UNIT 3 – MEMORY MANAGEMENT

Memory management: Functions, single contiguous, Partitioned memory management: multiple relocatable partitioned memory management, paging segmentation, demand paging virtual memory management.

Functions of Memory Management

- The memory management function keeps track of the status of each memory location, either *allocated* or *free*.
- It determines how memory is allocated among competing processes, deciding which gets memory, when they receive it, and how much they are allowed.
- When memory is allocated it determines which memory locations will be assigned.
- It tracks when memory is freed or *unallocated* and updates the status.

Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management
  - **Logical address** – generated by the CPU; also referred to as **virtual address**
  - **Physical address** – address seen by the memory unit
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme
- **Logical address space** is the set of all logical addresses generated by a program
- **Physical address space** is the set of all physical addresses generated by a program

Static vs Dynamic Loading

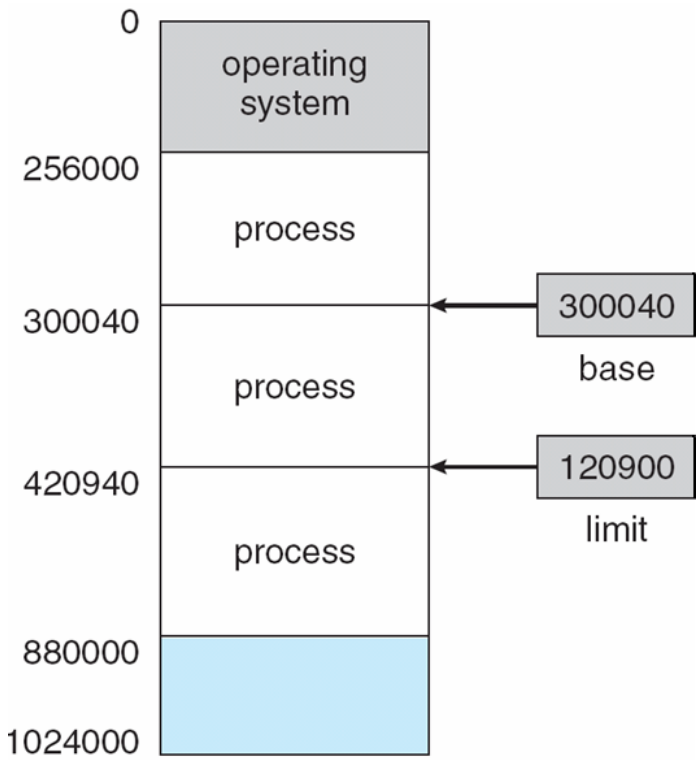
- At the time of loading, with **static loading**, the absolute program (and data) is loaded into memory in order for execution to start.
- If you are using **dynamic loading**, dynamic routines of the library are stored on a disk in relocatable form and are loaded into memory only when they are needed by the program.

Static vs Dynamic Linking

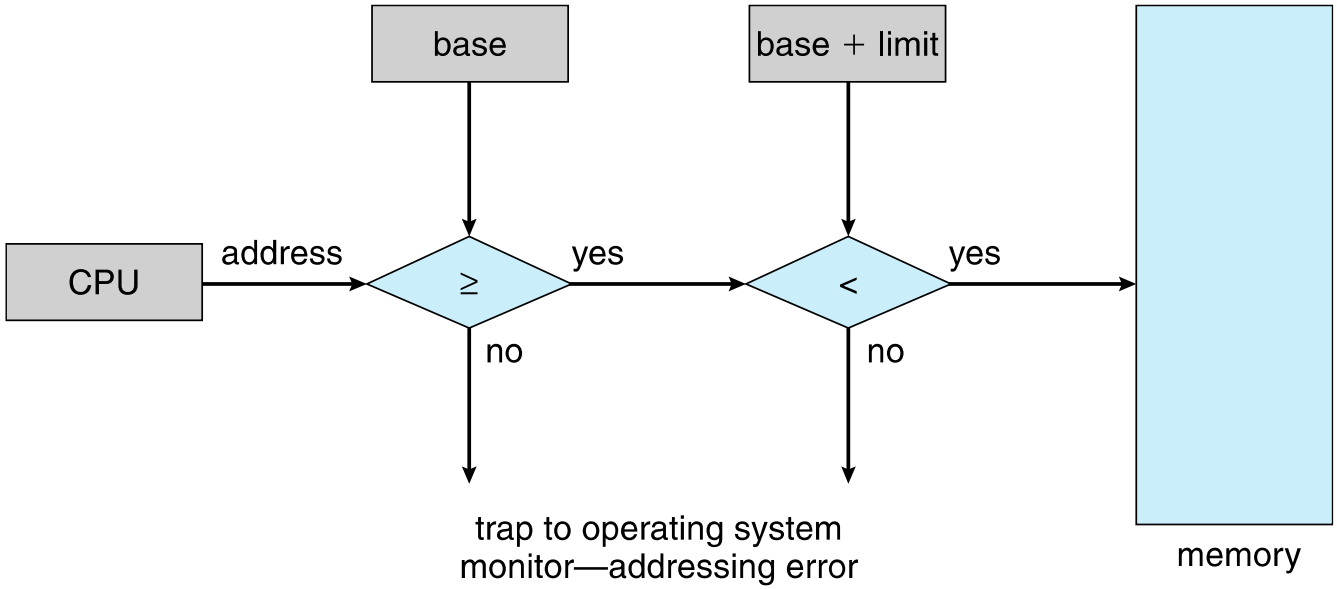
- when static linking is used, the linker combines all other modules needed by a program into a single executable program to avoid any runtime dependency.
- When dynamic linking is used, it is not required to link the actual module or library with the program, rather a reference to the dynamic module is provided at the time of compilation and linking. Dynamic Link Libraries (DLL) in Windows and Shared Objects in Unix are good examples of dynamic libraries.

Base and Limit Registers

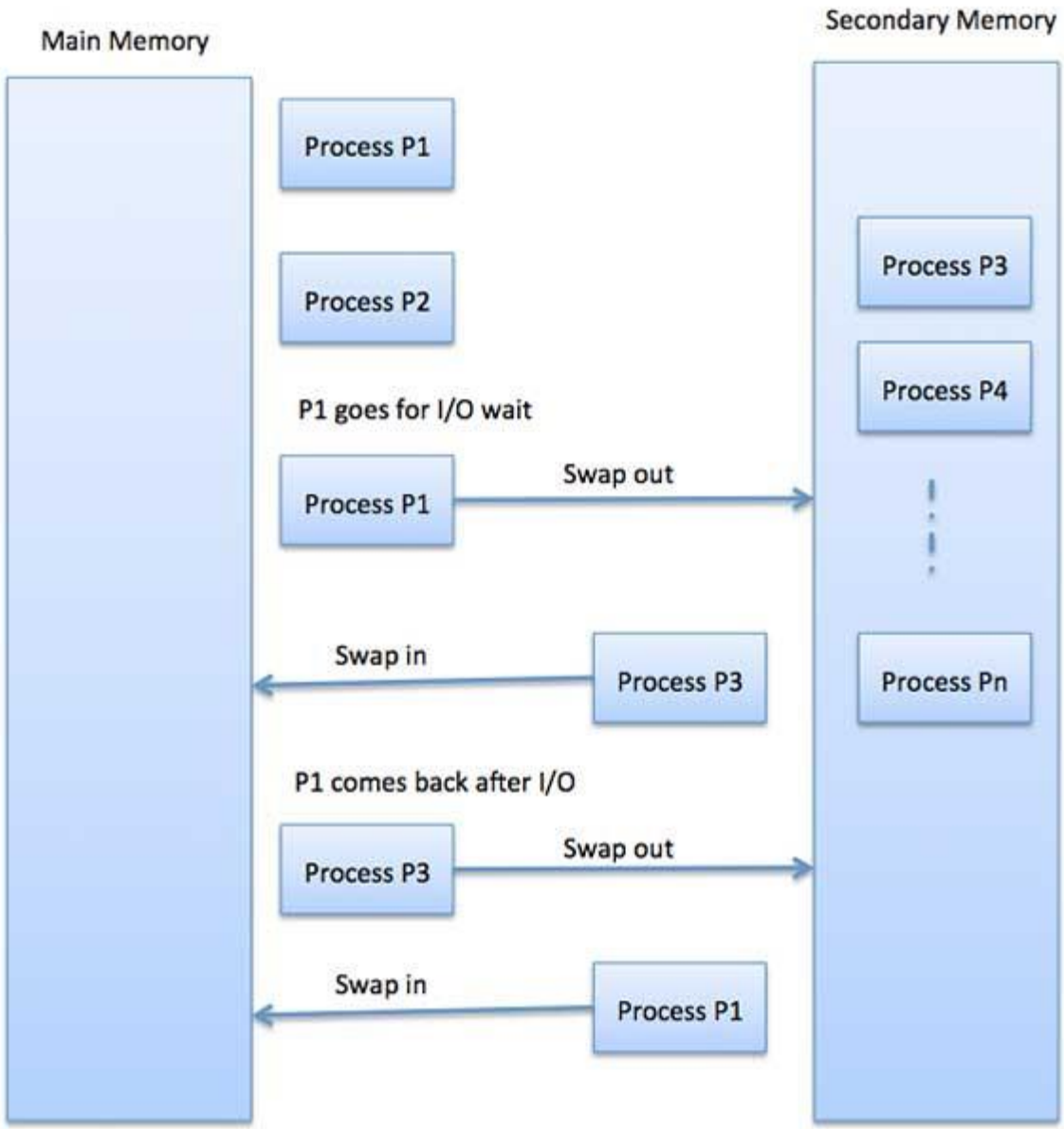
- A pair of **base** and **limit registers** define the logical address space
- CPU must check every memory access generated in user mode to be sure it is between base and limit for that user



Hardware Address Protection



- Swapping
- Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.
- Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.


Main Memory
Secondary Memory
Process P1
Process P2
Process P3
Process P4
P1 goes for I/O wait
Process P1
Swap out
Swap in
Process P3
Process Pn
P1 comes back after I/O
Process P3
Swap out
Swap in
Process P1

Advantages

- Allows higher degree of multi-programming.
- Better utilisation of store.
- Less CPU time wasted on compaction (swap is DMA).

Disadvantages

- Virtual address space <= real address space
- Whole program must be in store when it is executing

four common memory management techniques.

- **Single contiguous allocation:** Simplest allocation method used by MS-DOS. All memory (except some reserved for OS) is available to a process.
- **Partitioned allocation:** Memory is divided in different blocks
- **Paged memory management:** Memory is divided in fixed sized units called page frames, used in virtual memory environment.
- **Segmented memory management:** Memory is divided in different segments (a segment is logical grouping of process' data or code) In this management, allocated memory does'nt have to be contiguous.

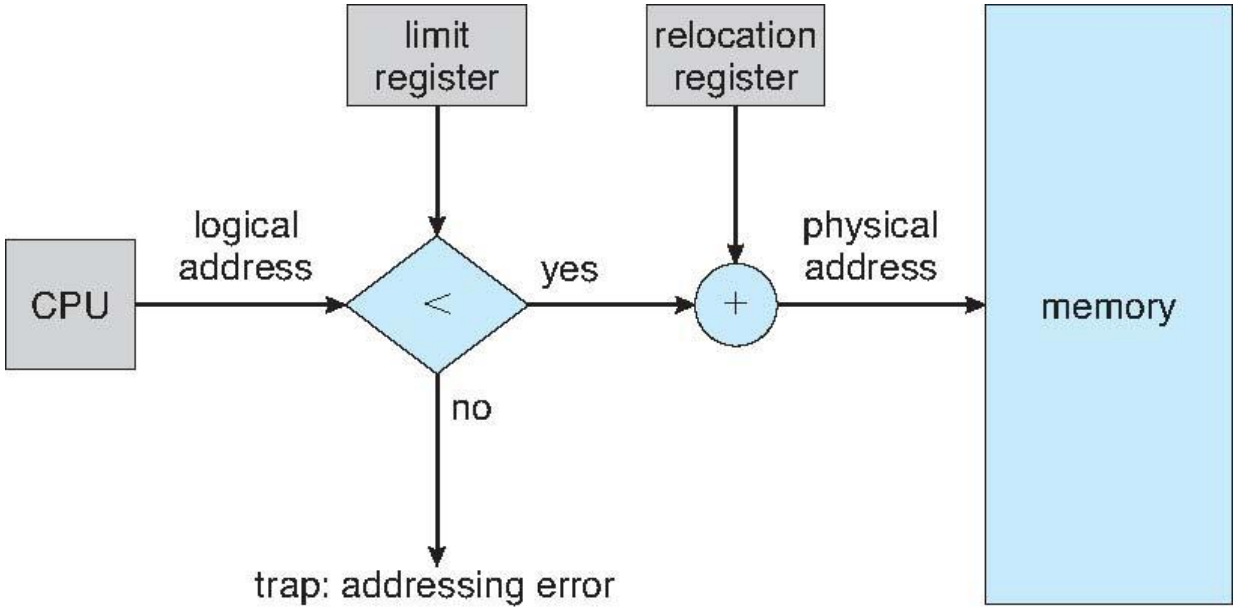
Contiguous Allocation

- Main memory must support both OS and user processes
- Limited resource, must allocate efficiently
- Contiguous allocation is one early method
- Main memory usually into two **partitions**:
  - Resident operating system, usually held in low memory with interrupt vector
  - User processes then held in high memory
  - Each process contained in single contiguous section of memory

Contiguous Allocation (Cont.)

