File Systems

Topics

- Concept, support
- Access and Allocation methods
- Directory system
- File protection
- Implementation issues

Introduction

- The most visible aspect of an OS
- Provides the mechanism for on-line storage and access of data and programs of OS and user
- Consists of three distinct parts :
  - Collection of Files : each storing related data
  - Directory structure : organizes and provides information about the files in the system
  - Partitions (optional) : physical or logical separation of large collection of directories

File Concept

- A logical storage unit that the OS abstracts from the physical properties of its storage devices
- A named collection of related information that is recorded on secondary storage
- The smallest allotment of logical secondary storage

File Attributes

- Name: only information in human readable form
- Identifier: unique number that identifies the file in the file system
- Type: needed in systems that support different types
- Location: pointer to a device and to the location of the file on that device
- Size: the current size of the file
- Protection: access control information
- Time, date and user identification: useful for protection, security and usage monitoring

File Operations

- OS provides different system calls to perform file operations:
  - Creating
  - Writing
  - Reading
  - Deleting
  - Truncation
  - Repositioning/seek
- Other operations can be implemented as a combination of the above calls (renaming, copying, changing attributes)
- To avoid constant directory search of frequently used files, OS maintains an open-file table
- To provide protection, OPEN call can accept access mode that is checked against file permissions
- System like Linux maintains two level of internal tables :
  - Per-process table : tracks all the files that a process has open
  - System-wide table : contains process independent information and count

File types

- OS dependent
- Usually implemented by including the type as part of the file name; name and extension
- Windows and Apple Macintosh allows creator attribute to be associated with the file
- Linux system supports 7 file types
- Extensions also help the programmer to identify a file

File Types

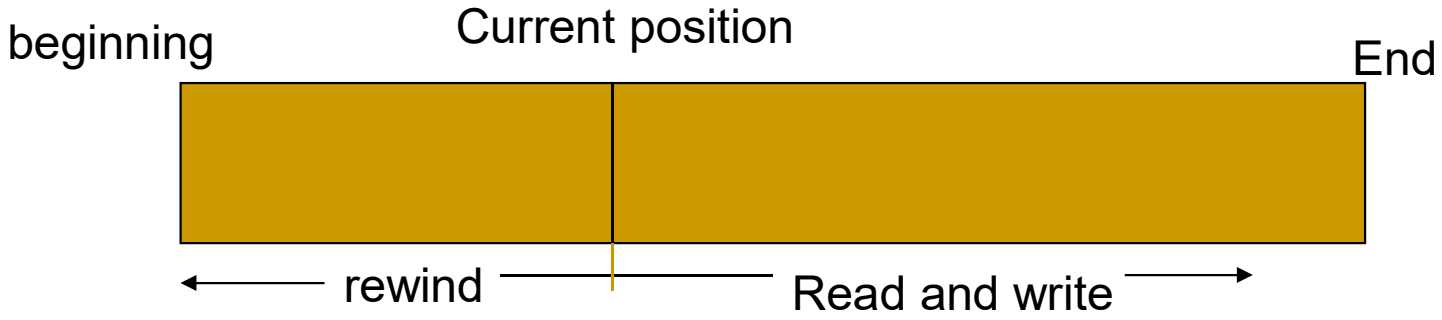
file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	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	binary file containing audio or A/V information

File Access Methods

- Some systems support only one method
- Systems that support more than one method; choice of the method depends on the application; a major design issue
- Different access methods :
  - Sequential access
  - Direct access
  - Indexed access

Sequential Access

- Most common method of access; information processed one record after other in order
- Used by editors and compilers
- A pointer is maintained that is advanced to next block after every Read/Write operation
- Provides facility to reset the pointer to the beginning and to skip forward or backwards n records.



- Based on a tape model of a file, works equally well for random access devices

Direct Access

- Based on disk model of a file
- File is viewed as a numbered sequence of blocks or records
- Allows arbitrary records to be read or written
- File operations need to be modified to include the block number as a parameter.
- Block numbers are relative to the beginning of the file (0 or 1)
- Some systems require the file access to be defined at the time of creation
- Sequential access can be simulated on a direct access file while the reverse is very clumsy

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

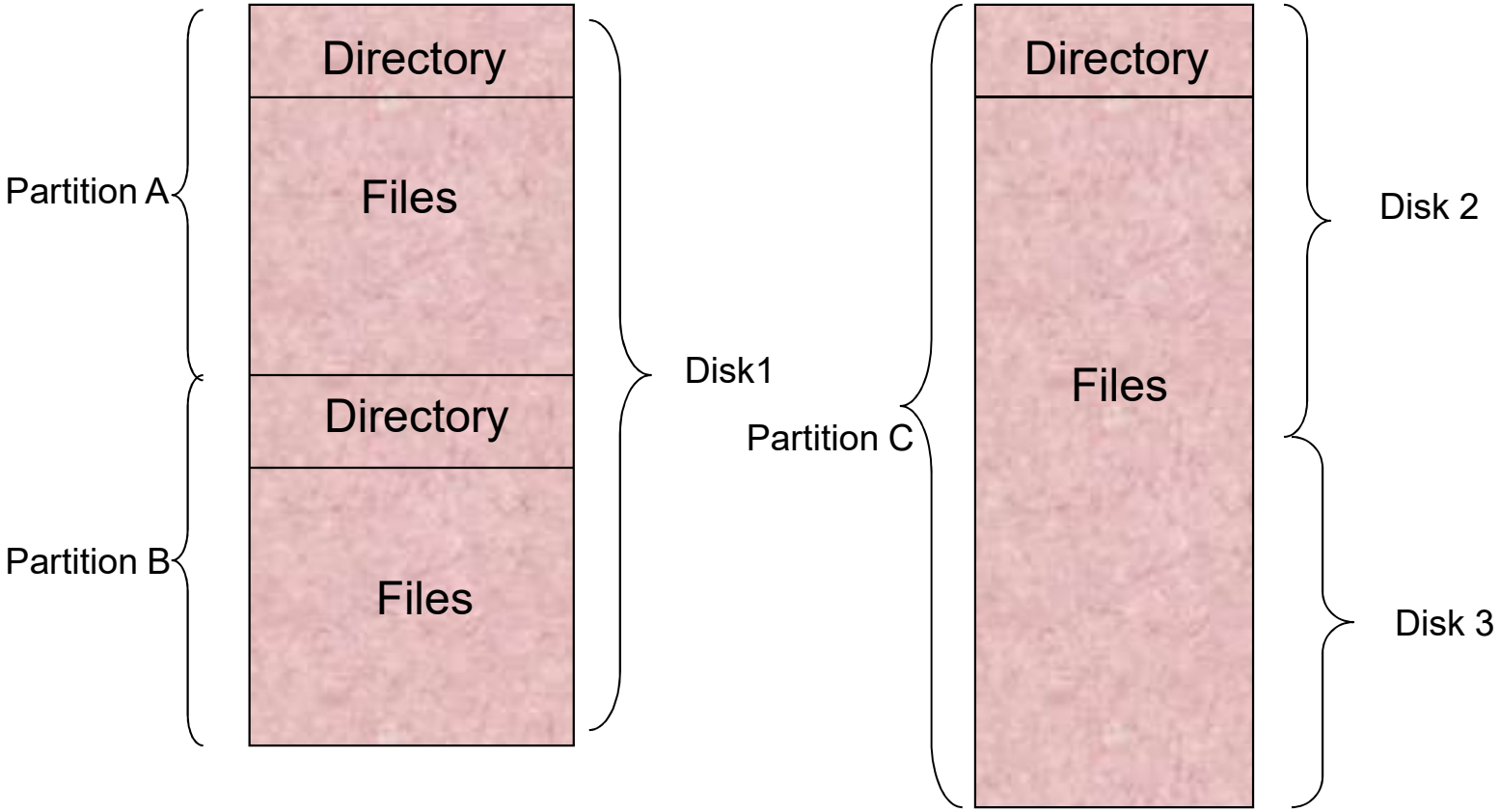
Indexed Access

- Built on top of Direct access
- An index file is maintained that points to the various blocks
- To find a record in the file, first search the index, and then use the pointer to access the file directly to get the desired record
- To handle large index files , an index for the index file is created; the primary index file contains pointer to secondary index which in turn point to the actual data e.g. IBM's indexed sequential access method (ISAM)

File Organization

- To manage huge amount of data, it needs to be organized
- Organization is usually in two parts :
  - Partitions: can provide several separate areas within one disk or can group disks into one logical structure
  - Device directory or volume table of content: records information such as name, location, size and type for all the files in a single partition

A Typical File System Organization



File Organization as Directory

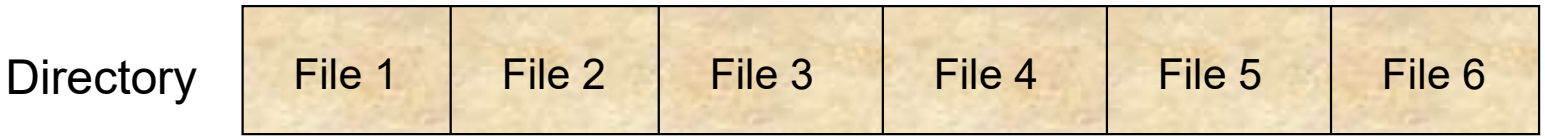
- Organization of files in directories as the benefits of:
  - **Efficiency** – locating a file quickly.
  - **Naming** – convenience of users.
    - Two users can have same name for different files.
    - The same file can have several different names.
  - **Grouping** – logical grouping of files by properties, (e.g., all Java programs, all games, …)

