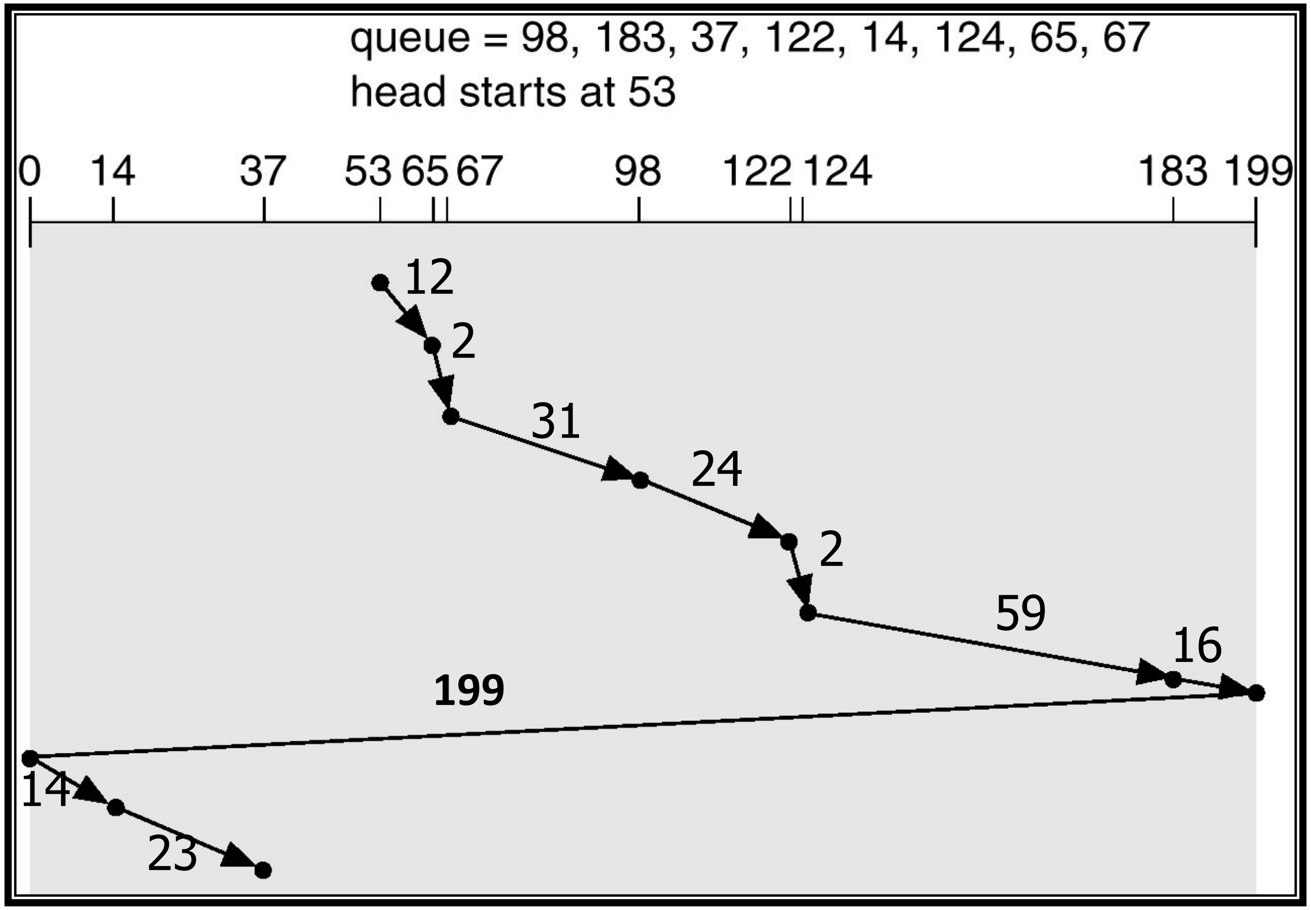
- Suppose that a disk drive has 200 cylinders, numbered from 0 to 199. The drive is currently serving a request at cylinder 53. the queue of pending requests, in FIFO order, is 98, 183, 37, 122, 14, 124, 65, 67

  Starting from current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

  a. FCFS
  b. SSTF
  c. SCAN
  d. LOOK
  e. C-SCAN
  f. C-LOOK

C-SCAN (Cont.)