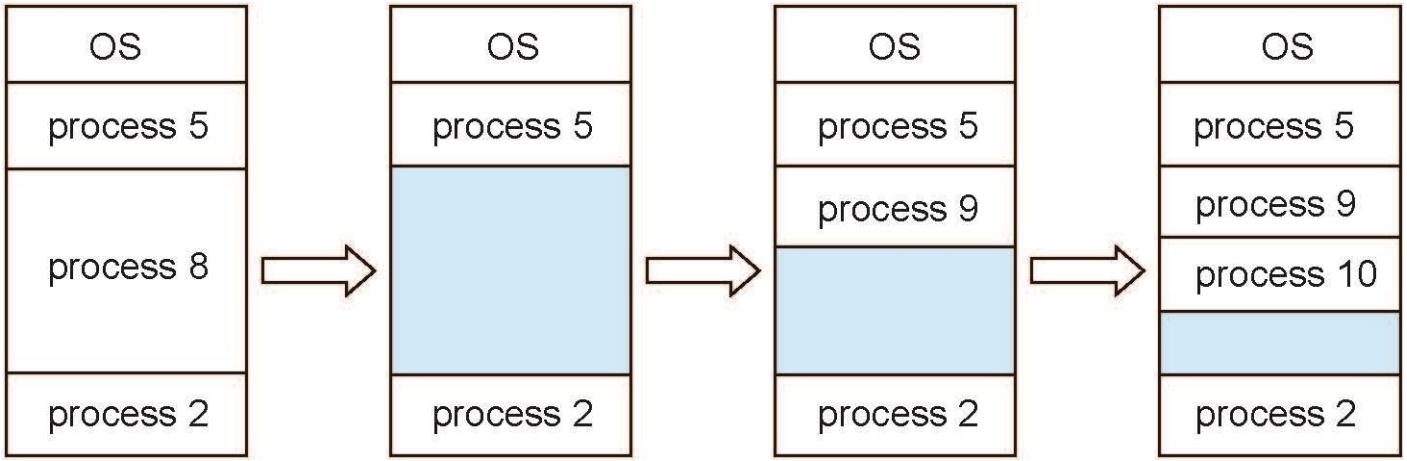
- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
  - Base register contains value of smallest physical address
  - Limit register contains range of logical addresses – each logical address must be less than the limit register
  - MMU maps logical address *dynamically*
  - Can then allow actions such as kernel code being **transient** and kernel changing size

Hardware Support for Relocation and Limit Registers



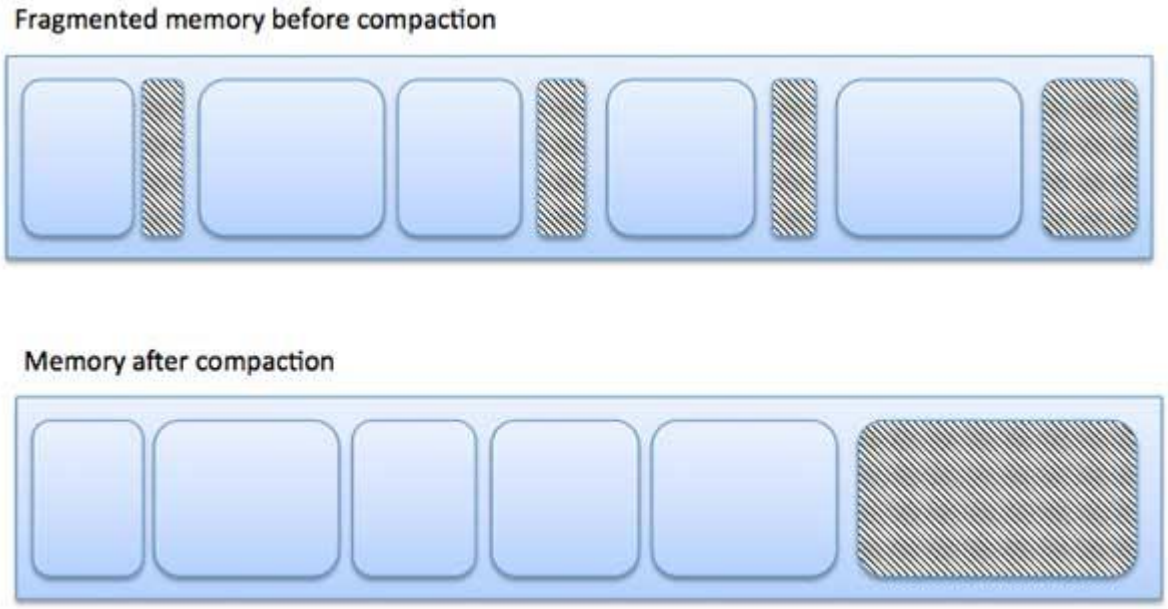
Multiple-partition allocation

- Multiple-partition allocation
  - Degree of multiprogramming limited by number of partitions
  - **Variable-partition** sizes for efficiency (sized to a given process' needs)
  - **Hole** – block of available memory; holes of various size are scattered throughout memory
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it
  - Process exiting frees its partition, adjacent free partitions combined
  - Operating system maintains information about:
    a) allocated partitions b) free partitions (hole)



FRAGMENTATIOM

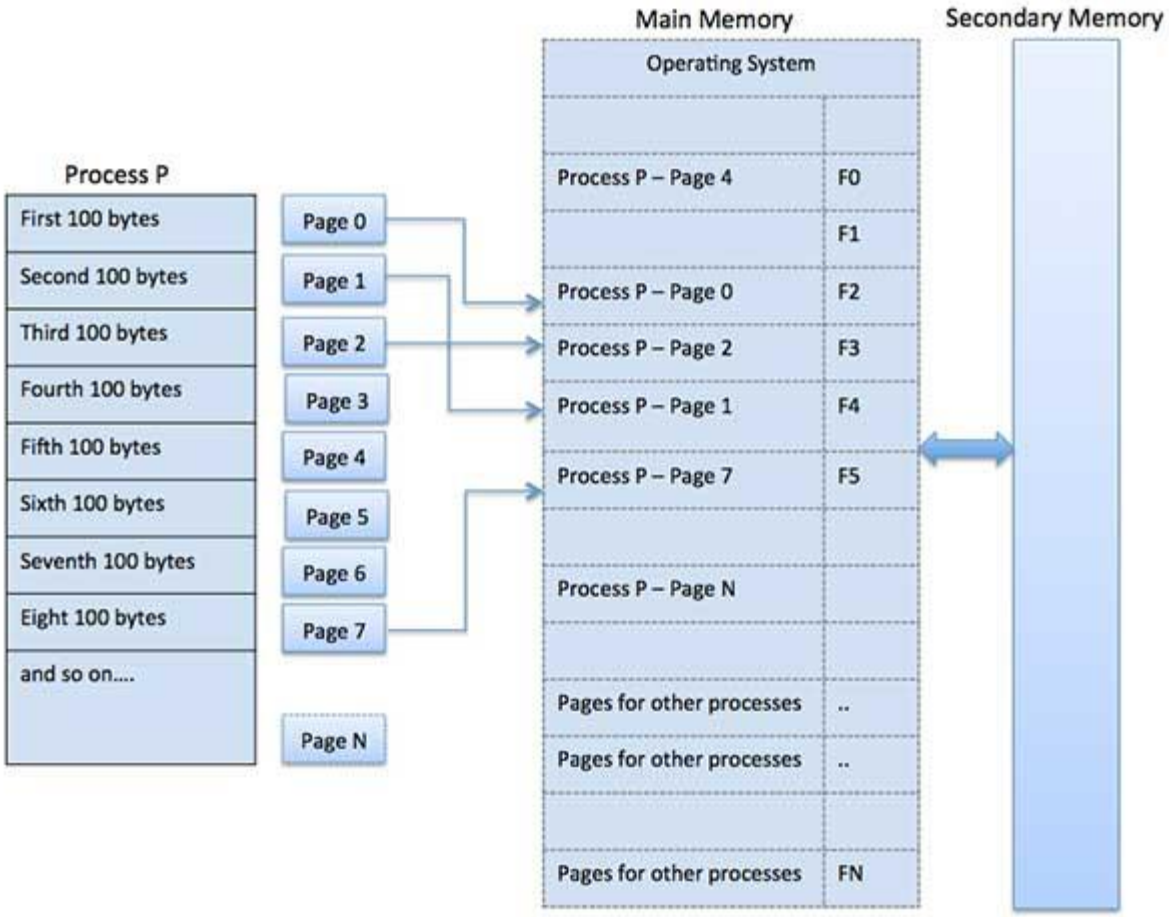
- As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.
- Fragmentation is of two types −
- S.N.Fragmentation & Description
- 1**External fragmentation**
- Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.
- 2**Internal fragmentation**
- Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.
- The following diagram shows how fragmentation can cause waste of memory and a compaction technique can be used to create more free memory out of fragmented memory −



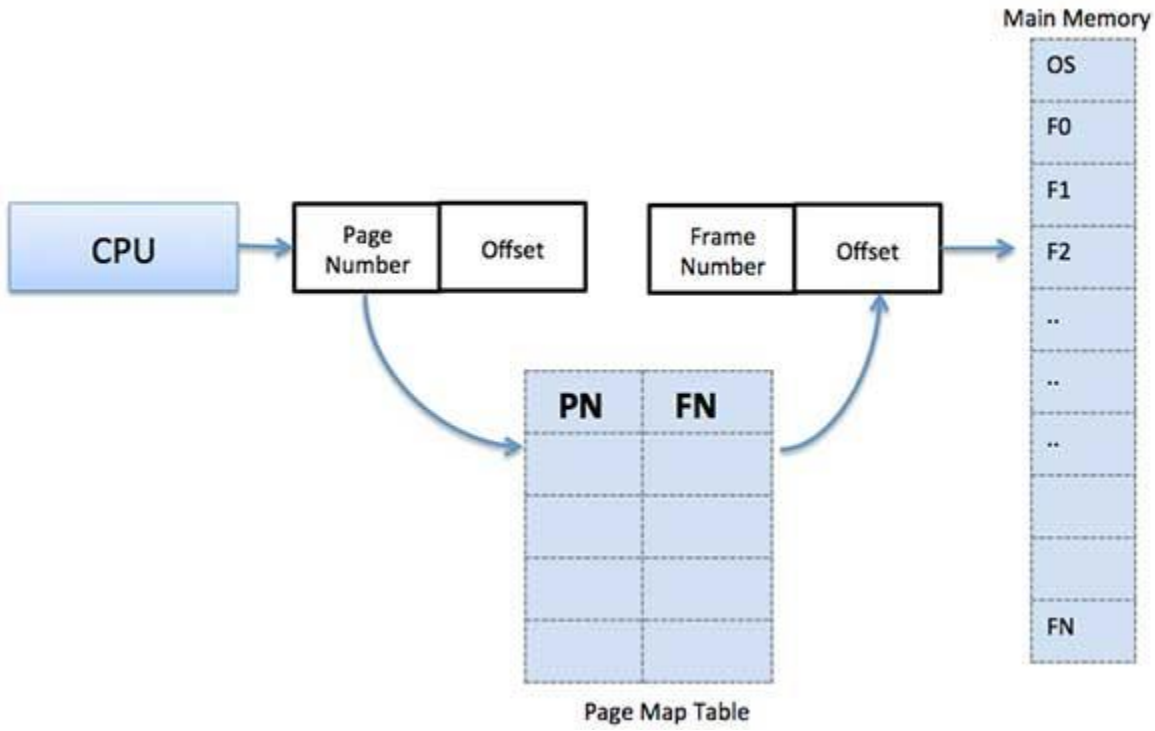
- External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.
- The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

Paging

- A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.
- Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.
- Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.