Operations performed on a Directory

- Search for a file
- Create a file
- List a directory
- Rename a file
- Traverse the file system

Single-Level Directory

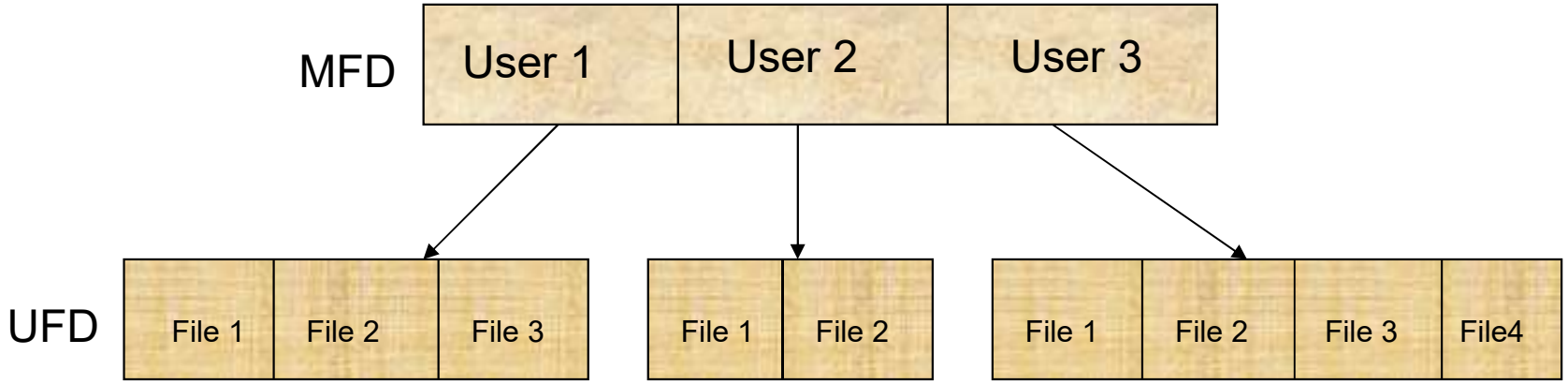
- All files contained in the same directory
- Easy to support and understand
- Significant limitations with increase in directory size
- Files within same directory must have unique names



Two-Level Directory

- Each user has his/her own user file directory (UFD)
- A Master file directory (MFD) keeps the account of all users and is indexed by user name or account number.
- Filenames can be reused by different users
- Effectively isolates one user form another; advantageous when users are completely independent but does not allow for cooperation among users

Two-Level Directory

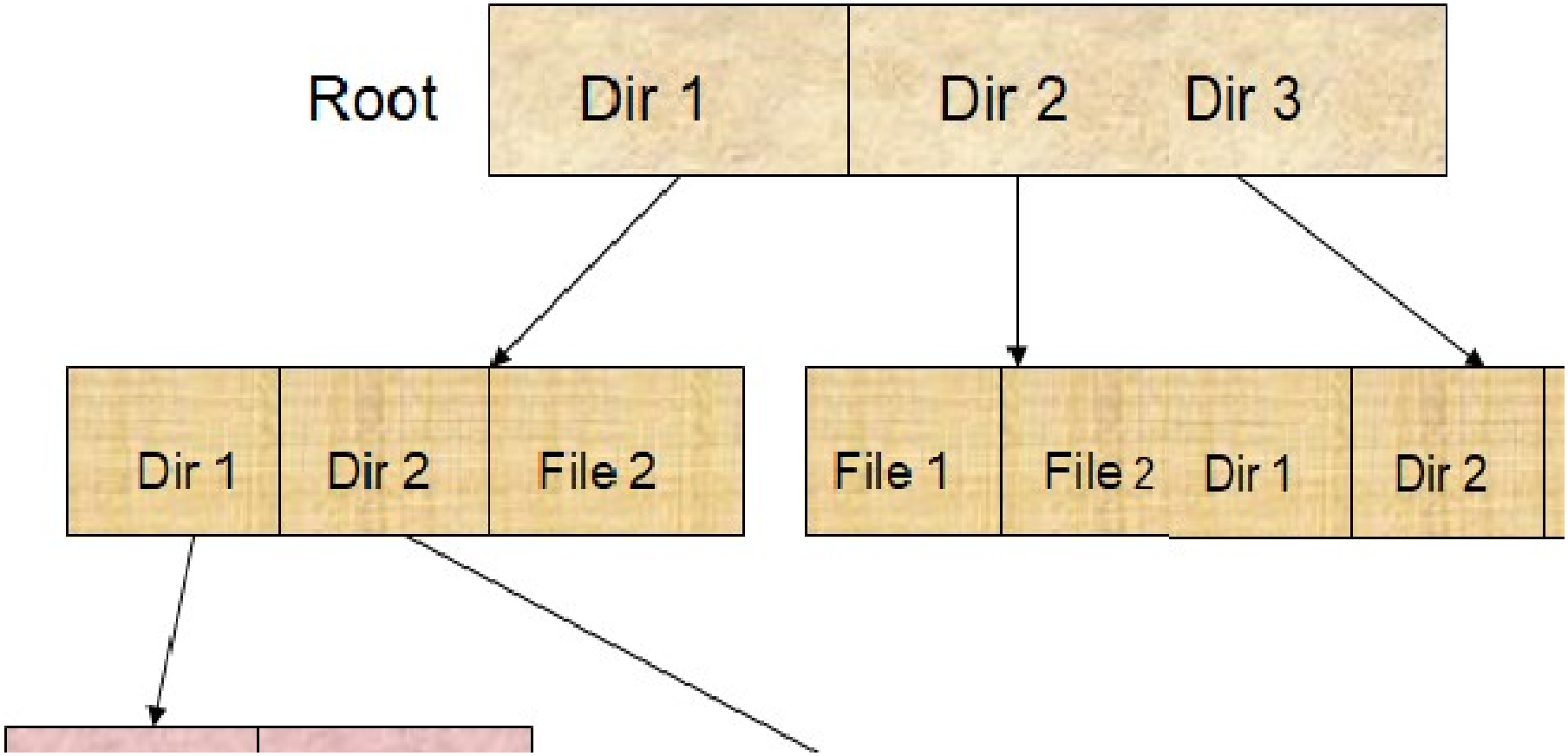


- Similar to a tree structure with MFD as the root and UFDs as its direct descendants
- A path is identified by the user name and the file name
- Every system uses a separate syntax for specifying the path name
- To allow sharing of system files, a separate user directory is defined
- A file is searched according to the search path

Tree- Structured Directory

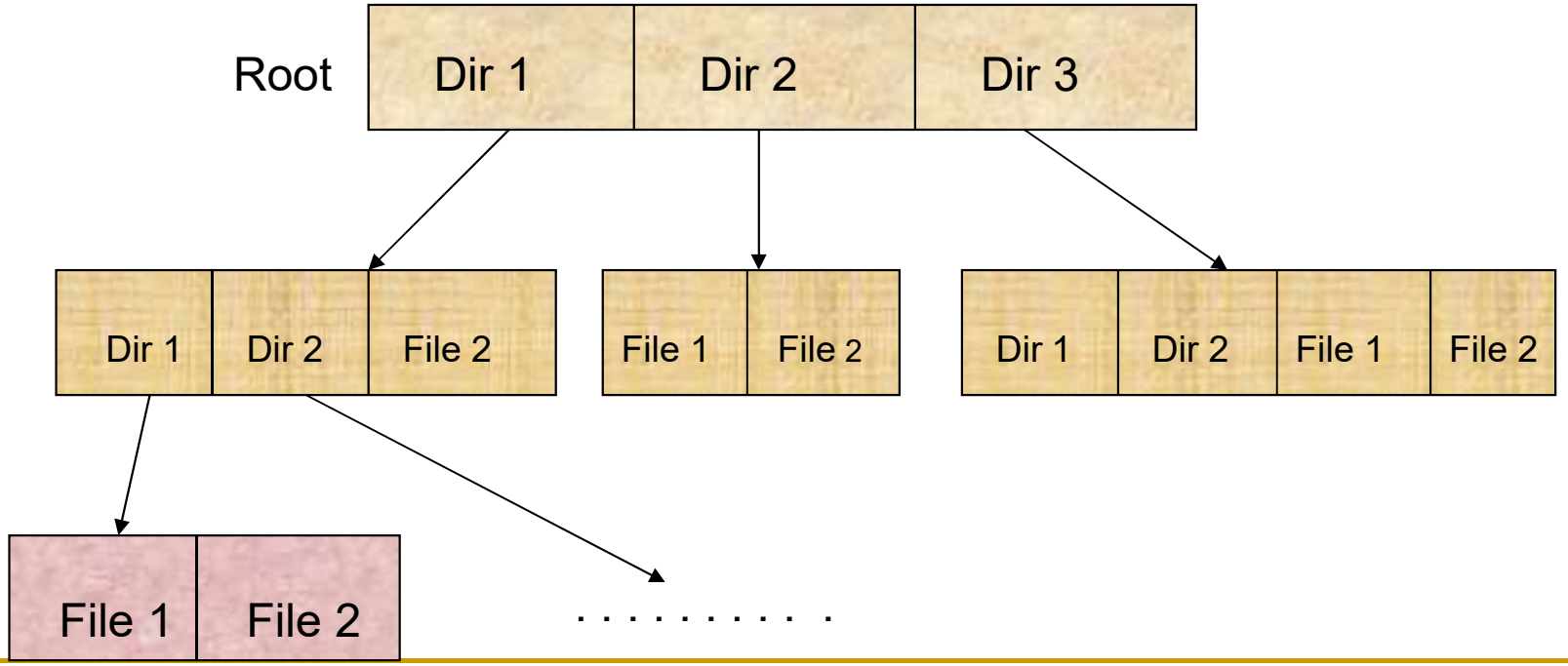
- Extension of two-level directory structure
- Organized as a tree like structure with root as the root directory
- A directory can contain a set of files or subdirectories. A directory is treated as a special file
- A path name is the path from the root through all the subdirectories, to a specified file
- One bit in each directory entry defines the entry as a file(0) or as a subdirectory(1)
- Path name for a file can be specified as an Absolute path or as a Relative path
- Allows the user to impose a structure on his files and reduces searching time