Head movement towards 199



Total head movement=14+23+12+2+31+24+2+59+16+199=382 cylinders

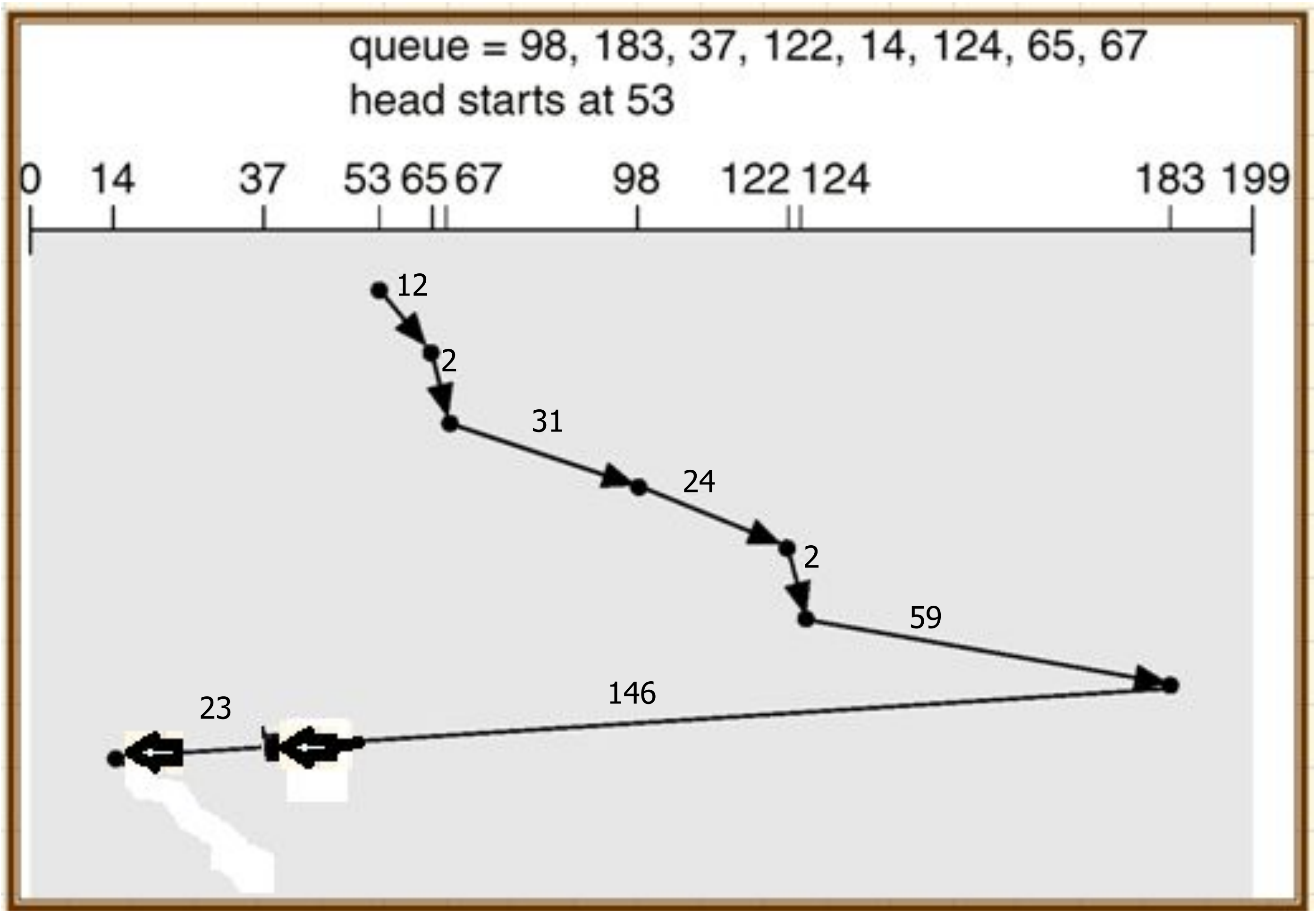
Problem

- Suppose that a disk drive has 200 cylinders, numbered from 0 to 199. The drive is currently serving a request at cylinder 53. the queue of pending requests, in FIFO order, is 98, 183, 37, 122, 14, 124, 65, 67

  Starting from current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

  a. FCFS
  b. SSTF
  c. SCAN
  d. LOOK
  e. C-SCAN
  f. C-LOOK

Look



Total head movement=12+2+31+24+2+59+146+23=299 cylinders

LOOK

It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Problem

- Suppose that a disk drive has 200 cylinders, numbered from 0 to 199. The drive is currently serving a request at cylinder 53. the queue of pending requests, in FIFO order, is 98, 183, 37, 122, 14, 124, 65, 67