Main Memory
Secondary Memory
Process P
First 100 bytes
Second 100 bytes
Third 100 bytes
Fourth 100 bytes
Fifth 100 bytes
Sixth 100 bytes
Seventh 100 bytes
Eight 100 bytes
and so on....
Page 0
Page 1
Page 2
Page 3
Page 4
Page 5
Page 6
Page 7
Page N
Operating System
Process P – Page 4
F0
F1
Process P – Page 0
F2
Process P – Page 2
F3
Process P – Page 1
F4
Process P – Page 7
F5
Process P – Page N
Pages for other processes
..
Pages for other processes
..
Pages for other processes
FN

- Address Translation
- Page address is called **logical address** and represented by **page number**and the **offset**.
- Logical Address = Page number + page offset
- Frame address is called **physical address** and represented by a **frame number** and the **offset**.
- Physical Address = Frame number + page offset A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.
-

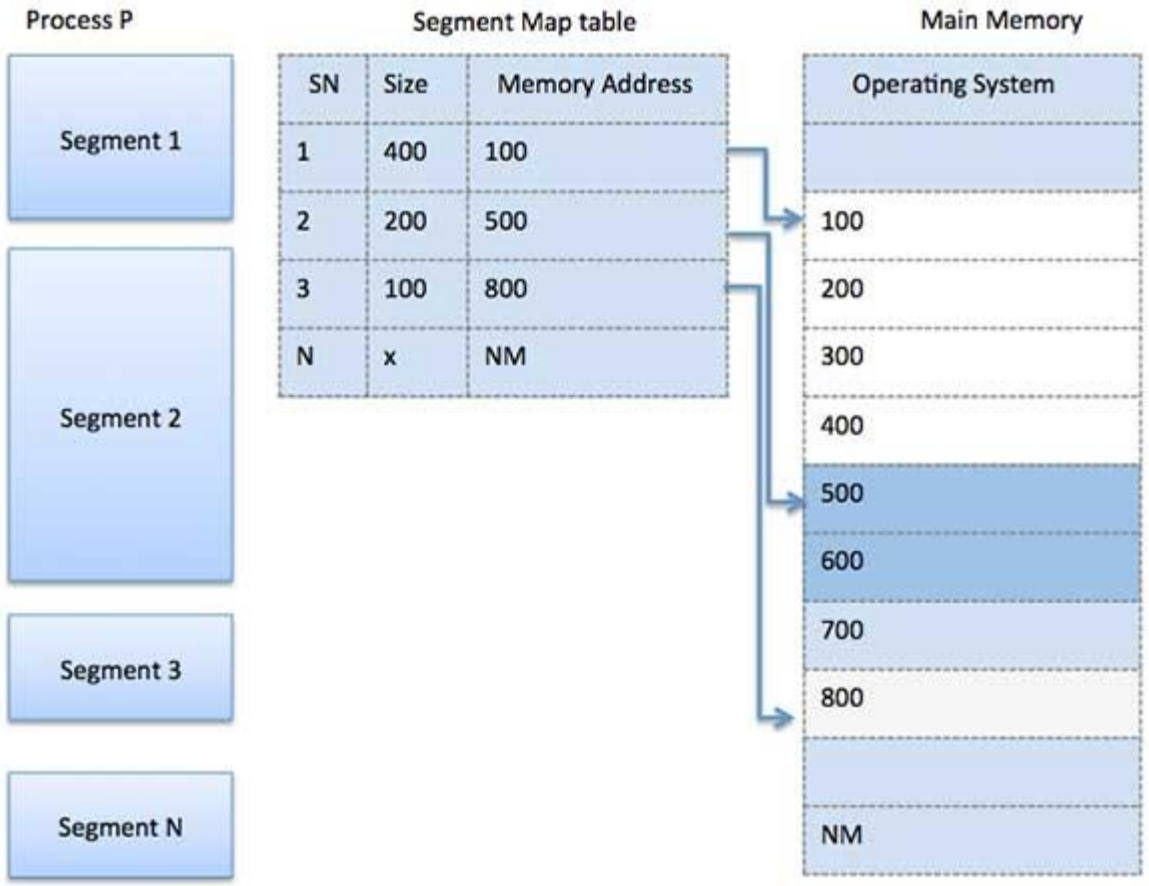

Main Memory
OS
F0
F1
F2
..
..
..
FN
CPU
Page Number
Offset
Frame Number
Offset
PN
FN
Page Map Table

- When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.
- This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

- Advantages and Disadvantages of Paging
- Here is a list of advantages and disadvantages of paging –
- Paging reduces external fragmentation, but still suffer from internal fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

Segmentation

- Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.
- When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.
- Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.
- A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.


Process P
Segment 1
Segment 2
Segment 3
Segment N
Segment Map table
SN
Size
Memory Address
1
400
100
2
200
500
3
100
800
N
x
NM
Main Memory
Operating System
100
200
300
400
500
600
700
800
NM

Virtual Memory

- A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.
- The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes.
- First, it allows us to extend the use of physical memory by using disk.
- Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

User written error handling routines are used only when an error occurred in the data or computation.

Certain options and features of a program may be used rarely.

Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.

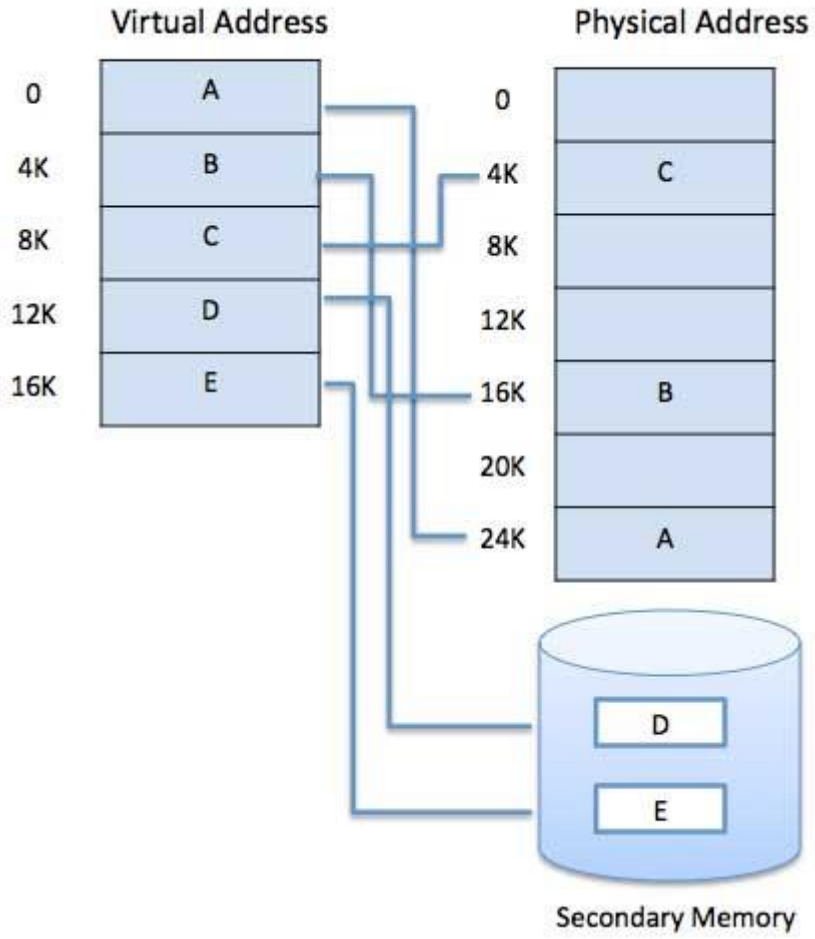
The ability to execute a program that is only partially in memory would counter many benefits.

Less number of I/O would be needed to load or swap each user program into memory.

A program would no longer be constrained by the amount of physical memory that is available.

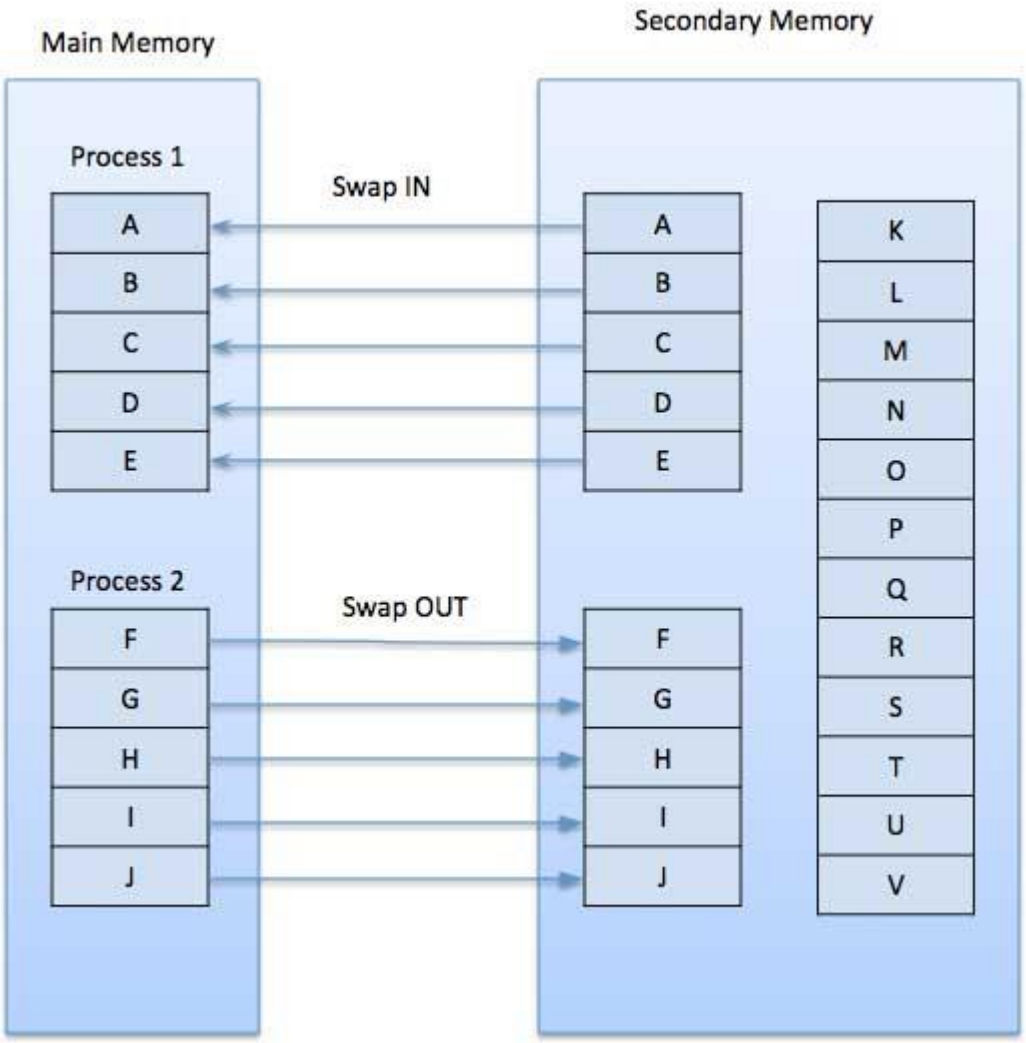
Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

- Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below −
- Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.


Virtual Address
Physical Address
0
4K
8K
12K
16K
A
B
C
D
E
0
4K
8K
12K
16K
20K
24K
C
B
A
D
E
Secondary Memory

Demand Paging

- A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.
- When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory
- Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.
-


Main Memory
Secondary Memory
Process 1
Swap IN
A
B
C
D
E
A
B
C
D
E
K
L
M
N
O
P
Q
R
S
T
U
V
Process 2
Swap OUT
F
G
H
I
J
F
G
H
I
J

- While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.
- Advantages
- Following are the advantages of Demand Paging −
- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.
- Disadvantages
- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

The various partition allocation schemes

- :
- **1. First Fit**: In the first fit, partition is allocated which is first sufficient from the top of Main Memory.
- **2. Best Fit** Allocate the process to the partition which is first smallest sufficient partition among the free available partition.
- **3. Worst Fit** Allocate the process to the partition which is largest sufficient among the freely available partitions available in the main memory.
- **4. Next Fit** Next fit is similar to the first fit but it will search for the first sufficient partition from the last allocation point.

Page Replacement Algorithms

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).

Types of Page Replacement Algorithms

There are various page replacement algorithms. Each algorithm has a different method by which the pages can be replaced.

Optimal Page Replacement algorithm → this algorithms replaces the page which will not be referred for so long in future. Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.

Least recent used (LRU) page replacement algorithm → this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.

FIFO → in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

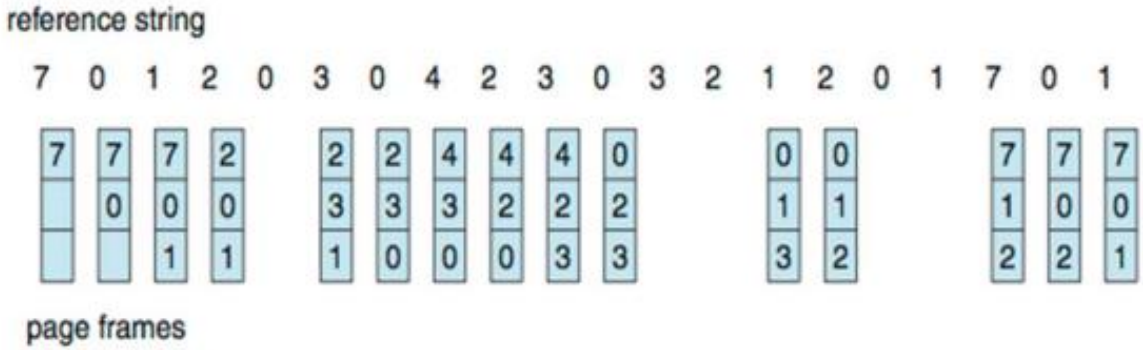
First In First Out (FIFO)

This is the simplest page replacement algorithm. In this algorithm, the OS maintains a queue that keeps track of all the pages in memory, with the oldest page at the front and the most recent page at the back.

When there is a need for page replacement, the FIFO algorithm, swaps out the page at the front of the queue, that is the page which has been in the memory for the longest time.

First-In-First-Out (FIFO) Algorithm

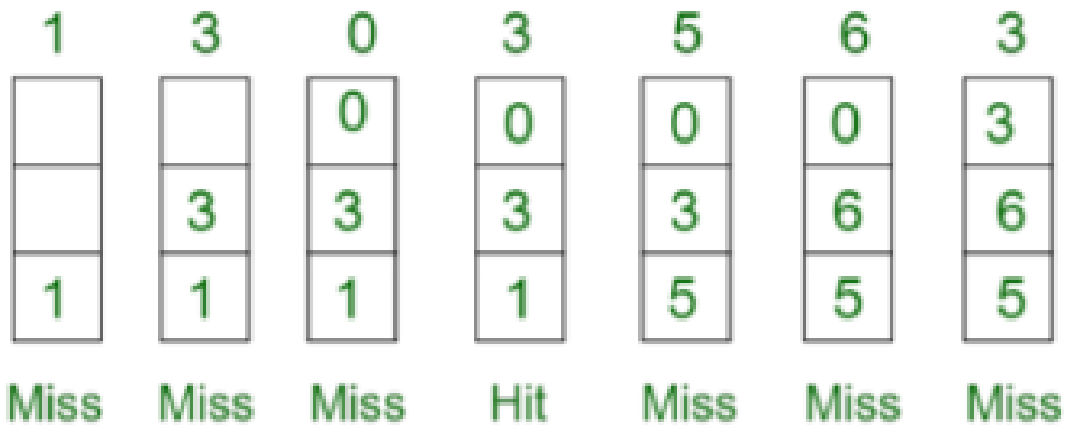
- Reference string: **7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1**
- 3 frames (3 pages can be in memory at a time per process)



15 page faults

Example Question: Consider the page reference string 1, 3, 0, 3, 5, 6 with 3-page frames. Find the number of page faults. Show your work.