Tree- Structured Directory



Tree- Structured Directory

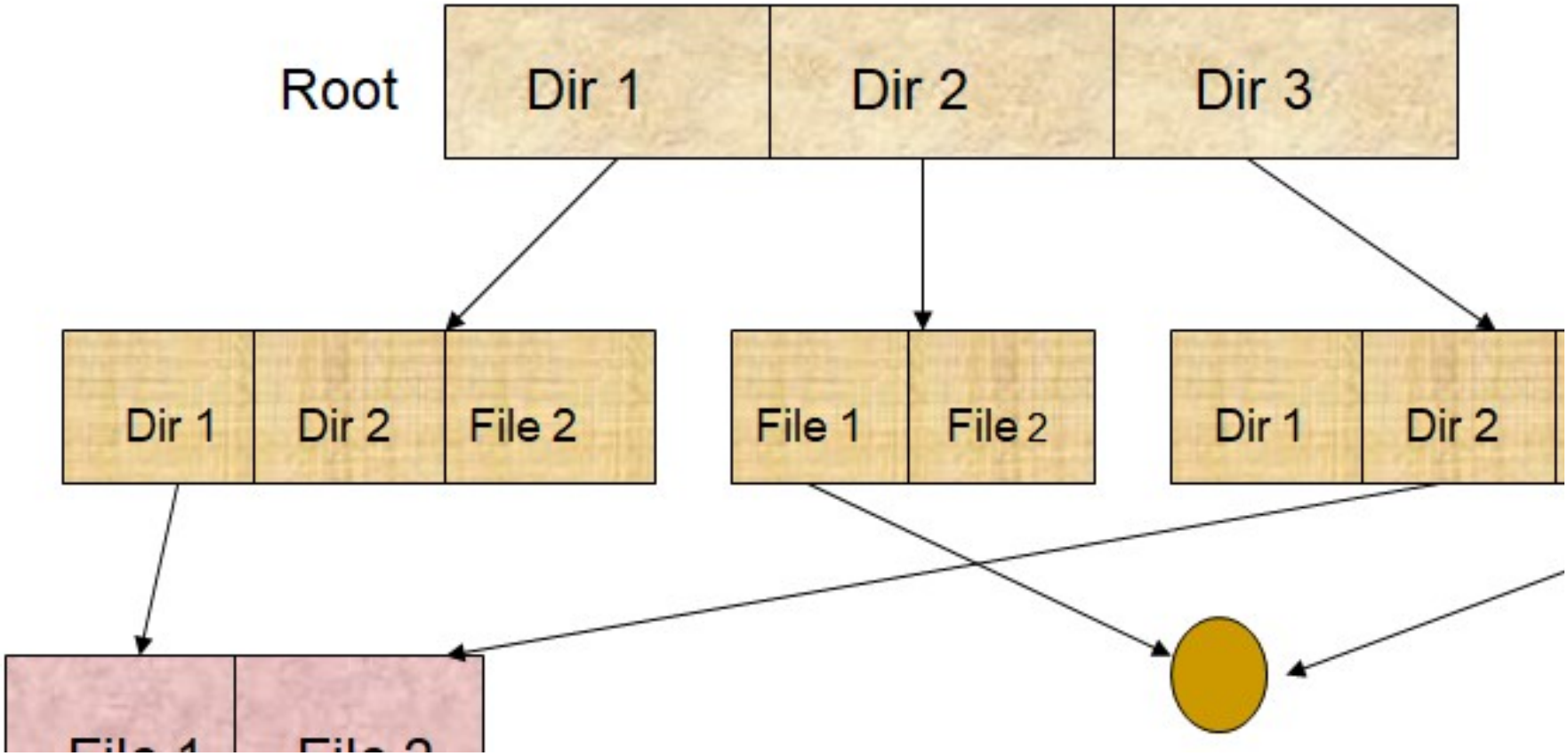
- An interesting policy decision is to handle directory deletion ; DOS requires the directory to be empty, Linux **rm** command recursively deletes all the subdirectories
- By specifying the proper path other users file can be accessed