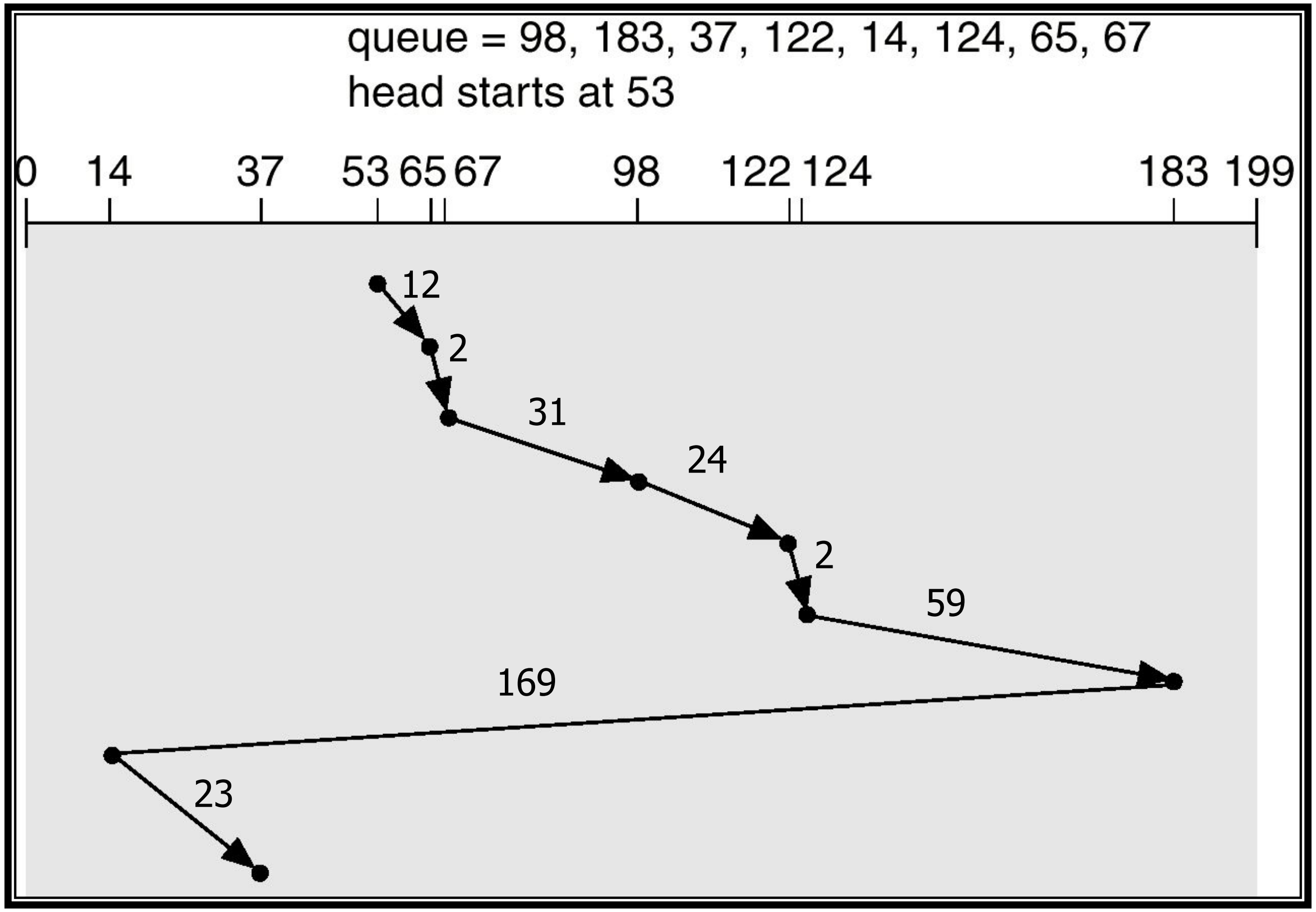
  Starting from current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

  a. FCFS
  b. SSTF
  c. SCAN
  d. LOOK
  e. C-SCAN
  f. C-LOOK

C-LOOK

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
- In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

C-LOOK (Cont.)