Page reference 1, 3, 0, 3, 5, 6, 3



Total Page Fault = 6

Advantages

Simple and easy to implement.

Low overhead.

Disadvantages

Poor performance.

Doesn’t consider the frequency of use or last used time, simply replaces the oldest page.

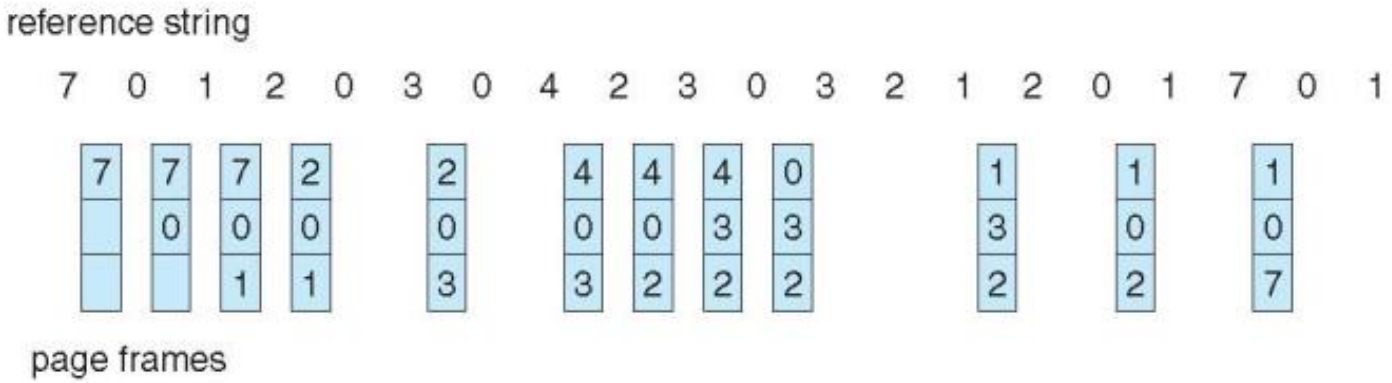
Least Recently Used (LRU)

Least Recently Used page replacement algorithm keeps track of page usage over a short period of time. It works on the idea that the pages that have been most heavily used in the past are most likely to be used heavily in the future too.

In LRU, whenever page replacement happens, the page which has not been used for the longest amount of time is replaced.

Least Recently Used (LRU) Algorithm

- Use past knowledge rather than future
- Replace page that has not been used in the most amount of time
- Associate time of last use with each page



- 12 faults – better than FIFO but worse than OPT

For Example

1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	5	5	5	5	5	5	2	2
	2	2	2	2	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	6	6	6	6
M	M	M	M	M	M	H	H	M	H	M	H

M = Miss
H = Hit

Total Page Fault = 8

Page Fault ratio = 8/12

Advantages
Efficient.

Disadvantages
Complex Implementation.
Expensive.
Requires hardware support.

Optimal Page Replacement

Optimal Page Replacement algorithm is the best page replacement algorithm as it gives the least number of page faults. It is also known as OPT, clairvoyant replacement algorithm, or Belady's optimal page replacement policy.

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future, i.e., the pages in the memory which are going to be referred farthest in the future are replaced.

For Example

1	2	3	4	5	1	3	1	6	3	2	3

1	1	1	1	1	1	1	1	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3
			4	5	5	5	5	5	5	5	5
M	M	M	M	M	H	H	H	M	H	H	H

M = Miss
H = Hit

Total Page Fault = 6

Page Fault ratio = 6/12

Advantages

Easy to Implement.

Simple data structures are used.

Highly efficient.

Disadvantages

Requires future knowledge of the program.

Time-consuming.

- References

- Operating System Concepts:8th Edition Wiley, by Abraham Silberschatz, Peter Baer Galvin
- https://slideplayer.com/slide/1659380/

Chapter 4

File Systems

A file is a collection of related information that is recorded on secondary storage.

File System Interface

The user level (more visible) portion of the file system.

- Access methods
- Directory Structure
- Protection

File System Implementation

The OS level (less visible) portion of the file system.

- Allocation and Free Space Management
- Directory Implementation

Types of files

- Text file: It is a sequence of characters
- Source file: is sequence of sub routines and functions
- Executable files: is a series of code sections that the loader can being into memory nd execute.

Attributes of a File

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring

- Information about files is kept in the directory structure, which is maintained on the disk.

File Naming

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

Typical file extensions.

File Operations

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set Attributes
11. Rename

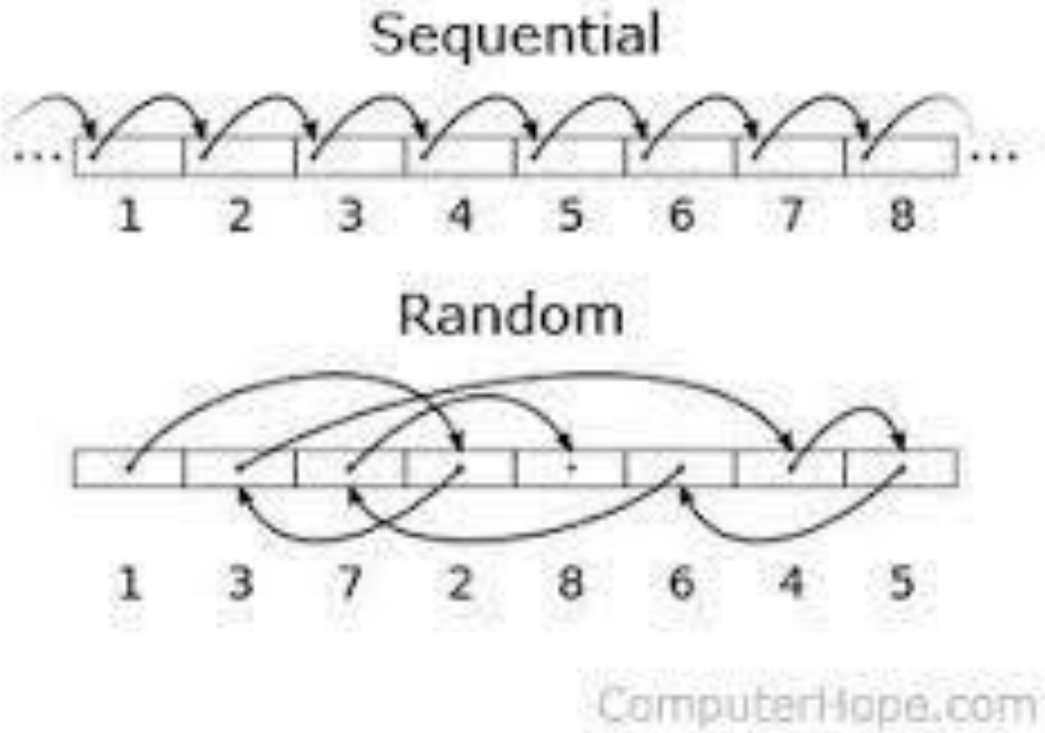
File Access

- Sequential Access
- Direct Access
- Indexed Sequential Access

File Access

- Sequential access
  - read all bytes/records from the beginning
  - cannot jump around, could rewind or back up
  - convenient when medium was mag tape
  - The bulk of operations on a file is reads and write.
  - A read operation needs the next portion of the file and automatically advance a file pointer, which tracks the I/O location.
  - When write appends to the end of the file and advances to the end newly written material.

Access Method Diagram



Direct Access –

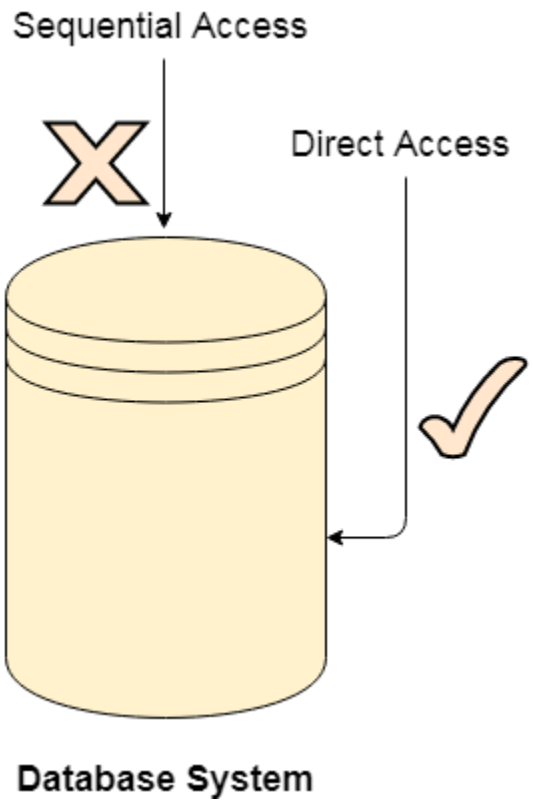
Another method is *direct access method* also known as *relative access method*.

A filed-length logical record that allows the program to read and write record rapidly. in no particular order.

The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record.

Thus, we may read block 14 then block 59, and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

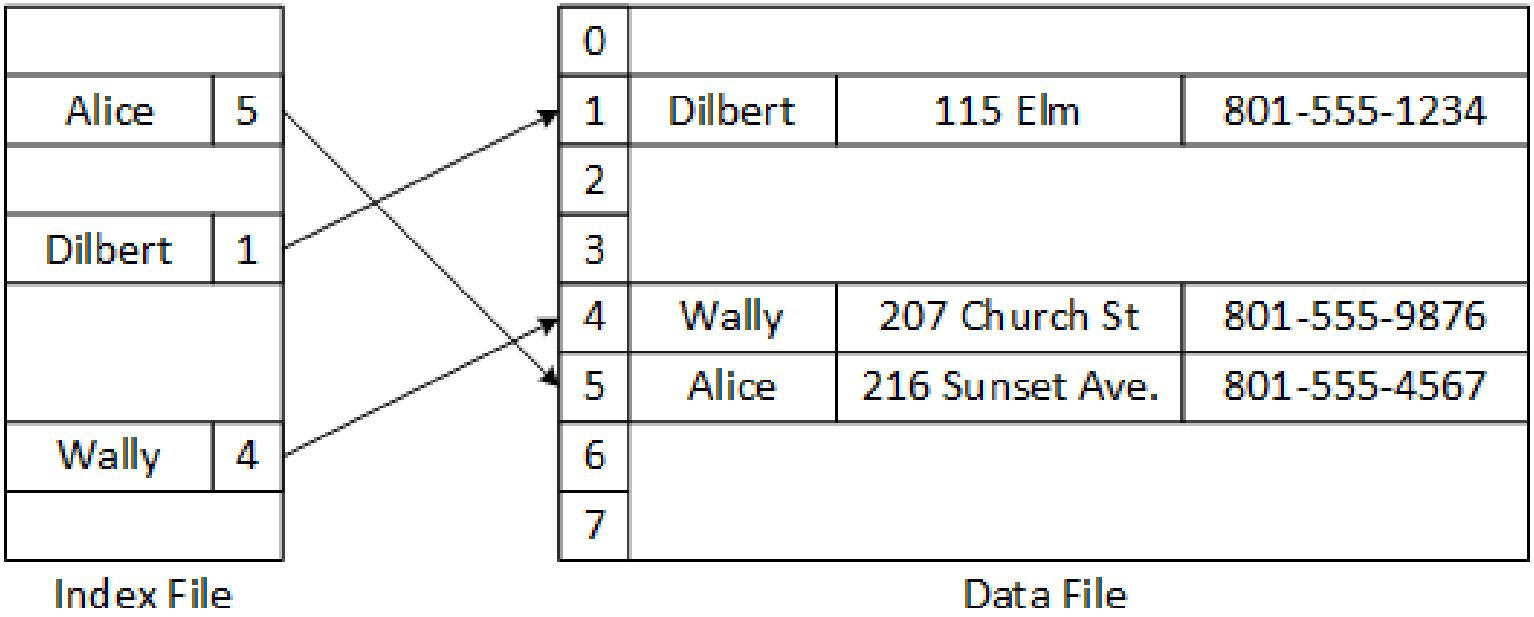

Sequential Access
Direct Access
Database System

- **Index sequential method –**
  It is the other method of accessing a file that is built on the top of the sequential access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks.
- To find a record in the file, we first search the index, and then by the help of pointer we access the file directly.

Key points:

•It is built on top of Sequential access.

•It control the pointer by using index.


Alice
5
Dilbert
1
Wally
4
Index File
0
1
2
3
4
5
6
7
Dilbert
115 Elm
801-555-1234
Wally
207 Church St
801-555-9876
Alice
216 Sunset Ave.
801-555-4567
Data File

- Random access
  - bytes/records read in any order
  - essential for data base systems
  - A file is made up of fixed length logical records that allows programs to read and write records rapidly in no particular order.
  - Immediate access to large amount of information.