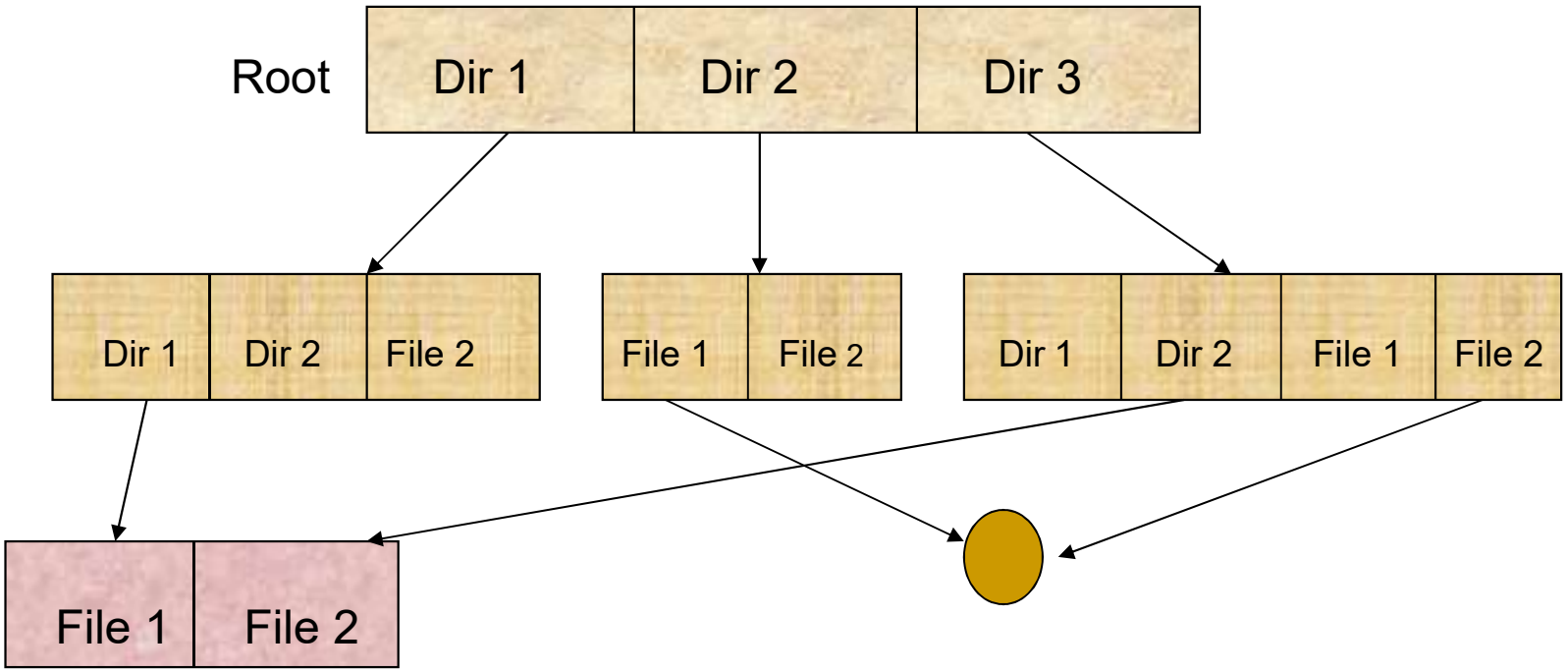
Acyclic-Graph Directory

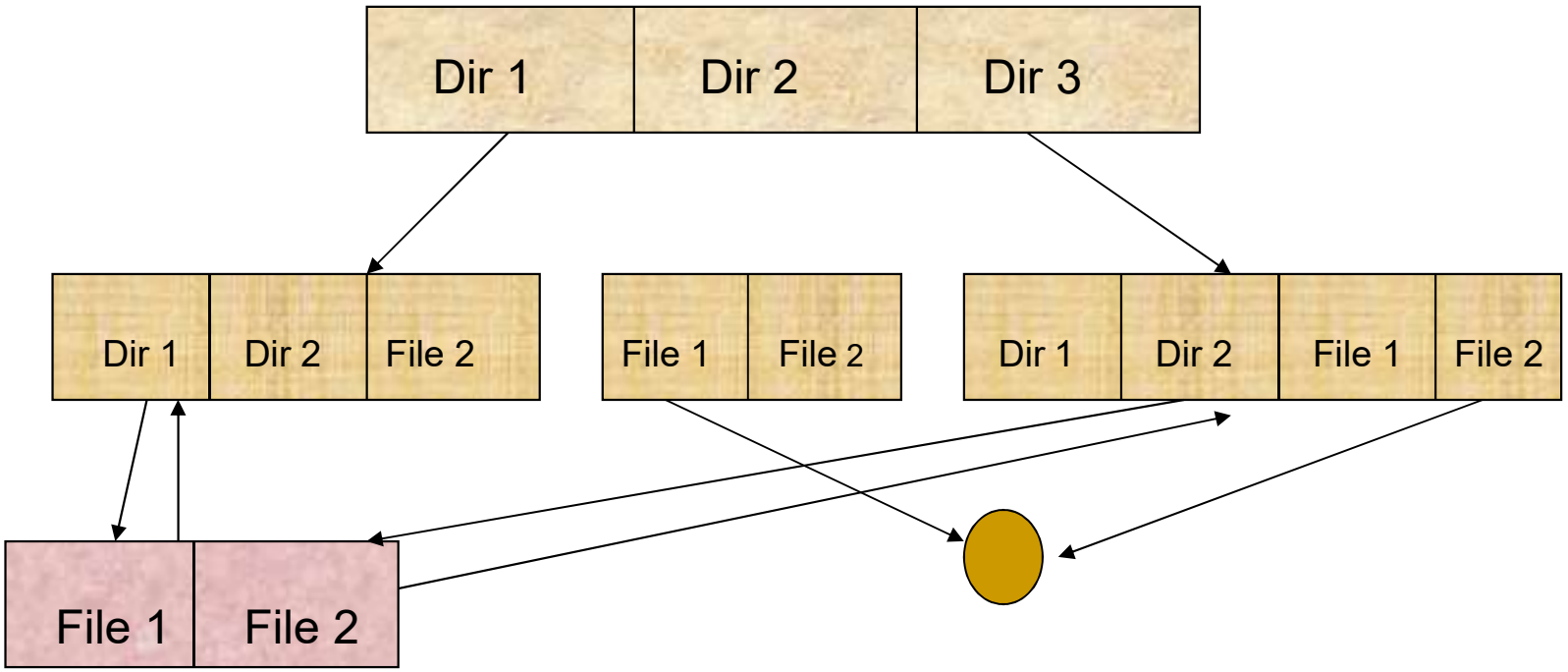
- A generalization of tree- structured directory scheme that allows for sharing of files and directories
- A shared file is different from copy of the file
- In Unix/Linux sharing is implemented using LINKS
- Another way is to maintain duplicate copies; a major problem is to maintain consistency
- More flexible but complex than simple tree structure; a file can have multiple absolute paths, same file can be traversed more than once
- Another major problem is deletion of the shared file

Acyclic-Graph Directory



Acyclic-Graph Directory




Dir 1
Dir 2
Dir 3
Dir 1
Dir 2
File 2
File 1
File 2
Dir 1
Dir 2
File 1
File 2
File 1
File 2

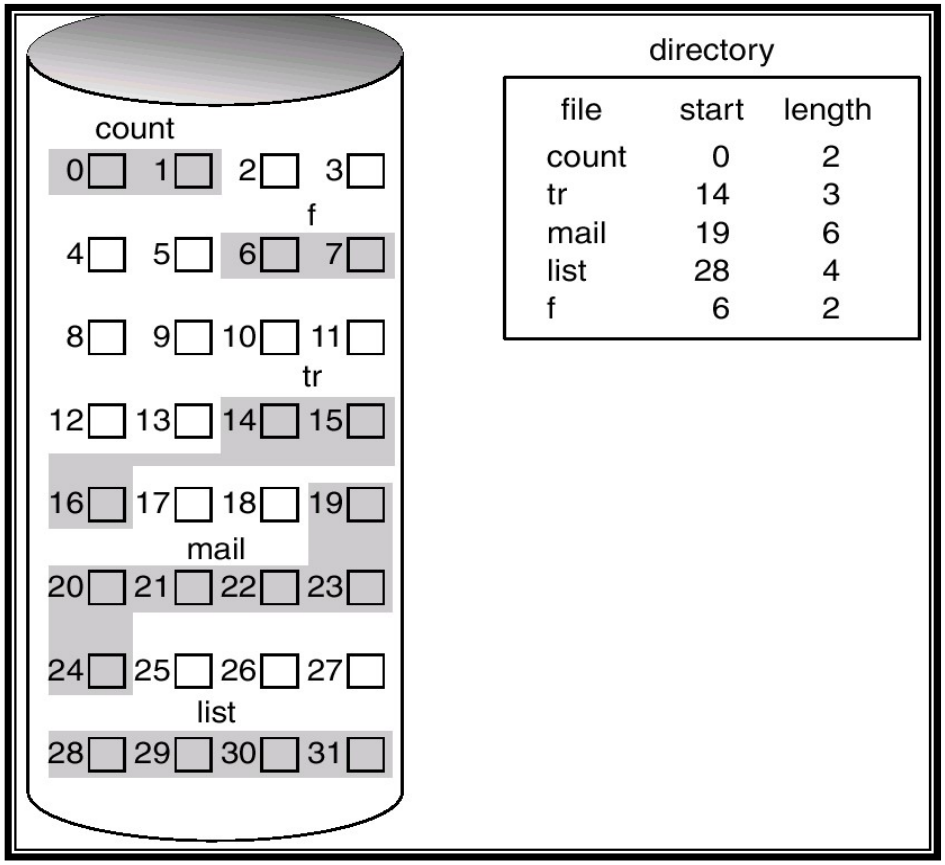
Disk Allocation

Contiguous Allocation

- Each file occupy a set of contiguous blocks on the disk
- Accessing block b+1 after block b normally requires no head movement; if head movement is required it is only one track
- Number of disk seeks are minimal as is the seek time when a seek is finally required
- Allocation of a file is defined by the disk address and length of the first block. If the file is n blocks long and starts at location b, then it occupies blocks b, n+1, ….b+n-1
- The directory entry of each file indicates the address of the starting block and the length of the area allocated for this file
- Both, sequential and direct access are supported by contiguous allocation
- IBM VM/CMS OS uses this method because of good performance

Contiguous Allocation

- Leads to External fragmentation; overhead of compaction
- Space requirement must be know a priori before allocation
- Over allocation of file leads to Internal fragmentation



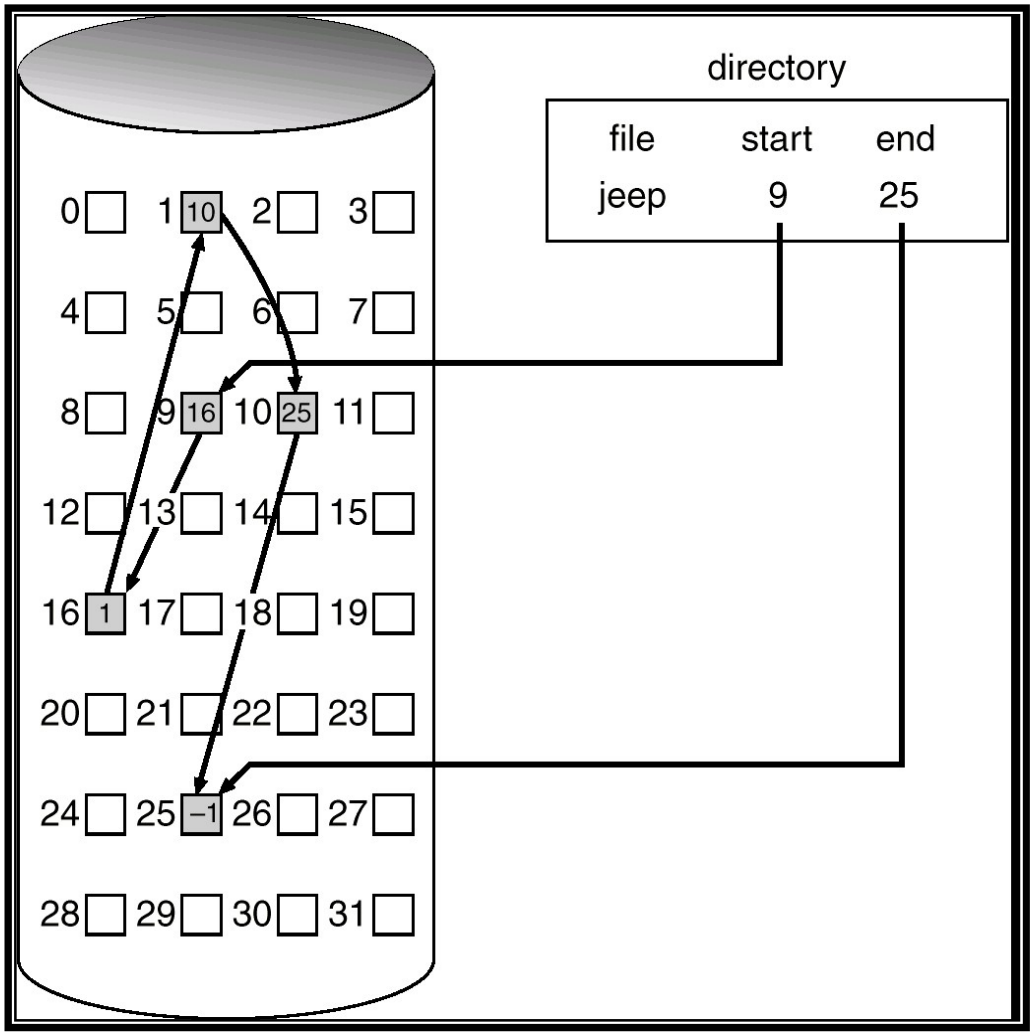
Linked Allocation

- Each file is a link list of disk blocks; the blocks may be scattered anywhere on the disk, each block contains a pointer to the next block
- The directory contains a pointer to the first and last blocks of the file
- To create a new file, simply create a new entry in the directory pointing to the first block of the file; this pointer is initially nil, signifying the file size to be 0
- A write to the file causes a free block to be found via the free-space management, this new block is written to and is linked to the end of the file
- A file is read by following the pointers from block to block
- There is no external fragmentation, size of file need not be declared at the time of creation, file can grow as long as free blocks are available, compaction is not required