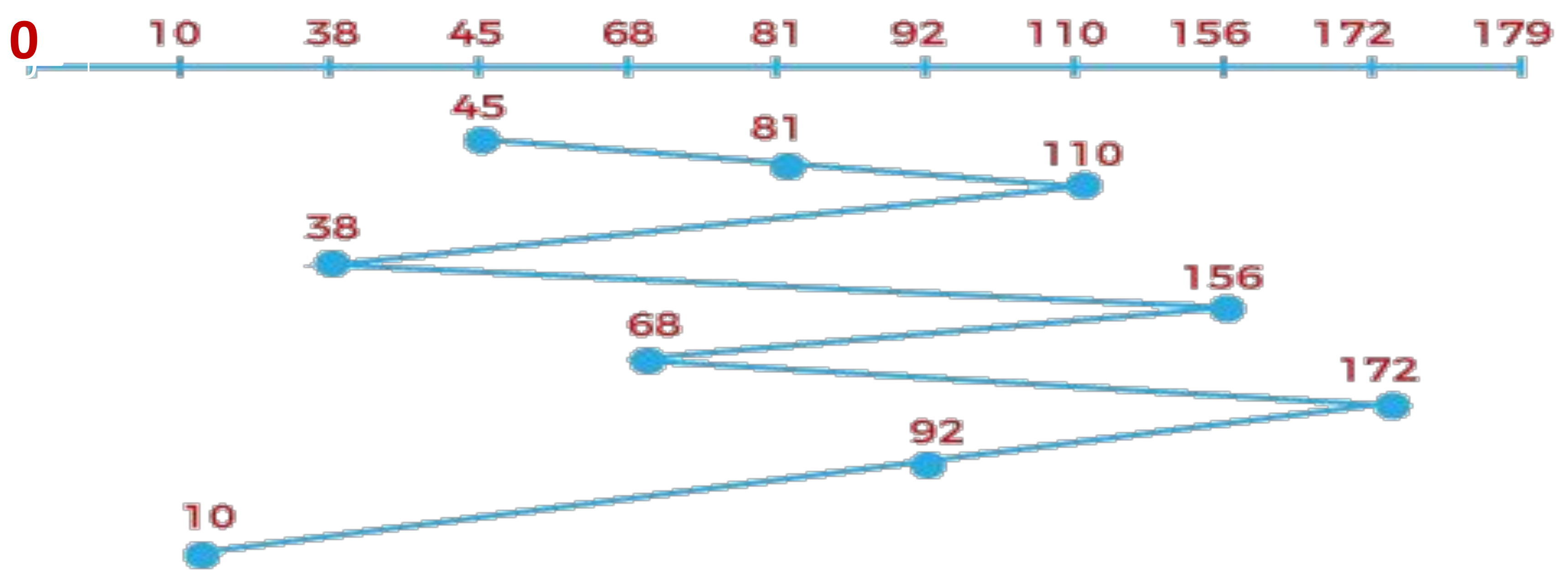


Total head movement=12+2+31+24+2+59+169+23=322 cylinders

Problem1

- Let's take a disk with 180 tracks (0-179) and the disk queue having input/output requests in the following order: 81, 110, 38, 156, 68, 172, 92, 10.
- The initial head position of the Read/Write head is 45.
- Find the total number of track movements of the Read/Write head using the FCFS algorithm.

Solution1

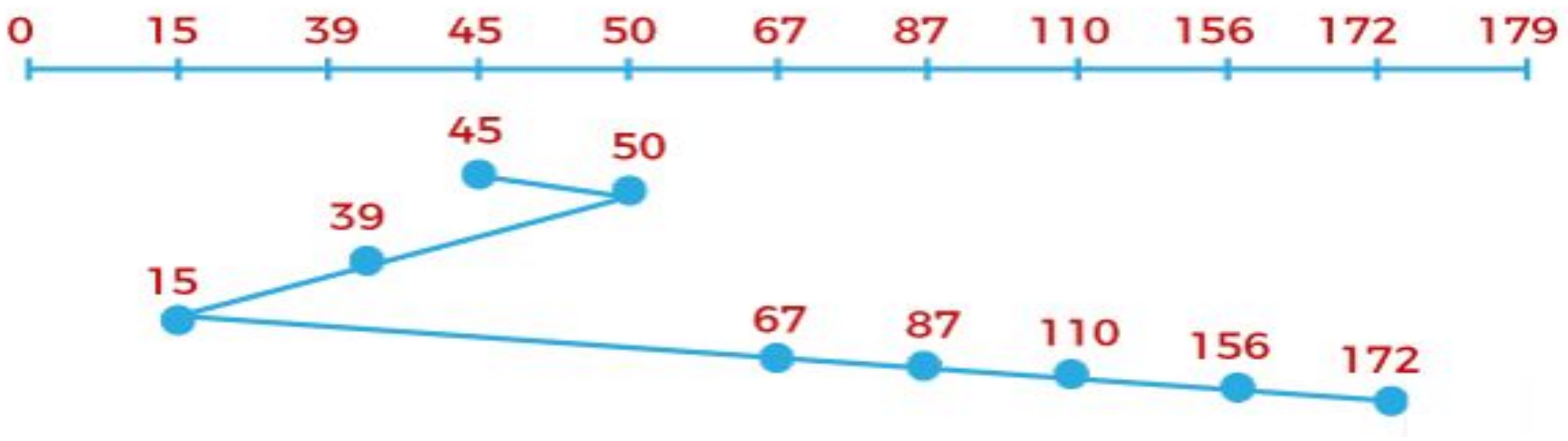


Total head movements
= (81-45) + (110-81) + (110-38) + (156-38) + (156-68) + (172-68) + (172-92) + (92-10)
= 36 + 29 + 72 + 118 + 88 + 104 + 80 + 82
= 609

Problem2

- Let's take a disk with 180 tracks (0-179) and the disk queue having input/output requests in the following order: 87, 110, 50, 172, 67, 156, 39, 15.
- The initial head position of the Read/Write head is 45 and will move in the left-hand side direction.
- Find the total number of track movements of the Read/Write head using the SSTF algorithm.