- Other access methods:
  - Based on index for a file.
  - It contains pointer to the various blocks
  - To find a record in the file, we first search the index, and then use the pointer to access the file directory and to find the desired record.

Directory Structures

Directories maintain information about files:

For a large number of files, may want a directory structure - directories under directories.

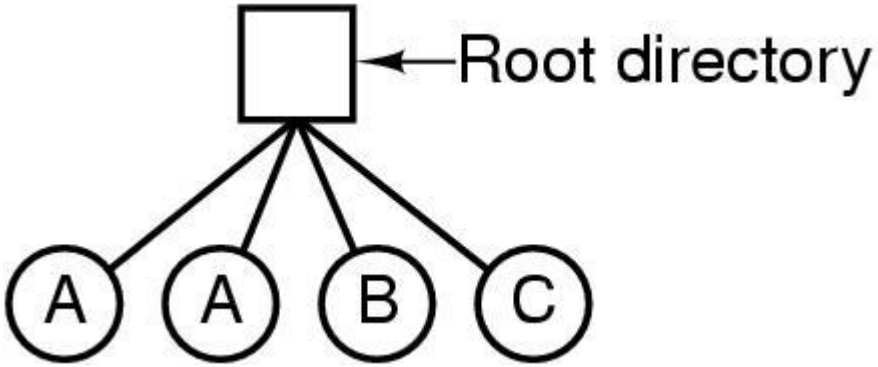
Directory Operations

1. Create
2. Delete
3. Opendir
4. Closedir
5. Search
6. Rename
7. Link
8. Unlink

DIRECTORY STRUCTURE

- Single Level Directory
- Two-level Directory
- Tree – Structured Directory

Directories
Single-Level Directory Systems



- A single level directory system
  - contains 4 files
  - owned by 3 different people, A, B, and C

Simple Structure for a Directory

- List of entries, one for each file
- Sequential file with the name of the file serving as the key
- Provides no help in organizing the files
- Forces user to be careful not to use the same name for two different files

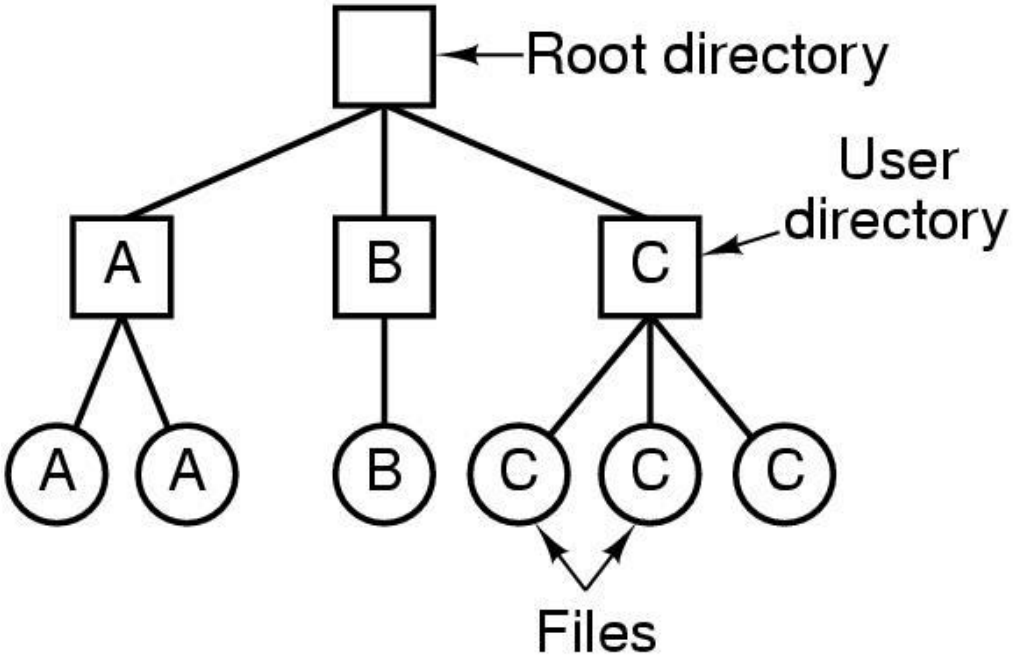
Advantages:

Since it is a single directory, so its implementation is very easy.
If the files are smaller in size, searching will become faster.
The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

Disadvantages:

There may chance of name collision because two files can have the same name.
Searching will become time taking if the directory is large.
This can not group the same type of files together.

Two-level Directory Systems



Letters indicate *owners* of the directories and files

Two-level Scheme for a Directory

- One directory for each user and a master directory
- Master directory contains entry for each user
  - Provides address and access control information
- Each user directory is a simple list of files for that user
- Still provides no help in structuring collections of files

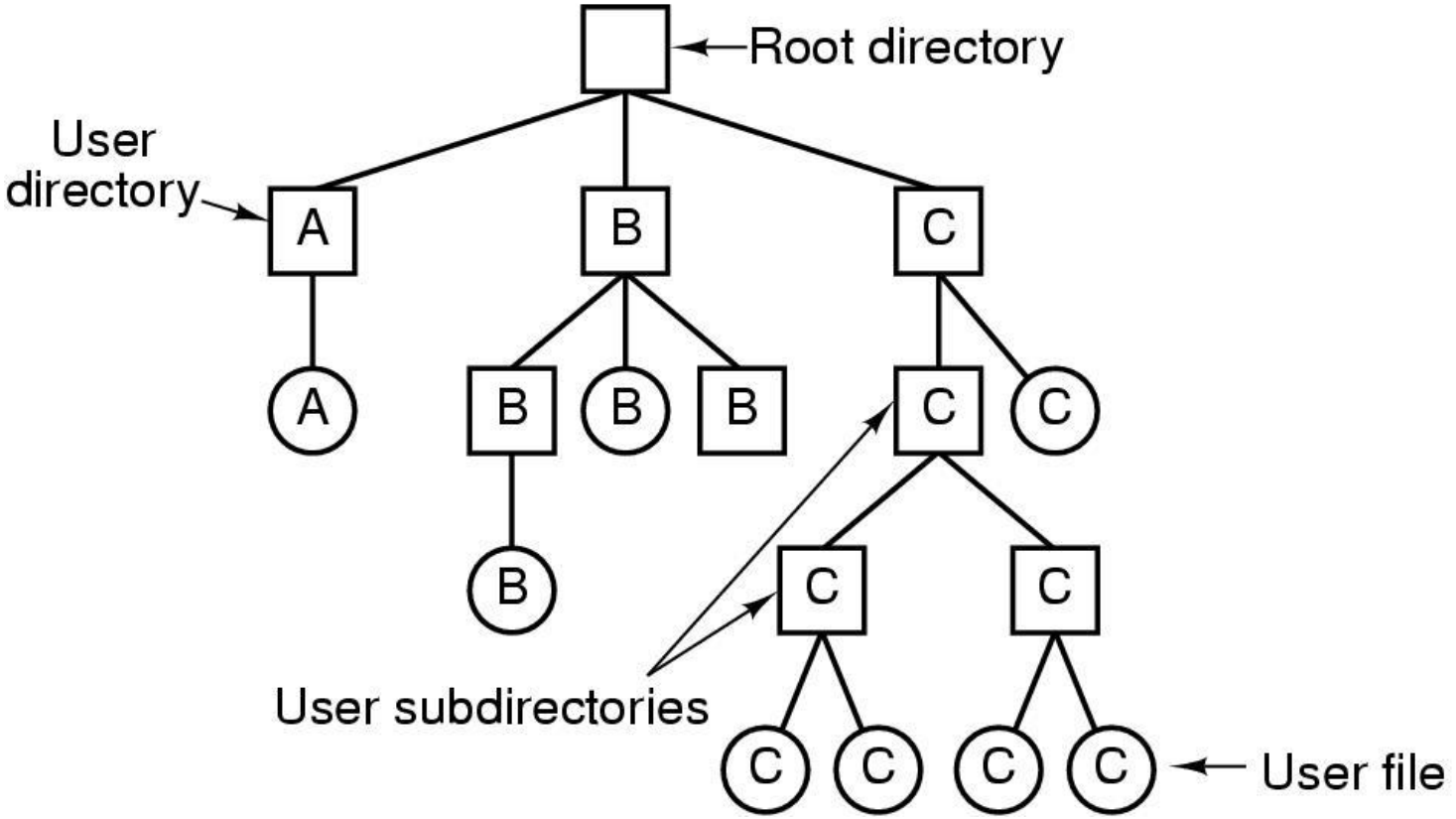
Advantages:

We can give full path like /User-name/directory-name/.
Different users can have the same directory as well as the file name.
Searching of files becomes easier due to pathname and user-grouping.

Disadvantages:

A user is not allowed to share files with other users.
Still, it not very scalable, two files of the same type cannot be grouped together in the same user.

Hierarchical Directory Systems



A hierarchical directory system

Hierarchical, or Tree-Structured Directory

- Master directory with user directories underneath it
- Each user directory may have subdirectories and files as entries

Hierarchical, or Tree-Structured Directory

- Files can be located by following a path from the root, or master, directory down various branches
  - This is the pathname for the file
- Can have several files with the same file name as long as they have unique path names
- Current directory is the working directory
- Files are referenced relative to the working directory

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

File Protection

- None
  - User may not know of the existence of the file
  - User is not allowed to read the user directory that includes the file
- Knowledge
  - User can only determine that the file exists and who its owner is

- Execution
  - The user can load and execute a program but cannot copy it
- Reading
  - The user can read the file for any purpose, including copying and execution
- Appending
  - The user can add data to the file but cannot modify or delete any of the file's contents

- Updating
  - The user can modify, deleted, and add to the file's data. This includes creating the file, rewriting it, and removing all or part of the data
- Changing protection
  - User can change access rights granted to other users
- Deletion
  - User can delete the file

- Owners
  - Has all rights previously listed
  - May grant rights to others using the following classes of users
    - Specific user
    - User groups
    - All for public files

Other file security

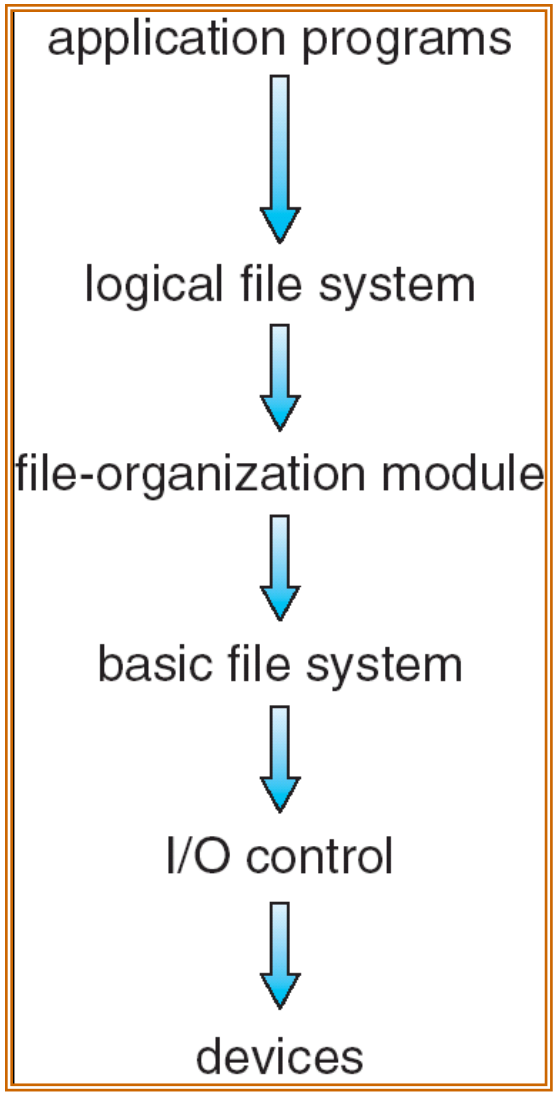
- Naming a file
- Password
- Access control

FILE SYSTEM STRUCTURE

When talking about “the file system”, you are making a statement about both the rules used for file access, and about the algorithms used to implement those rules. Here’s a breakdown of those algorithmic pieces.

Application Programs	The code that's making a file request.
Logical File System	This is the highest level in the OS; it does protection, and security. Uses the directory structure to do name resolution.
File-organization Module	Here we read the file control block maintained in the directory so we know about files and the logical blocks where information about that file is located.
Basic File System	Knowing specific blocks to access, we can now make generic requests to the appropriate device driver.
IO Control	These are device drivers and interrupt handlers. They cause the device to transfer information between that device and CPU memory**.**
Devices	The disks / tapes / etc.

Layered File System



Handles the CONTENT of the file. Knows the file's internal structure.

Handles the OPEN, etc. system calls. Understands paths, directory structure, etc.

Uses directory information to figure out blocks, etc. Implements the READ. POSITION calls.

Determines where on the disk blocks are located.

Interfaces with the devices – handles interrupts.