Linked Allocation

- Supports only sequential access
- Pointers consumes space; if a pointer requires 4 bytes out of a 512 byte block, 0.78% of disk space is used by pointers
- Reliability is another problem; files are linked by pointers, damaged or lost pointer pose problems; doubly linked list can be a solution but require even more space for pointers
- One solution is to collect multiple blocks in cluster; can lead to internal fragmentation

Linked Allocation

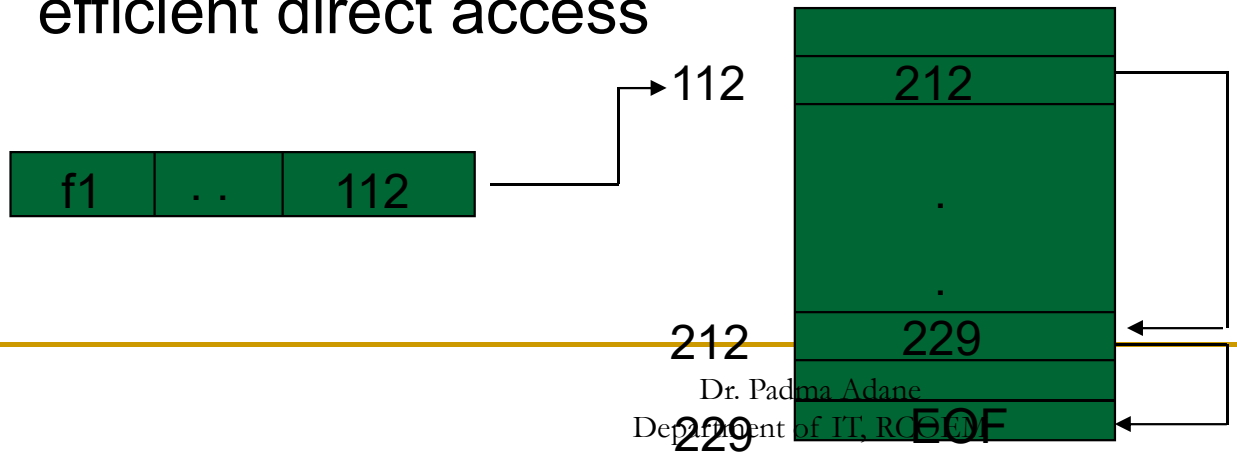


A variation of the linked allocation method : FAT

- Used by MS/DOS and OS/2 Operating systems
- A section of the disk at the beginning of each partition is set aside to contain the table; the table has one entry for each block and is indexed by block numbers
- The directory entry contains the block number of the first block of the file; the FAT contains the block number of the next block in the file, this chain continues until the last block which has a special end-of-file entry
- Unused block are indicated by a 0 table value
- Allocating a new block to a file requires locating a block with 0 entry and replacing the previous end-of-file value with the address of this new block, the 0 is then replaced with the end-of-file

FAT

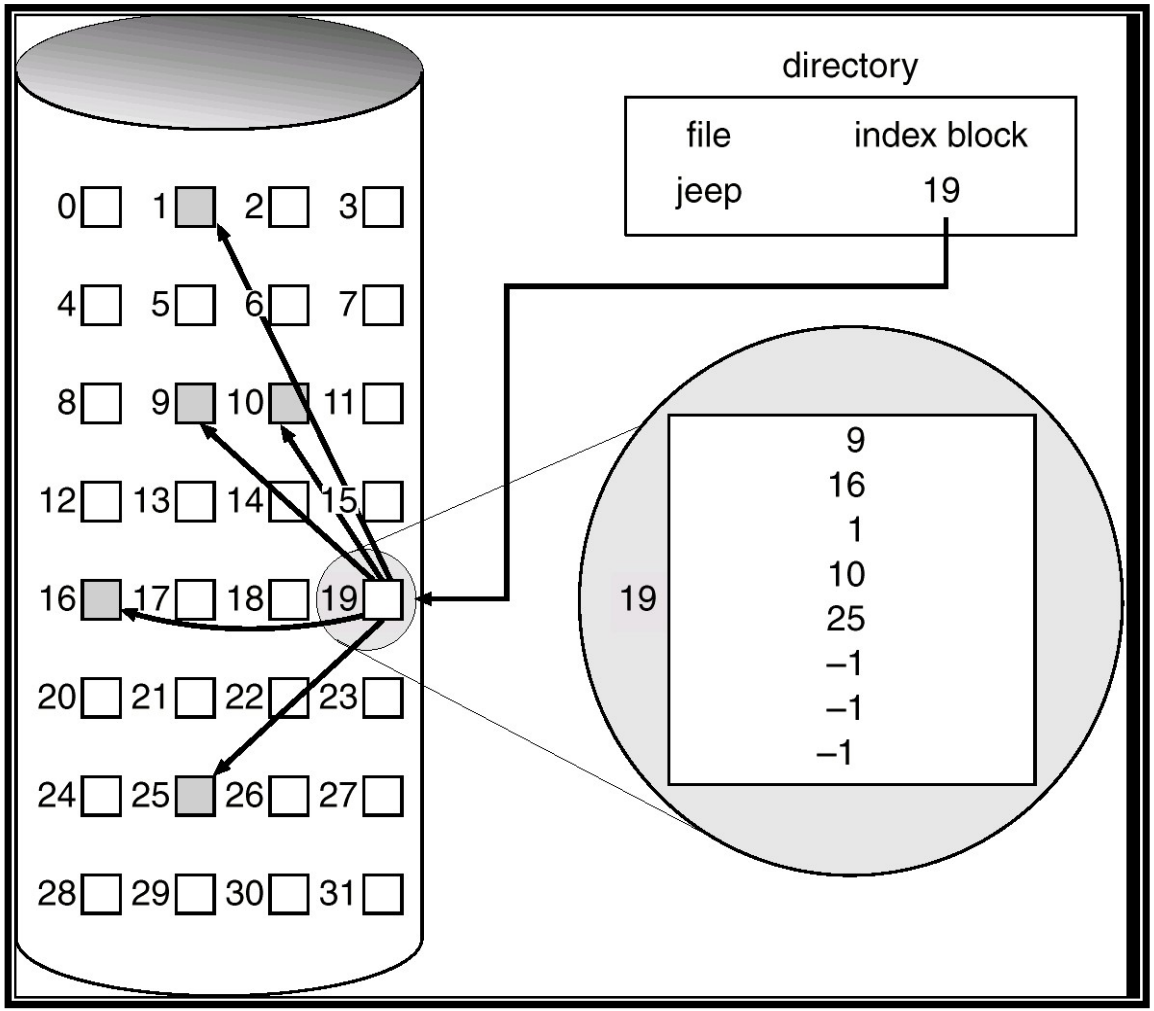
- Requires significant number of disk head seeks; the disk head must move to the start of the partition to read the Fat and find the location of the block in question, then move to the location of the block itself
- In worst case, both moves occur for each block
- Random access time is improved because disk head can find the location of any block by reading the information in the FAT
- In absence of a FAT, linked allocation cannot support efficient direct access



Indexed Allocation

- Each file has its own index block, which is an array of disk-block addresses
- The directory contains the address of the index block, the i^{th} entry in the index block points to the i^{th} block of the file
- When the file is created, all pointers in the index block are set to nil; when the i^{th} block is first written, a block is obtained and its address is put in the i^{th} index block entry; reading is done by using the i^{th} index block entry to find and read the desired block
- Supports direct access, without suffering from external fragmentation
- Pointer overhead is greater than that of Linked allocation

Indexed Allocation



Performance

- Allocation methods vary in their storage efficiency and data block access time; both are important criteria in selecting the proper methods for an OS to implement
- Systems with contiguous allocation support both sequential as well as direct access; requires only one access to get the desired block
- Systems with linked allocation can support only sequential access
- Some systems support direct access files using contiguous allocation and sequential access using linked allocation; requires access type to be declared at the time of file creation

Performance

- Systems using index allocation are more complex; the performance of index allocation depends on the index structure, size of the file and on the position of the block desired
- Some systems combine contiguous allocation and indexed allocation; contiguous allocation used for small files and switch automatically to index allocation as the file size grows

Free Space Management

- Since the disk space is limited, the freed space from deleted files needs to be reused
- To keep track of the free space, the system maintains a free space list that records the free blocks in the disk
- When a request for disk allocation is received, this free list is searched and the required number of blocks are allocated (provided space is there)