Solution2



Total head movements
= (50-45) + (50-39) + (39-15) + (67-15) + (87-67) + (110-87) + (156-110) + (172-156)
= 5 + 11 + 24 + 52 + 20 + 23 + 46 + 16
= 197

Problem3

- Find the total number of track movements of the Read/Write head using FCFS algorithm. 87, 160, 40, 140, 36, 72, 66, 15
- In that order, if the disk head is initially at cylinder 55.

Solution3

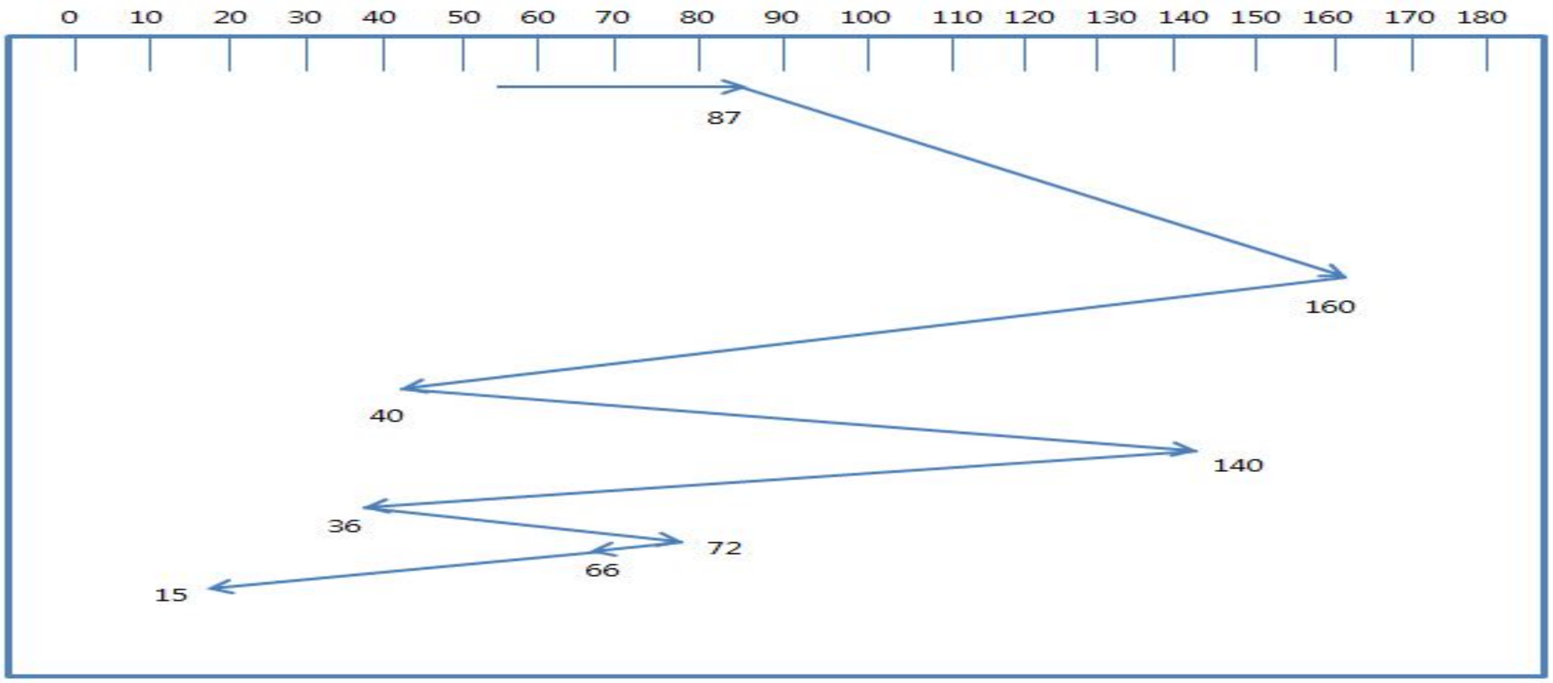


Fig: FCFS Scheduling

Then the total head movement

=(55 to 87)+(87 to 160)+(160 to 40)+(40 to 140)+(140 to 36)+(36 to 72)+(72 to 66)+(66 to 15)

= (32+83+130+110+114+36+6+51)