File Allocation Method

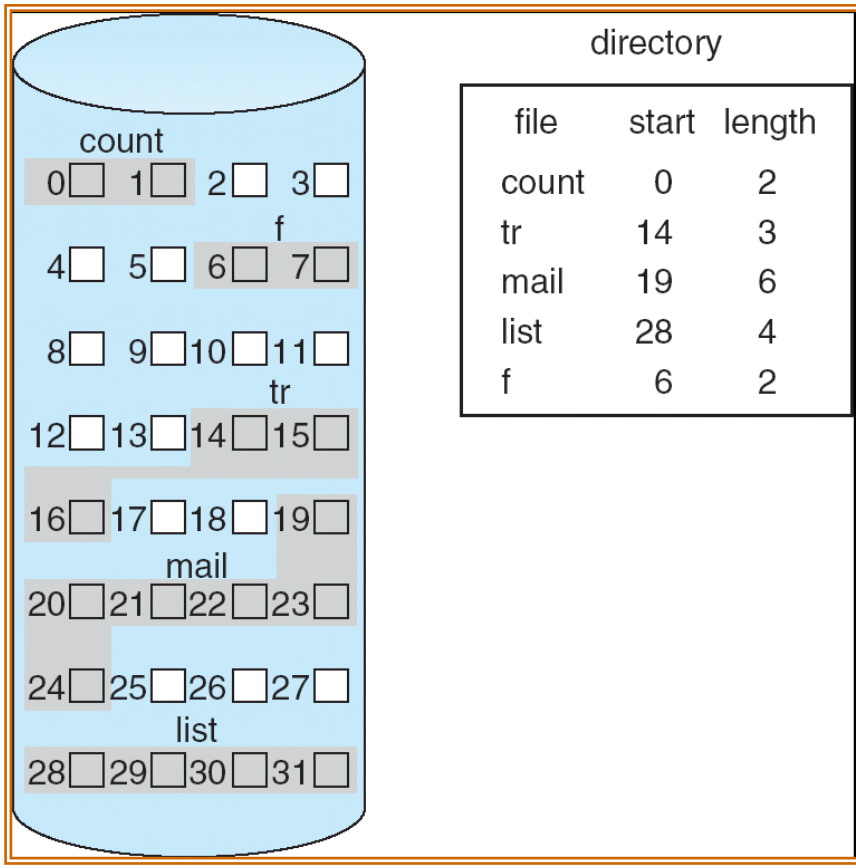
CONTIGUOUS ALLOCATION

- Method: Lay down the entire file on contiguous sectors of the disk. Define by a dyad <first block location, length >.

a) Accessing the file requires a minimum of head movement.

b) Easy to calculate block location: block i of a file, starting at disk address **b**, is **b + i.**

c) Difficulty is in finding the contiguous space, especially for a large file. Problem is one of dynamic allocation (first fit, best fit, etc.) which has external fragmentation. If many files are created/deleted, compaction will be necessary.

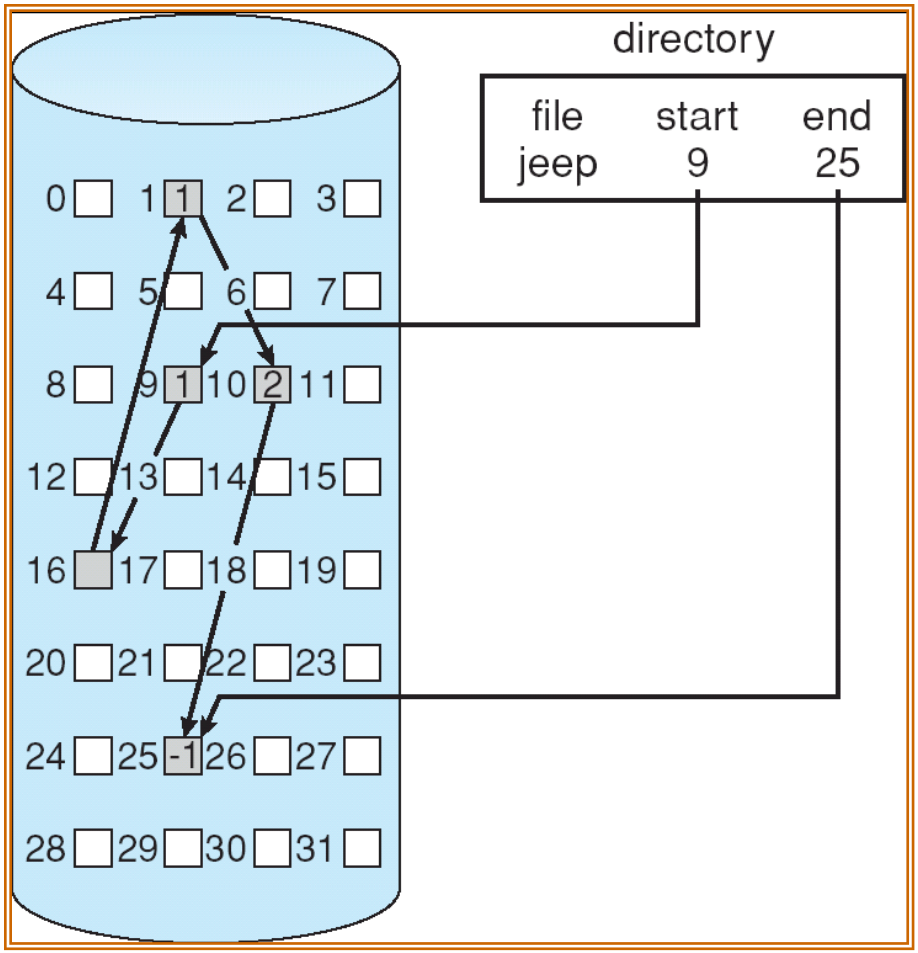


- It's hard to estimate at create time what the size of the file will ultimately be. What happens when we want to extend the file --- we must either terminate the owner of the file, or try to find a bigger hole.

FILE ALLOCATION METHOD

LINKED ALLOCATION

Each file is a linked list of disk blocks, scattered anywhere on the disk.



At file creation time, simply tell the directory about the file. When writing, get a free block and write to it, enqueueing it to the file header.

There's no external fragmentation since each request is for one block.

Method can only be effectively used for sequential files.

LINKED ALLOCATION

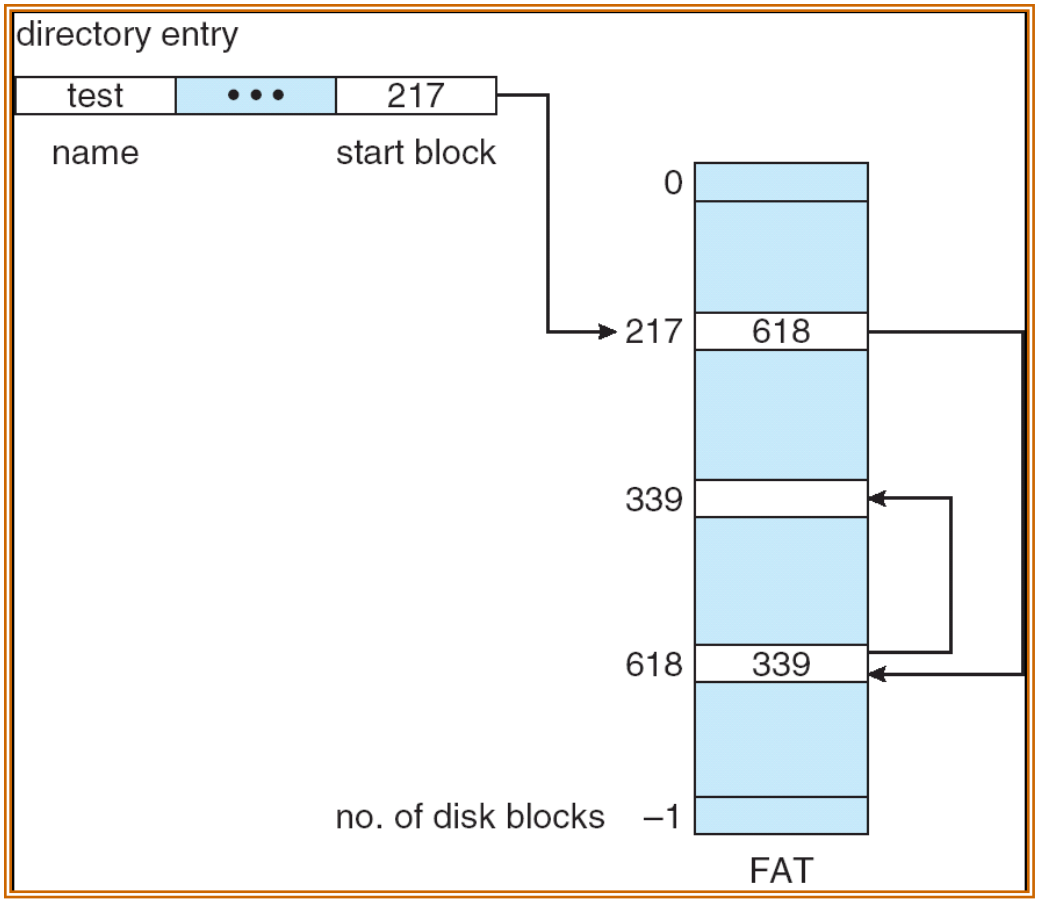
Pointers use up space in each block. Reliability is not high because any loss of a pointer loses the rest of the file.

A File Allocation Table is a variation of this.

It uses a separate disk area to hold the links.

This method doesn't use space in data blocks. Many pointers may remain in memory.

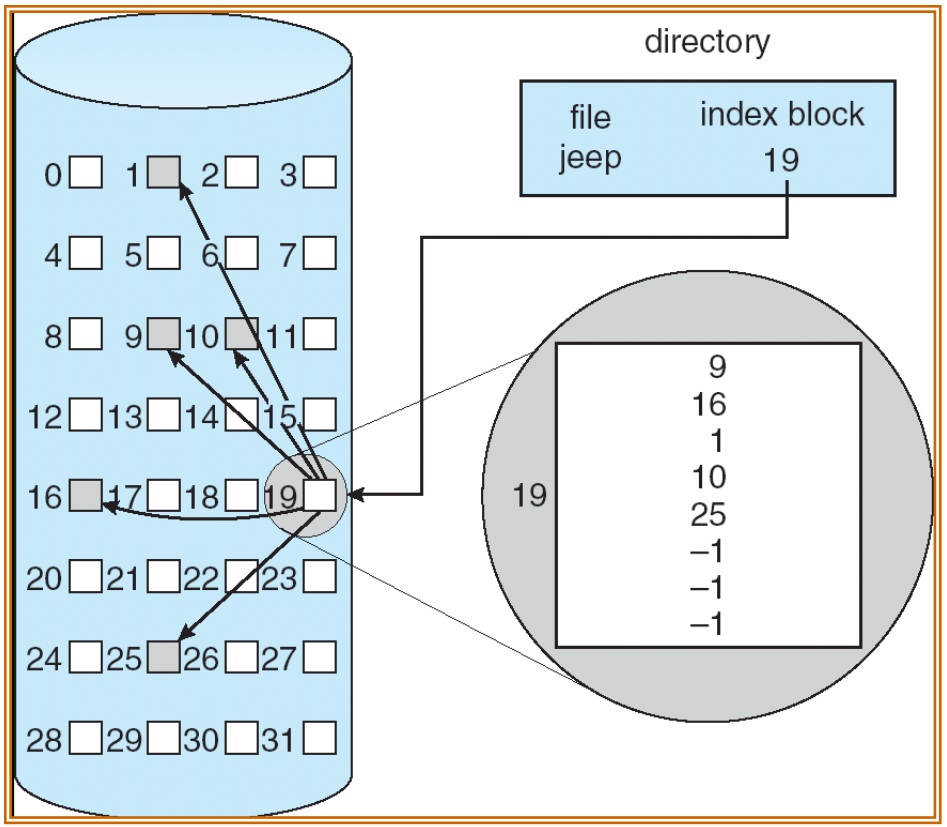
A FAT file system is used by MS-DOS.



Allocation Methods

INDEXED ALLOCATION

- Each file uses an index block on disk to contain addresses of other disk blocks used by the file.
- When the **i th** block is written, the address of a free block is placed at the **i th** position in the index block.
- Method suffers from wasted space since, for small files, most of the index block is wasted. What is the optimum size of an index block?
- If the index block is too small, we can:
    a) Link several together
    b) Use a multilevel index



UNIX keeps 12 pointers to blocks in its header. If a file is longer than this, then it uses pointers to single, double, and triple level index blocks.

Free Space Management

We need a way to keep track of space currently free. This information is needed when we want to create or add (allocate) to a file. When a file is deleted, we need to show what space is freed up.

BIT VECTOR METHOD

- Each block is represented by a bit

  1 1 0 0 1 1 0 means blocks 2, 3, 6 are free.

- This method allows an easy way of finding contiguous free blocks. Requires the overhead of disk space to hold the bitmap.

- A block is not REALLY allocated on the disk unless the bitmap is updated.

FREE LIST METHOD

- Free blocks are chained together, each holding a pointer to the next one free.
- This is very inefficient since a disk access is required to look at each sector.

GROUPING METHOD

- In one free block, put lots of pointers to other free blocks. Include a pointer to the next block of pointers.

COUNTING METHOD

- Since many free blocks are contiguous, keep a list of dyads holding the starting address of a "chunk", and the number of blocks in that chunk.
- Format < disk address, number of free blocks >

Disk Strucuture

- Kernel – it is a bridge between applications and the actual data processing done at the hardware level
- I/O sub system – portions of kernel called software layers
- Device Driver-A device driver is a program that controls a particular type of device that is attached to your computer. There are device drivers for printers, displays, CD-ROM readers, diskette drives, and so on.