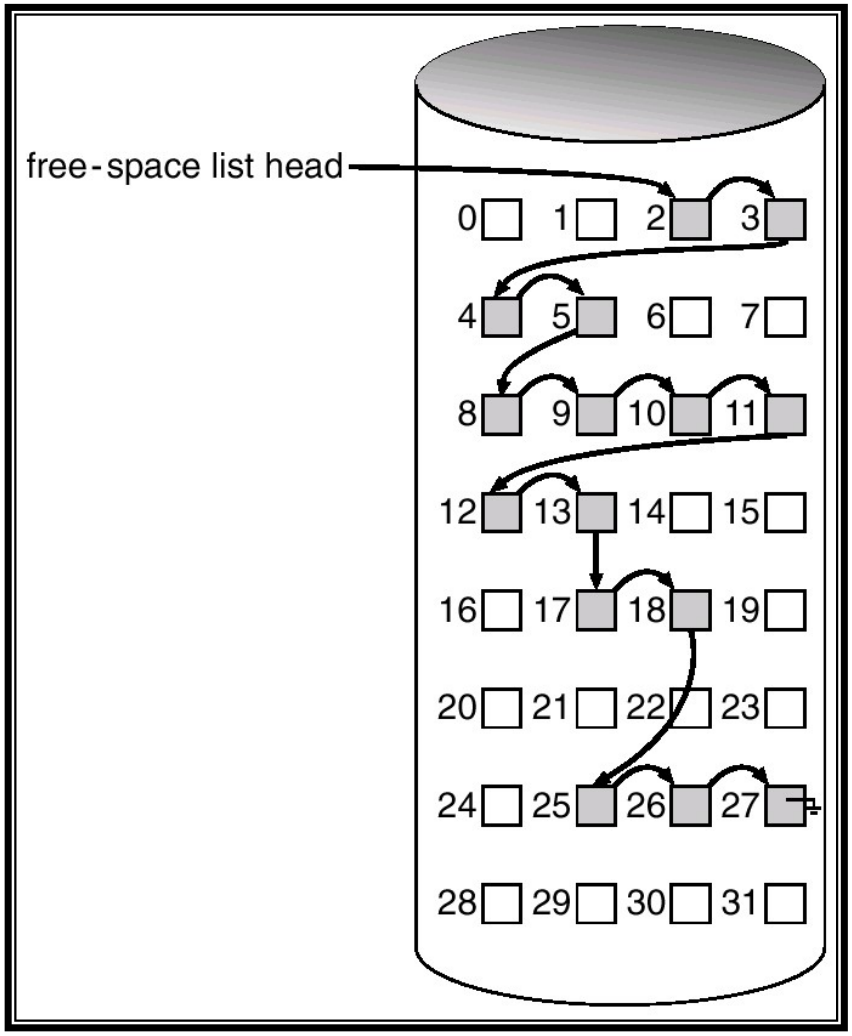
Bit Vector

- In a bit map or bit vector, each block is represented by 1 bit; free block is indicated by 1 and allocated block by 0
- Example : 00000000 01111000 11111000
- Relatively simple and efficient in finding the first free blocks, or n consecutive free blocks on the disk
- Apple Macintosh system uses the method; to find the first free blocks, it checks sequentially each word to see whether the value is 1, a 0 valued word has all 0 bits and represents a set of allocated blocks; the first non-0 word is scanned for the first 1 bit, which is location of the first free block
- The block number is calculated as

(no. of bits per word) * (no. of 0-valued words) + offset of first 1 bit

- Unless entire vector kept in memory, quite inefficient; possible only for small disks

Linked List


free-space list head
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

Grouping

- Stores the address of n free blocks in the first free bloc; the first n-1 of these are actually free
- The last block contains the address of another n free blocks and so on
- Addresses of a large number of free blocks can be found quickly

Counting

- With contiguous allocation, contiguous blocks are allocated and freed
- Keeps the address of the first free block and the number n of free contiguous blocks that follow the first block
- Each entry consists of a disk address and a count
- Requires more space than a simple disk address, with count greater than 1,the overall list is short

File Protection

- Systems needs to ensure the safety of the information kept in computer system from Physical Damage (Reliability) Improper Access (Protection)
- Reliability : File systems can be damaged by hardware problems, power surges of failures, head crashes, dirt, temperature extremes, and vandalism. They may be deleted accidentally or bugs in the system software may cause the data to be lost
- Reliability is generally provided by duplicate copies of files

- Improper Access :Systems that allow users to access other users file must provide for controlled access
- Access is permitted or denied depending on several factors, one of which is the type of access. Several different types of operations needs to be controlled :
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Access Control

- The most common approach is to make access dependent on the identity of the user
- A general scheme is to associate an Access Control List (ACL) with each file or directory; enables complex access methodologies, however, the list could be too lengthy , requires a list of users in the system and directory entry to be of variable size

Access Control

- Another way is to use a condensed access list; classify the users as USER, GROUP and OTHER (e.g. Unix)
- This scheme requires the granting of permissions and access list to be controlled
- To handle special cases, the two cases can be combined with priorities defined

Password

- Passwords can be associated not only at the login level but also at file and directory level

File Implementation

- In order to manage all the files, the file manager maintains many data structures, some are maintained on-disk, while others are in memory
- On-disk structures :
  - Boot control block : contains information needed for booting the system, typically the first block of the partition. In UFS referred as boot block , in NTFS referred as The partition boot sector
  - Partition control block: contains partition details such as the no. of blocks in the partition, size of blocks, free-block count, free block pointers, free FCB count, FCB pointers. In UFS referred as the super block and in NTFS referred as the Master File Table

- A directory structures that organizes the files
- File Control Block : it contains the file's details, including the size, ownership, permissions, location of the data block. In UFS referred as the inode, in NTFS this information is in the master file table

- In-memory structures :
  - Information about each mounted partition
  - Directory structure holding information of recently accessed directories
  - The system wide Open-file table containing the copy of FCB of each opened file
  - The per-process open-file table containing pointer to the appropriate entry in the system-wide open table

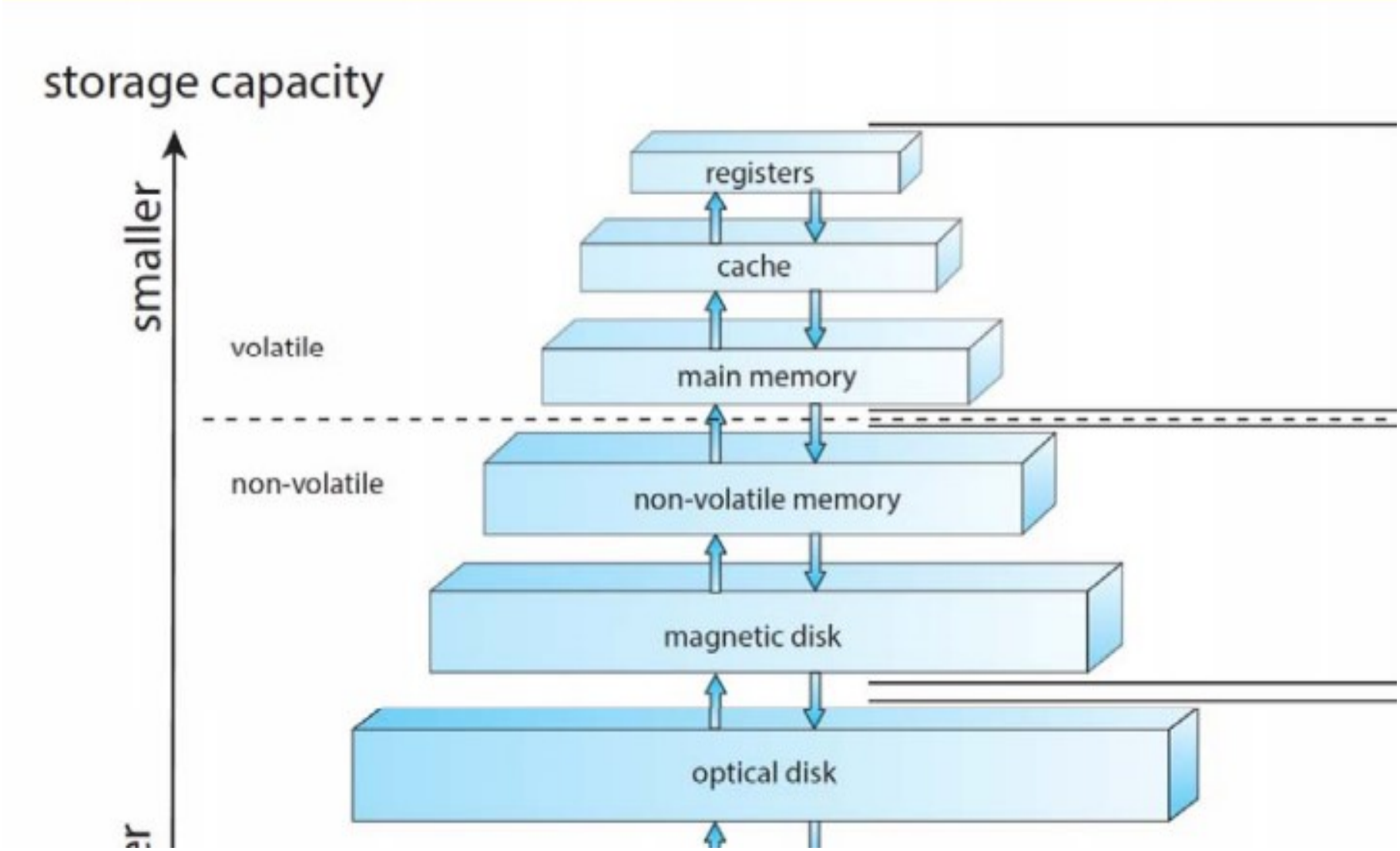
Implementing different file operations

- To create a file, application program calls the logical file system which allocates a new FCB, reads the appropriate directory into memory, updates it with the new file name and FCB and writes it back to the disk. The logical file system can call the file organization module to map the directory I/O into the disk block numbers , which are passed on to the basic file system and the I/O control system
- Once the file is created, it is ready for I/O. But before any operation is performed it must be opened. The open call passes a file name to the file system. The directory structure is searched for the given file name. Once the file is found, the FCB is copied into the system wide open-file table in memory