= 562 cylinders

∴ The average head movements are= 562/8 = 70.25 cylinders

Problem4

- **Find the total number of track movements of the Read/Write head using the SSTF algorithm.**
- **87, 160, 40, 140, 36, 72, 66, 15**
- **In that order, if the disk head is initially at cylinder 60.**

Solution4

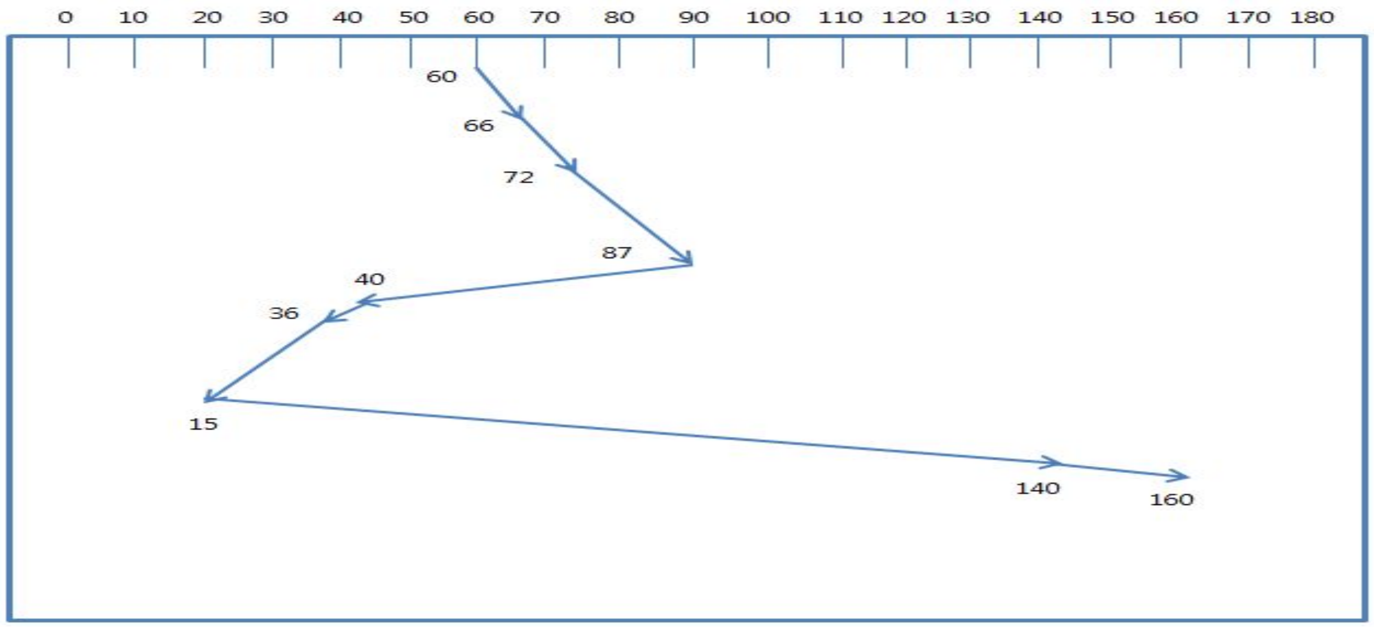
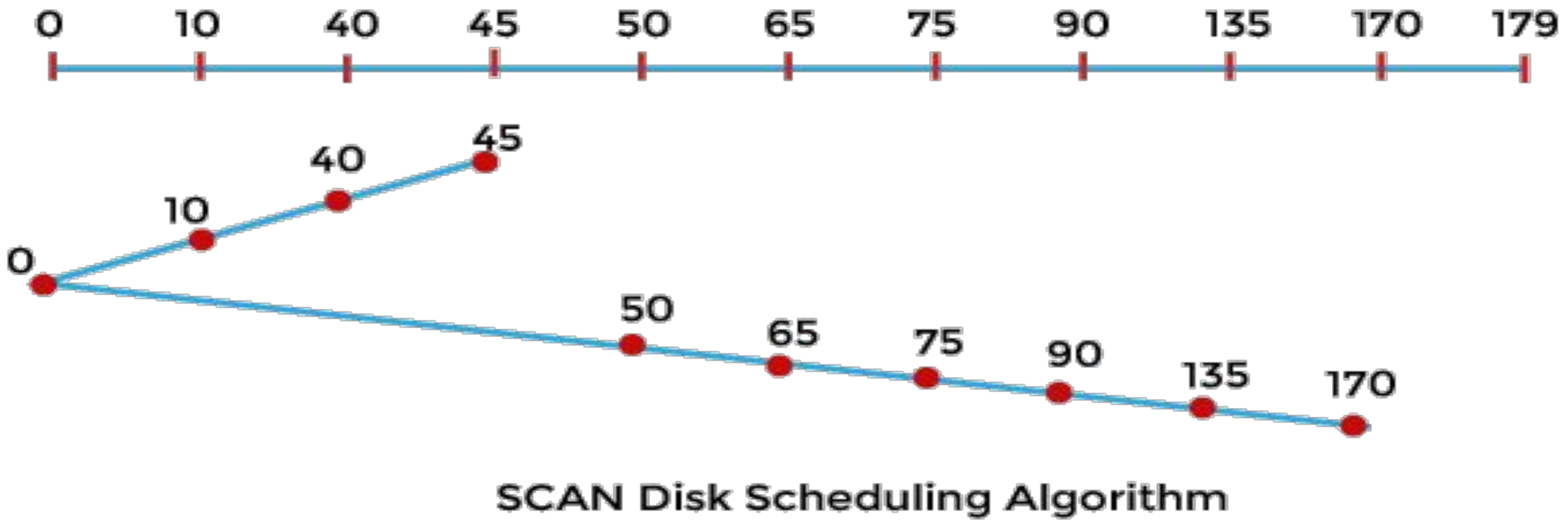