- Device Controller - A *device controller* is a part of a computer system that makes sense of the signals going to, and coming from the CPU processor.

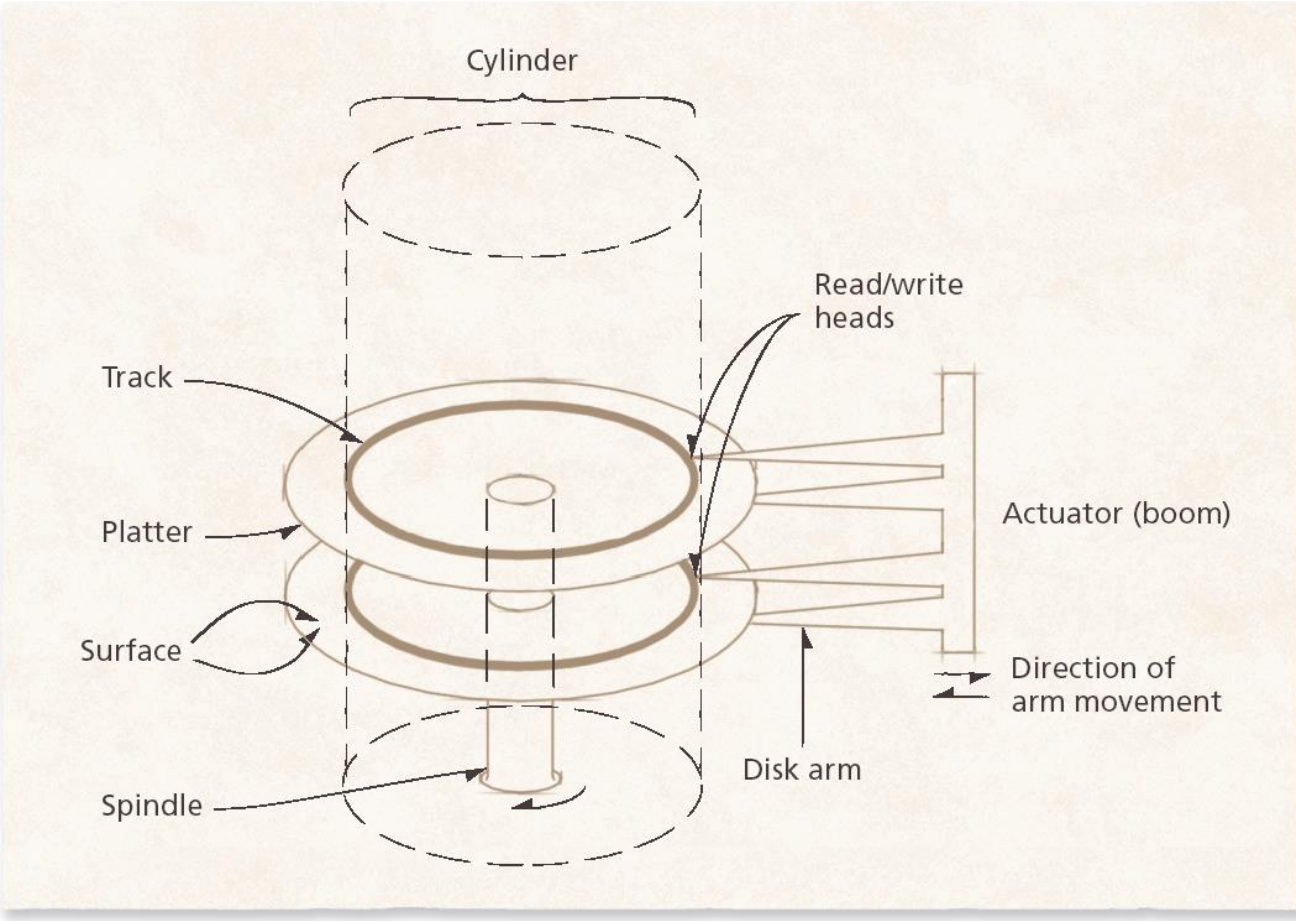
DISK STRUCTURE

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.

- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

12.3 Characteristics of Moving-Head Disk Storage

- Physical layout of disk drives
  - Set of magnetic platters
    - Rotate on spindle
    - Made up of tracks, which in turn contain sectors
    - Vertical sets of tracks form cylinders

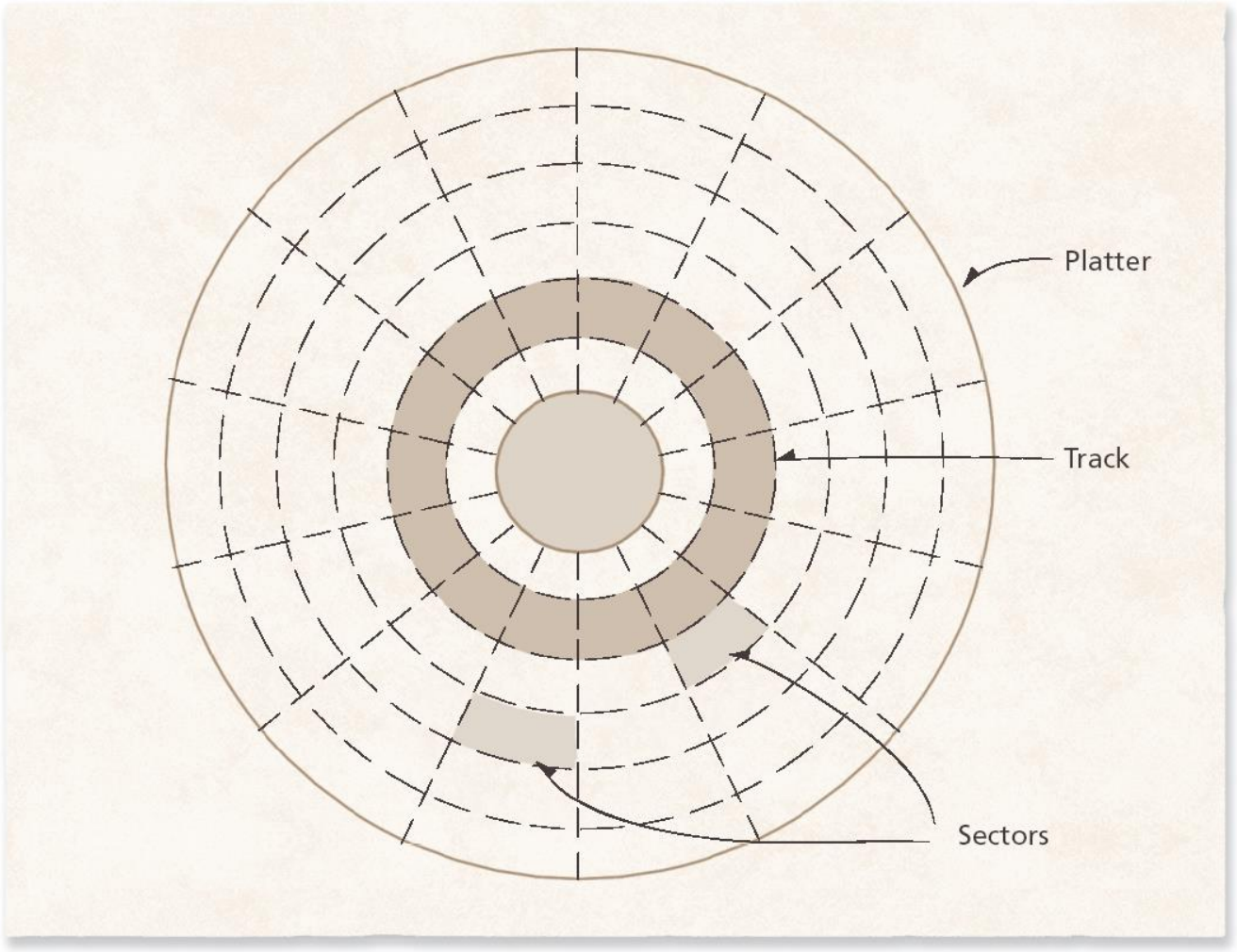


12.3 Characteristics of Moving-Head Disk Storage

- Performance measurements
  - Rotational latency
    - Time for data to rotate from current position to read-write head
  - Seek time
    - Time for read-write head to move to new cylinder
  - Transmission time
    - Time for all desired data to spin by read-write head

Model (Environment)	Average Seek Time (ms)	Average Rotational Latency (ms)
Maxtor DiamondMax Plus 9 (High-end desktop)	9.3	4.2
WD Caviar (High-end desktop)	8.9	4.2
Toshiba MK8025GAS (Laptop)	12.0	7.14
WD Raptor (Enterprise)	5.2	2.99
Cheetah 15K.3 (Enterprise)	3.6	2.0

- Disks divide tracks into several sectors, each typically containing 512 bytes



12.4 Why Disk Scheduling Is Necessary

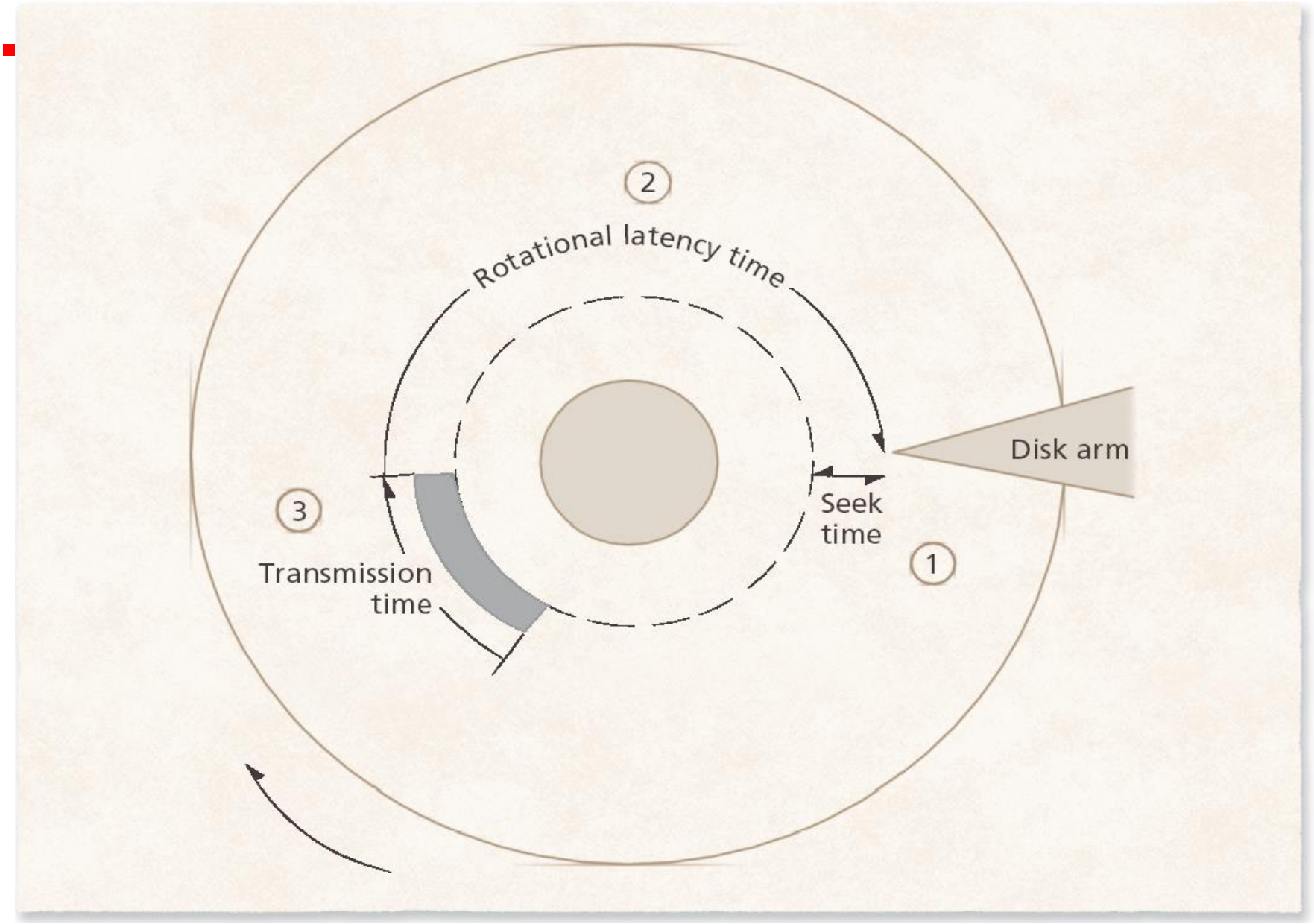


Figure 12.4 Components of a disk access.