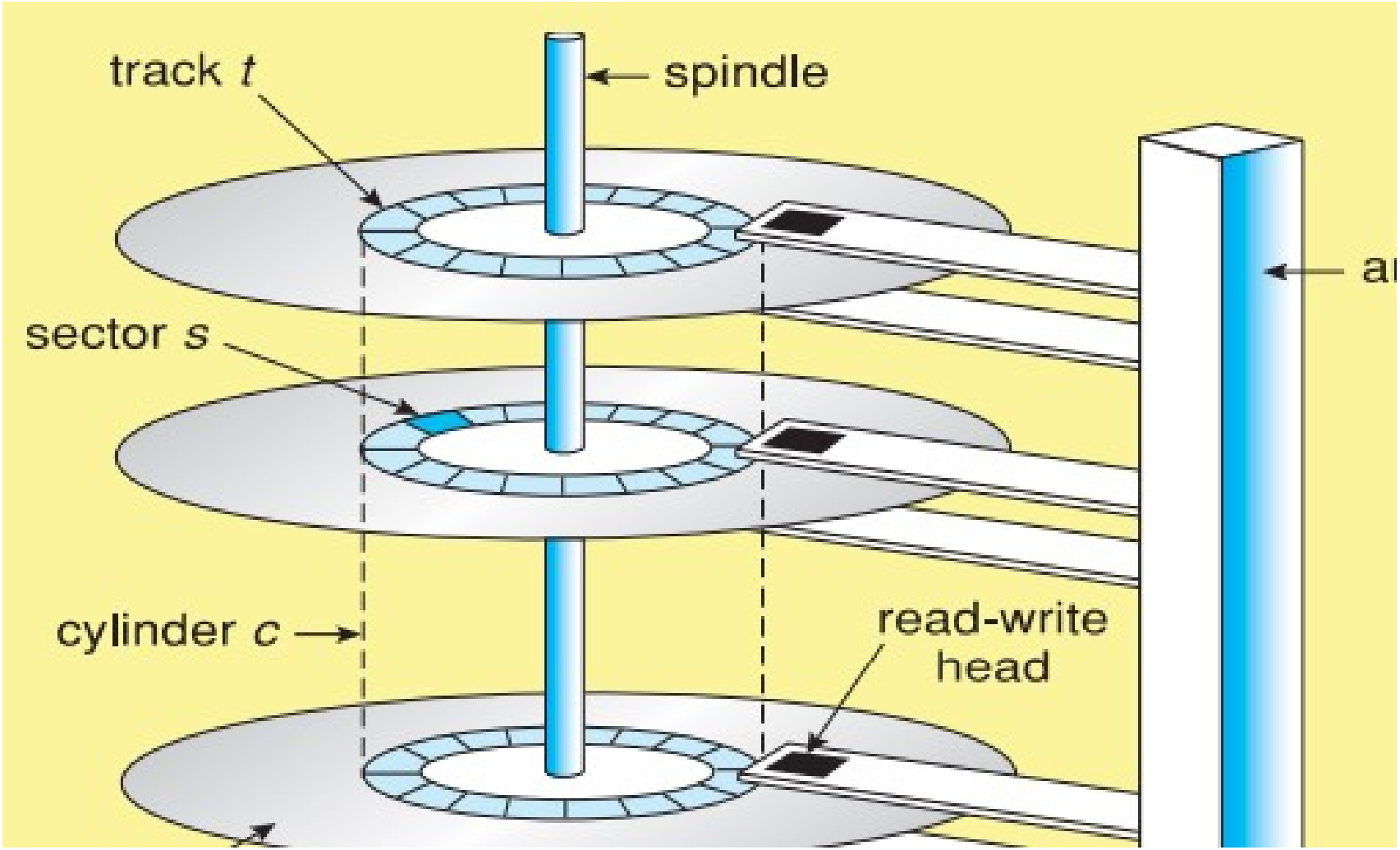
- Next, an entry is made in the per-process open file table, with a pointer to the entry in the system wide open file table and some other fields. The open calls returns a pointer to the appropriate entry in the per-process open file table. All file operations are the performed via this pointer. In Unix it is refers as the file descriptor while in Windows 2000 as a file handle. As long as the file is not closed, all file operations are done on the open file table
- When a process closes the file, the per-process table entry is removed, and the system wide table entry's open count is decremented. When all users closes the file, the updated information is copied back to the disk based directory structure and the system wide open file table entry is removed

Disk Scheduling

Storage Device Hierarchy



Hard Disk Drive Moving Head Mechanism



Disk Scheduling

- In multi programmed systems many processes generate request for I/O; the requests are usually made faster than the they can be serviced by the moving- head disks and hence waiting queues are build up for each device
- Disk scheduling involves careful examination of pending requests to determine the most efficient way to service the request
- A disk scheduler examines the positional relationships among waiting requests and reorder the queue so that the requests can be serviced with minimum mechanical movement

Disk Scheduling

- Most common type of disk scheduling algorithms aim for :
  - Seek optimization
  - Rotational optimization
- Disk speed has two parts
  1. Transfer Rate : rate at which data flow between the drive and the computer
  2. Positioning time/Random access : it consists of
     1. Seek time : time to move the disk arm to the desired cylinder
     2. Rotational Latency : time for the desired sector to rotate to the disk head
- Disk bandwidth : the total number of bytes transferred , divided by the total time between the first request of service and the completion of the last transfer

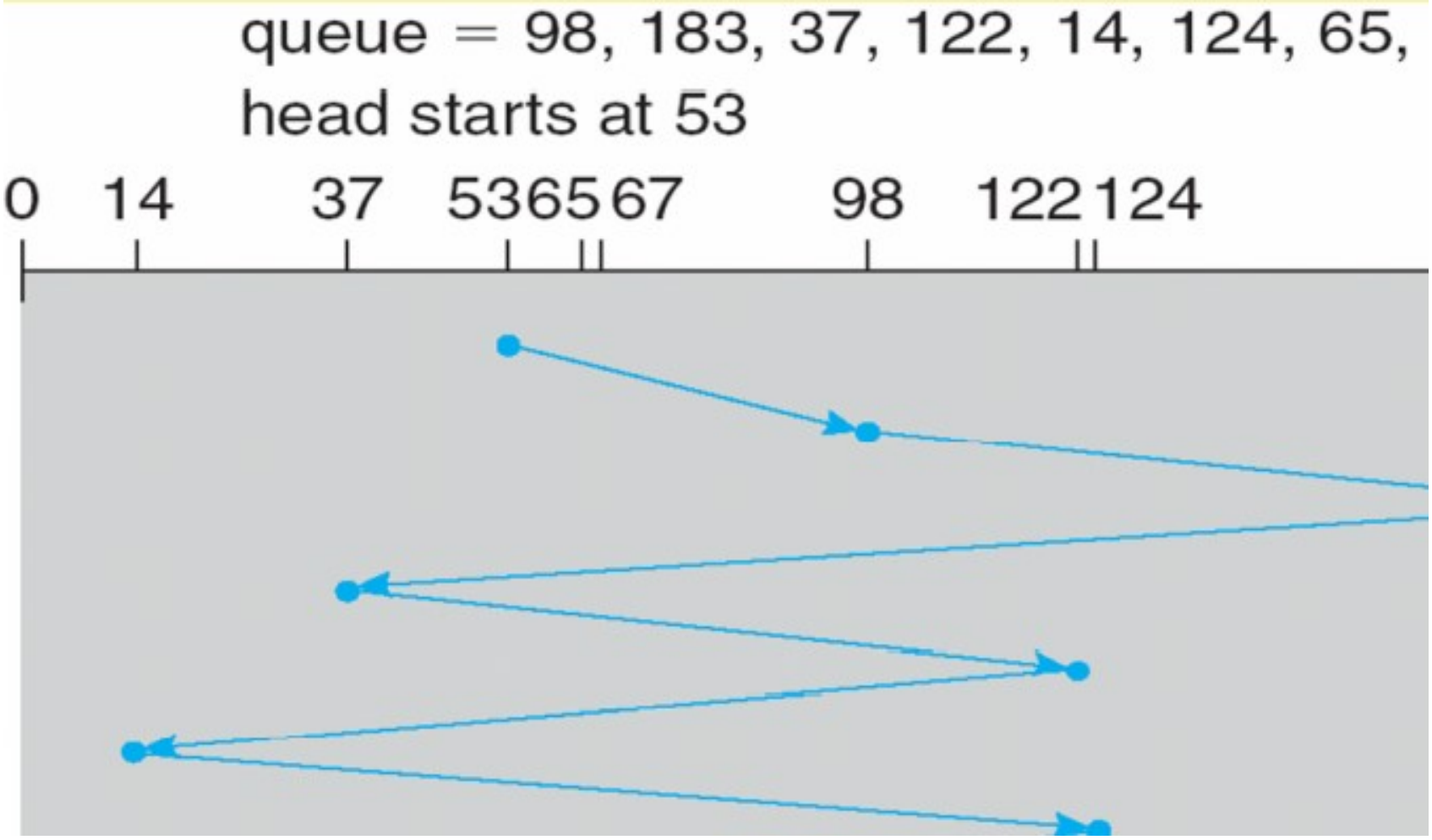
Desirable Characteristics of Scheduling Algorithms