Fig: SSTF Scheduling

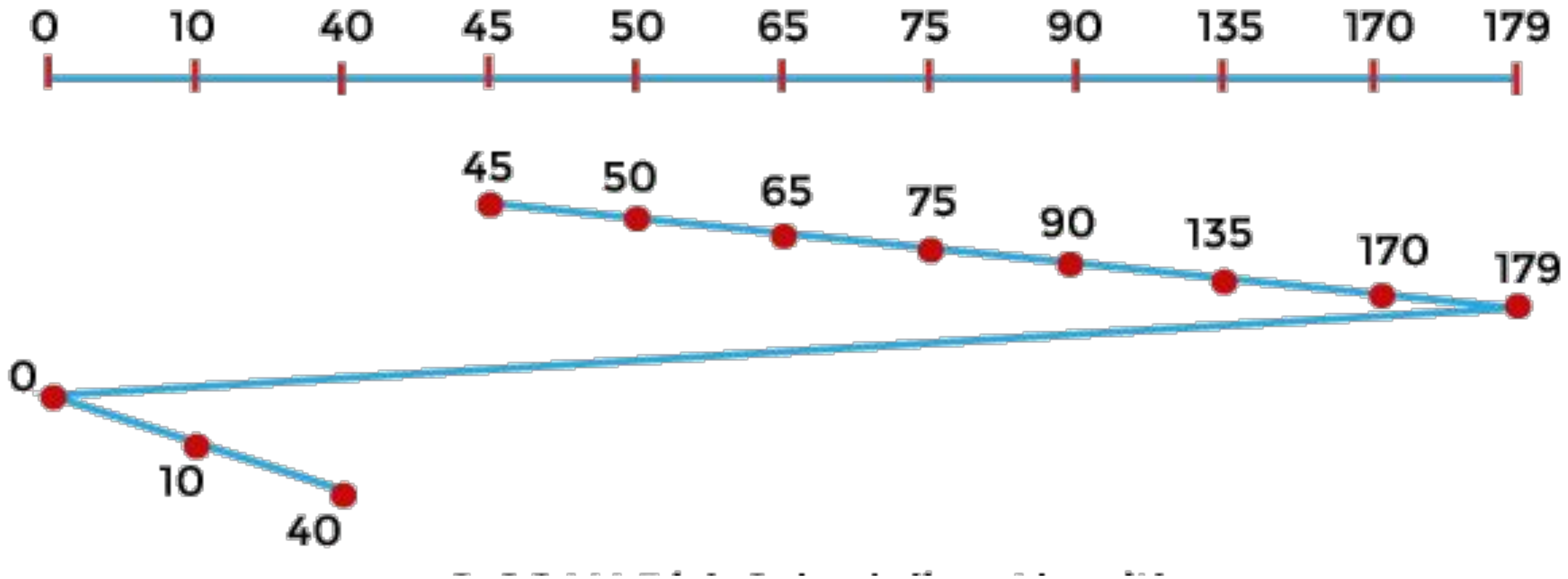
The total head movements of this algorithm
=(60 to 66)+(66 to 72)+(72 to 87)+(87 to 40)+(40 to 36)+(36 to 15)+(15 to 140)+(140 to 160)
= (6+6+15+37+4+21+135+20)
= 244 cylinders
∴ The average head movements are = 244/8 = 30.5 cylinders

Let's take a disk with 180 tracks (0-179) and the disk queue having input/output requests in the following order: 75, 90, 40, 135, 50, 170, 65, 10. The initial head position of the Read/Write head is 45 and will move on the left-hand side. Find the total number of track movements of the Read/Write head using the SCAN algorithm.