Physical disk organization

- To read or write, the disk head must be positioned on the desired track and at the beginning of the desired sector
- **Seek time** is the time it takes to position the head on the desired track
- **Rotational delay** or **rotational latency** is the additional time its takes for the beginning of the sector to reach the head once the head is in position
- **Transfer time** is the time for the sector to pass under the head

12.5 Disk Scheduling Strategies

- Three criteria to measure strategies
  - Throughput
    - Number of requests serviced per unit of time
  - Mean response time
    - Average time spent waiting for request to be serviced
  - Variance of response times
    - Measure of the predictability of response times
- Overall goals
  - Maximize throughput
  - Minimize response time and variance of response times

First-Come-First-Served (FCFS) Disk Scheduling

- FCFS scheduling: Requests serviced in order of arrival
  - —Advantages
    - Fair
    - Prevents indefinite postponement
    - Low overhead
  - —Disadvantages
    - Potential for extremely low throughput
      - + FCFS typically results in a random seek pattern because it does not reorder requests to reduce service delays

FCFS

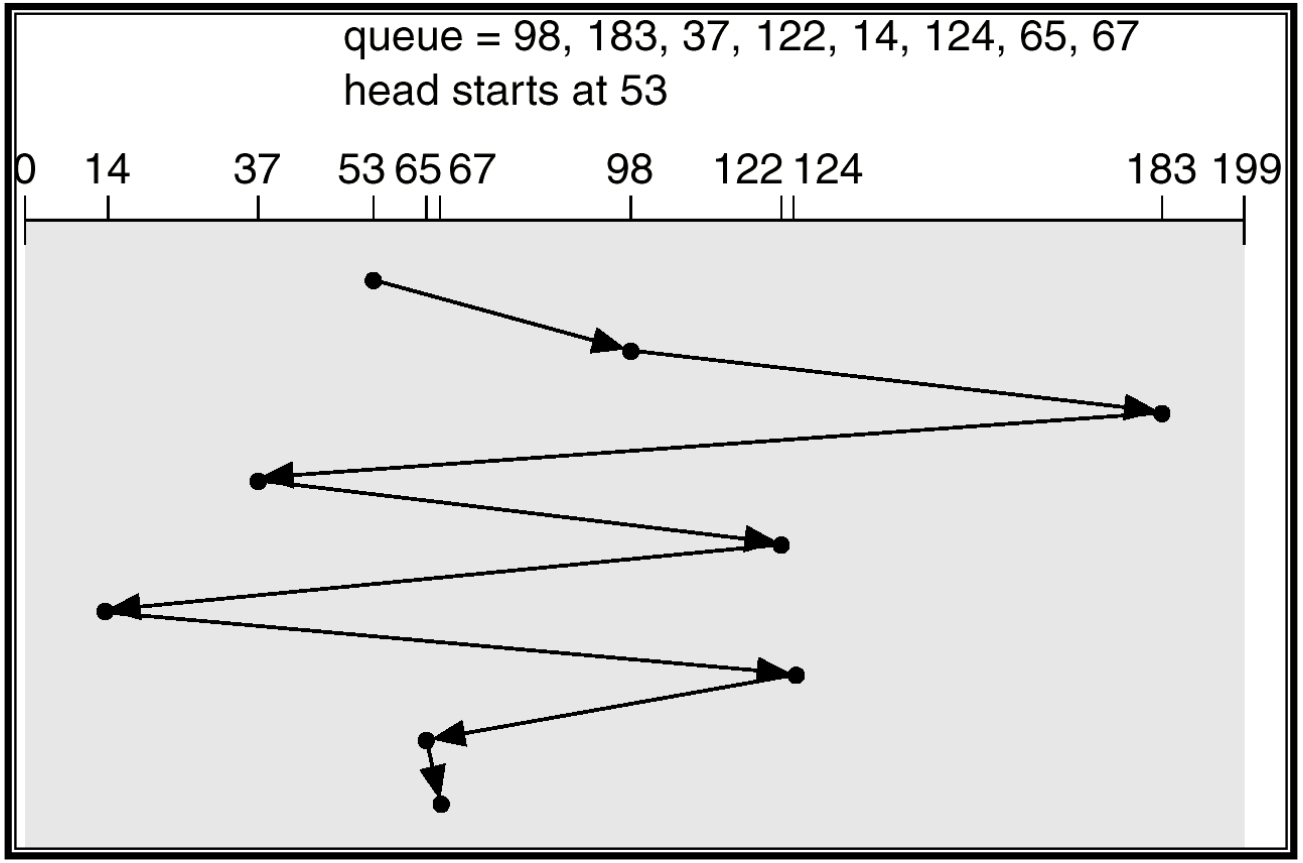


Illustration shows total head movement of 640 cylinders.

Disk Scheduling (Cont.)

- 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

- The disk head is initially at 53, it will move from 53 to 98, then 98 to 183, 183 to 37, 37 to 122, 122 to 14,124 to 65 and finally 67
- Total head movement= (98-53)+(183-98)+(183-37)+(122-37)+(122-14)+(124-14)+(124-65)+(67-65)

= 45+85+146+85+108+110+59+2

=640 tracks

- Average head movement (seek length)= Total no. of head movement/ no. of request.

- =640/8=80 tracks
- Assuming seek rate is 5ms
- Seek time=total head movement*seek rate
- =640*5ms
- =3200ms

Disk scheduling policies

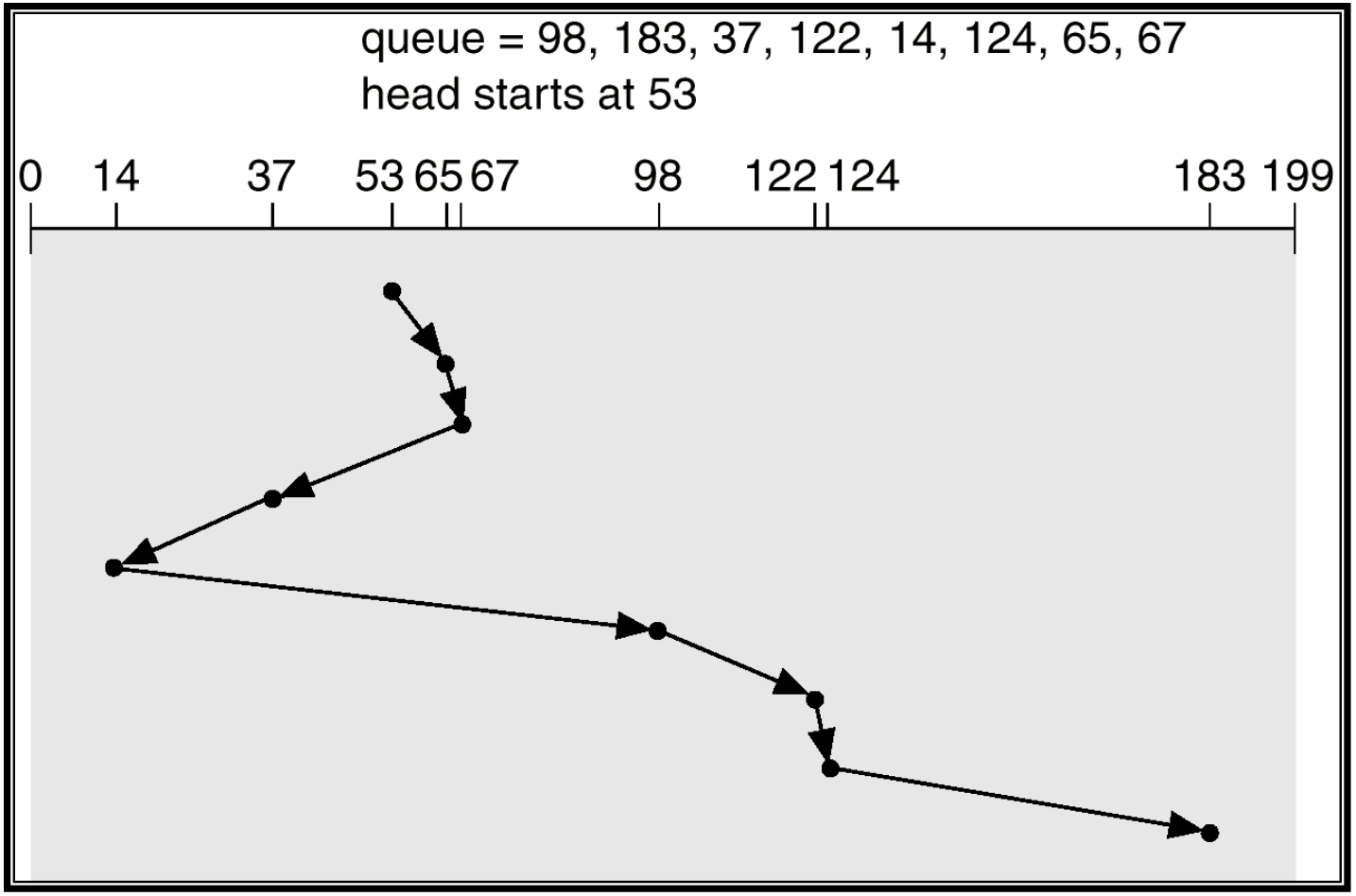
- SSTF (Shortest Service Time First)
  - —From requests currently in the queue, choose the request that minimizes movement of the arm (read/write head)
  - —Always chooses minimum seek time
  - —Resolves ties in a fair manner

    (both inward and outward)
  - —Doesn’t guarantee minimum total arm movement
  - —Starvation possible

SSTF

- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.
- Illustration shows total head movement of 236 cylinders.

SSTF (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

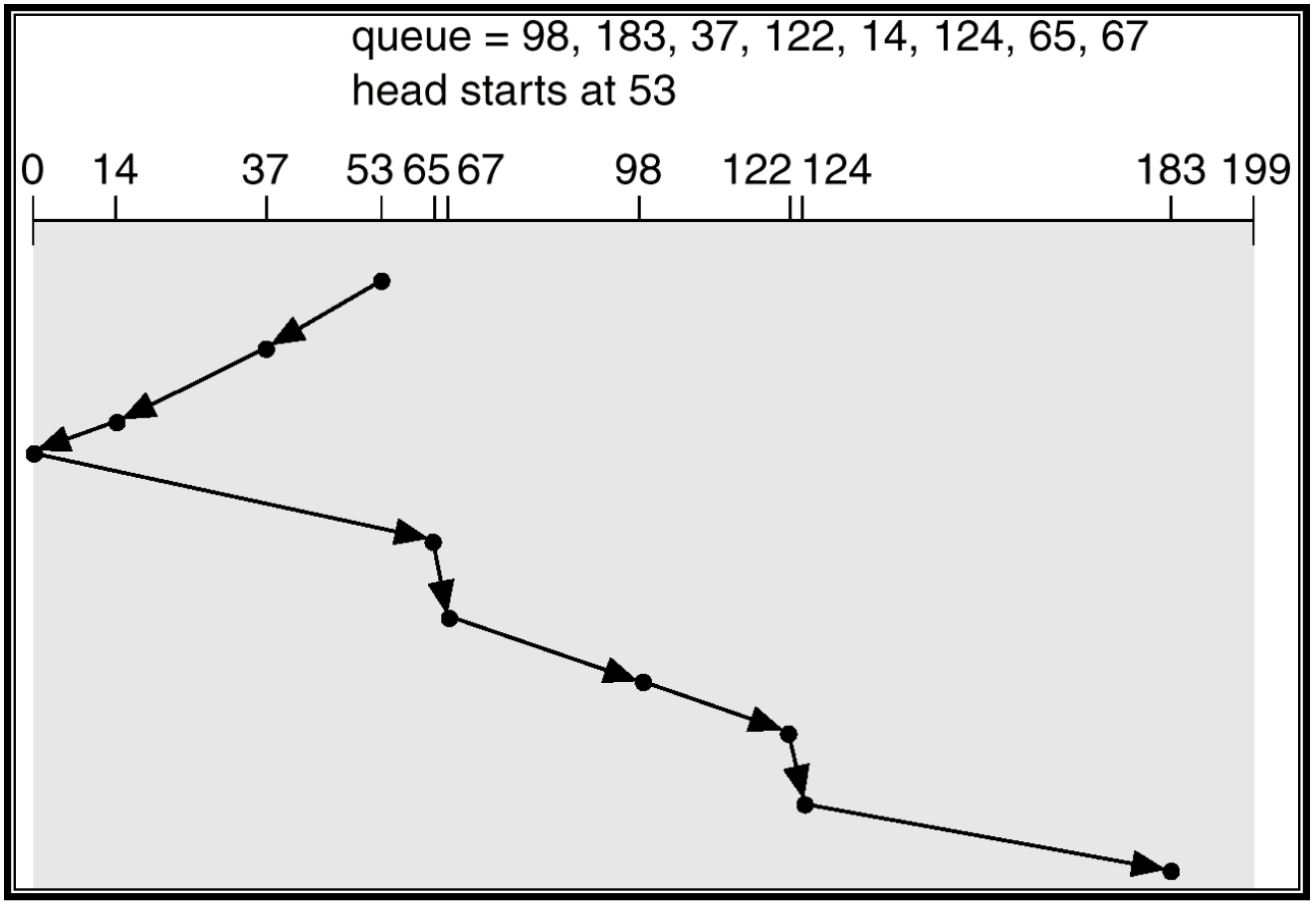
- The disk head is initially at 53, it will move from 53 to 65, then 65 to 67,67 to 37,37 to 14, 14 to 98, 98 to 122, 122 to 124 and finally 183
- Total head movement=(65-53) + (67-65)+(67-37)+(37-14)+(98-14)+(122-98)+(124-122)+(183-124)
- 12+2+30+23+84+24+2+59
- 236tracks

- Avg head movement(seek length)=236/8=29.5 tracks
- Assuming seek rate is 5ms
- Seek time=total head movement*seek rate
- =236*5=1180ms

SCAN scheduling

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- Illustration shows total head movement of 208 cylinders.

SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Disk scheduling policies

- SCAN
  - Arm moves in one direction only until it reaches the last request in that direction
  - Then the arm reverses and repeats
  - Avoids starvation
- C-SCAN
  - Like SCAN, but in one direction only
  - Then returns arm to the opposite side and repeats
  - Reduces maximum wait in the queue near the disk edge