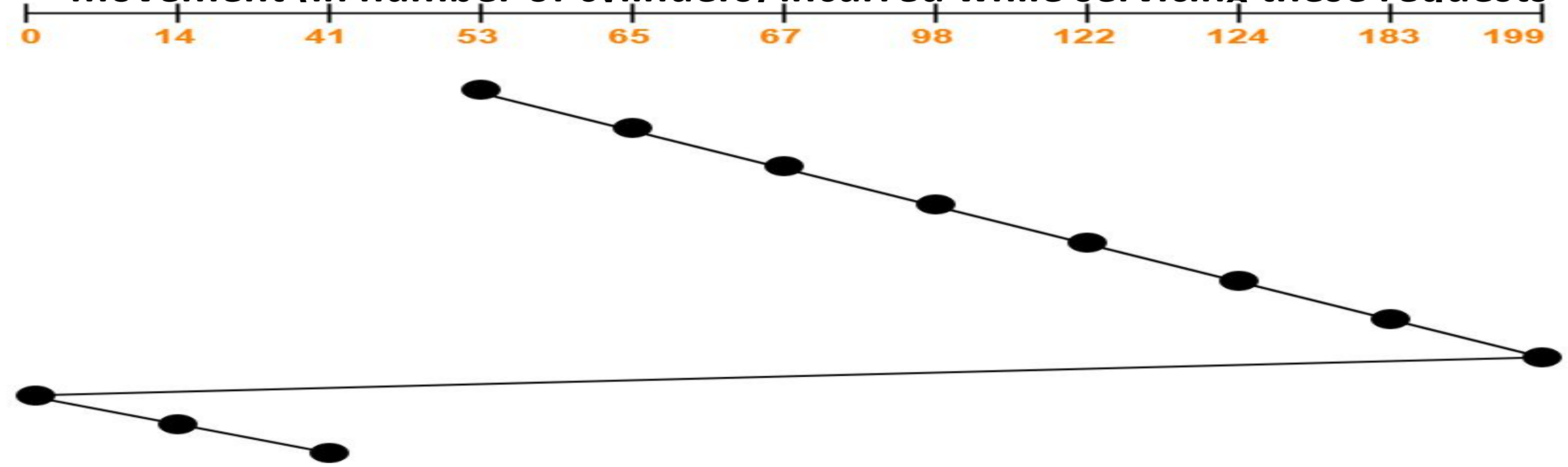
Total head movements
= (45-40) + (40-10) + (10-0) + (50-0) + (65-50) + (75-65) + (90-75) + (135-90) + (170-135)
= 5 + 30 +10 +50 +15 + 10 +15 + 45 + 35
= 215

Let's take a disk with 180 tracks (0-179) and the disk queue having input/output requests in the following order: 75, 90, 40, 135, 50, 170, 65, 10. The initial head position of the Read/Write head is 45 and will move on the right-hand side. Find the total number of track movements of the Read/Write head using the C-SCAN algorithm.



Total head movements
= (50-45) + (65-50) + (75-65) + (90-75) + (135-90) + (170-135) + (179-170) + (179-0) + (10-0) + (40-10)
= 5 + 15 + 10 +15 + 45 + 35 + 9 +179 + 10 + 30
= 353

Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The C-SCAN scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests



Total head movements incurred while servicing these requests

= (65 – 53) + (67 – 65) + (98 – 67) + (122 – 98) + (124 – 122) + (183 – 124) + (199 – 183) + (199 – 0) + (14 – 0) + (41 – 14)

= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 199 + 14 + 27

= 386

Alternatively,

Total head movements incurred while servicing these requests

= (199 – 53) + (199 – 0) + (41 – 0)

= 146 + 199 + 41

= 386

1. Suppose that a disk drive has 200 cylinders, numbered from 0 to 199 and a disk queue with requests for I/O to blocks on cylinders 95, 180, 34,119, 11, 123, 62, 64. Disk head is initially at cylinder 50 and movement direction towards zero.
Compare the performance of SCAN and LOOK in terms of total head movement to satisfy all the pending requests. 4marks

2. Suppose a disk has 201 cylinders, numbered from 0 to 200. Drive is currently serving a request at cylinder 88, and the previous request was at cylinder 50. There is a queue of disk access requests for cylinders in FIFO order is:
30, 85, 90, 100, 105, 110, 135 and 145.
What is the total head movement to satisfy these requests for FCFS, SSTF disk scheduling algorithms? 3marks

3. Suppose that a disk drive has 2000 cylinders, numbered 0 to 1999. The drive is currently serving at cylinder 150, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order is:
68,1280,391,1824,648 ,1120,1570,130
Starting from the current head position, what is the total distance (in cylinder) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithm? 5marks
1) FCFS
2) SSTF
3) SCAN
4) C-LOOK