- Throughput: the number of requests serviced per unit time; should be maximum
- Mean response time: average waiting time plus average service time; should be minimum
- Variance of response times (predictability): variance is a measure of how far individual items tend to deviate from the average of items. The smaller the variance, the greater is the predictability

First Come First Served

- The first request to arrive is the first one serviced
- Place of request in the schedule is fixed; request cannot be displaced because of the arrival of a higher priority request
- Results in random seek patterns
- Ignores positional relationships among the pending requests in the queue
- Makes no attempt at optimizing the seek pattern
- Acceptable when the load on a disk is light
- Under heavy load, tends to saturate the device and response times become large

FCFS



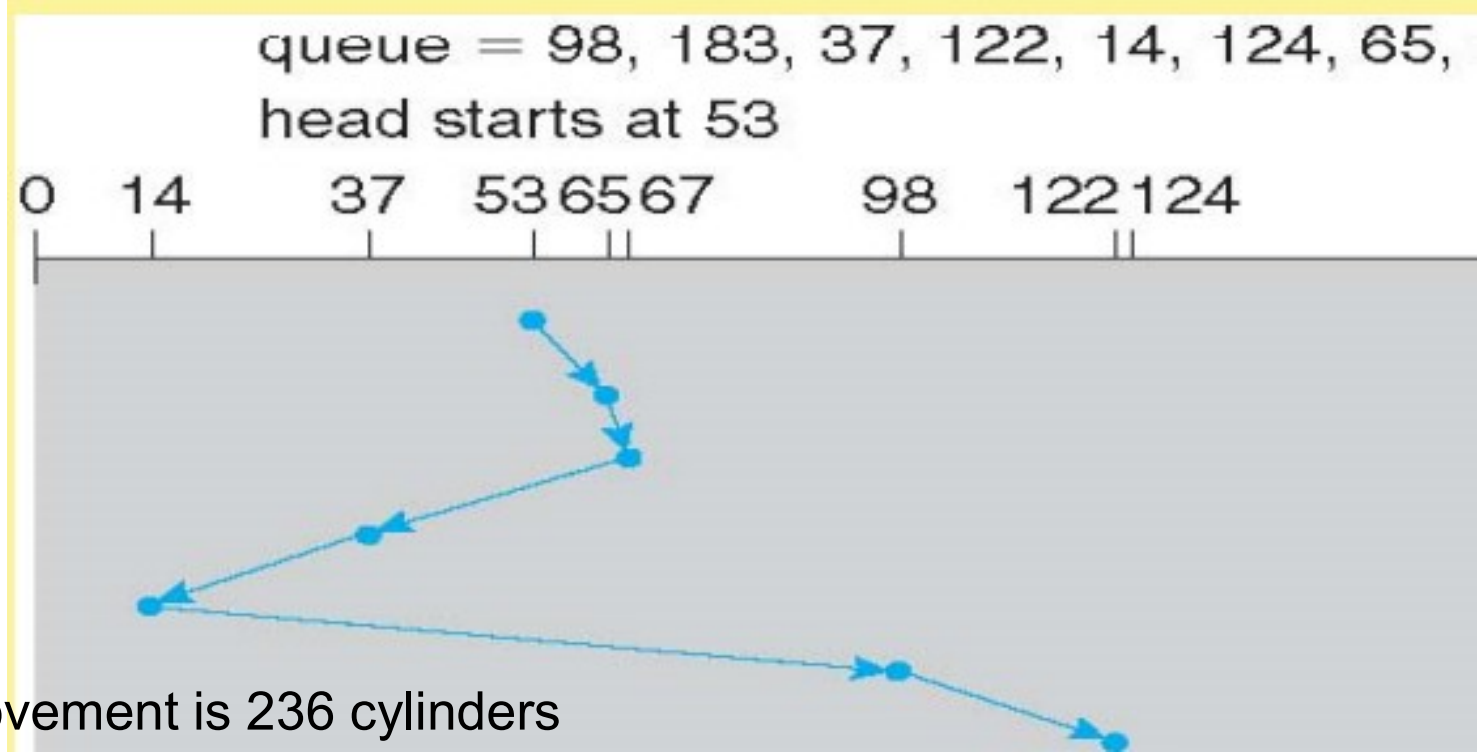
Total head movement is 640 cylinders

Shortest Seek time First

- Requests that results in shortest seek distance is serviced first; even if the request is not the first one in the queue
- Seek patterns tend to be highly localized with the result that the innermost and outermost tracks can receive poor service compared with the mid range tracks
- Better throughput rates than FCFS; mean response times tend to be lower for moderate loads
- Higher variance occur on response times because of the discrimination against the outermost and innermost tracks; can cause starvation of requests far from the read-write heads

SSTF

- Useful in batch processing systems where throughput is the major consideration
- Unacceptable in interactive systems because of high variance of response times

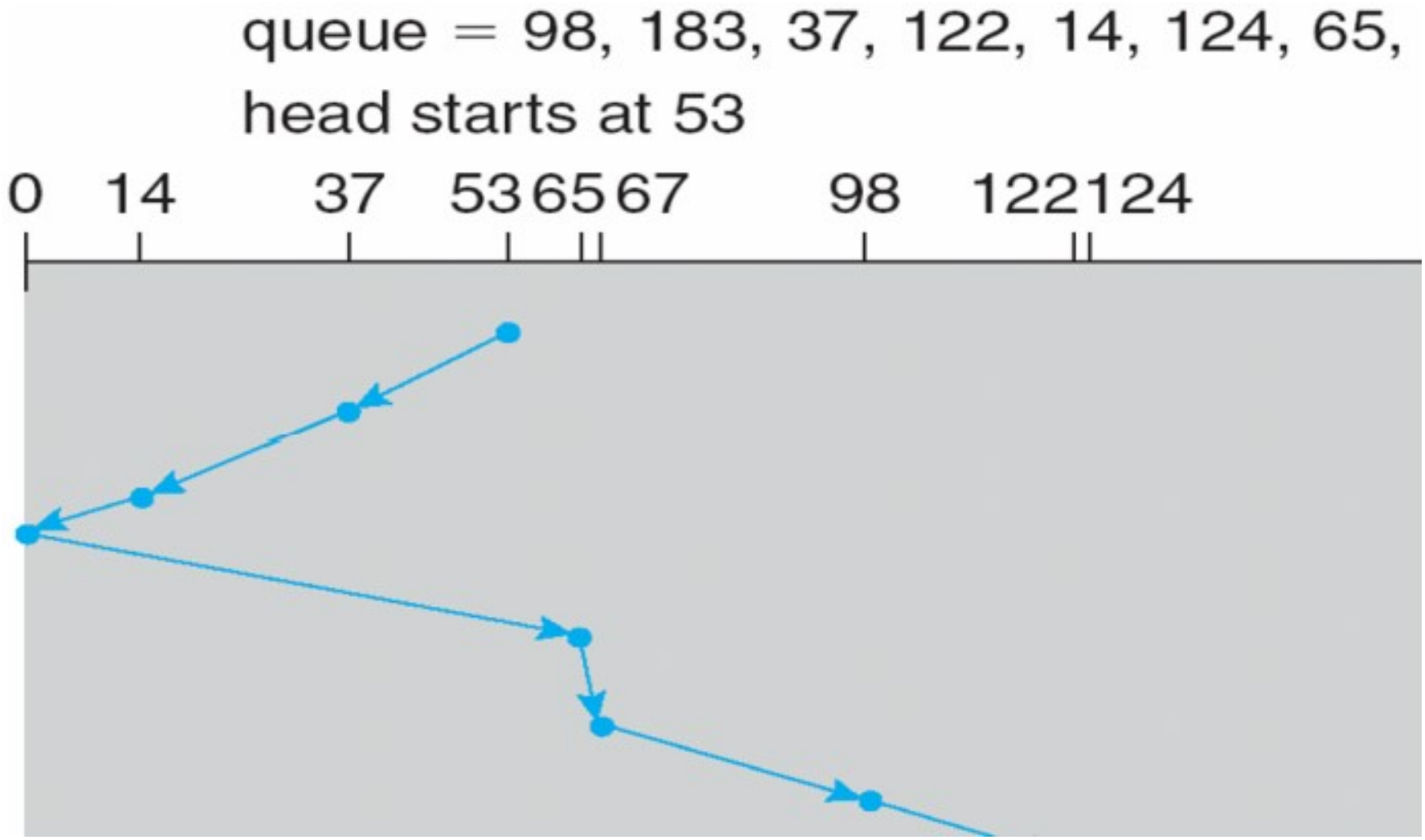


Total head movement is 236 cylinders

SCAN

- Overcomes the discrimination and high variance in response times of SSTF; chooses the requests that results in the shortest seek distance in a preferred direction
- Does not change direction until it reaches the outermost or innermost cylinder or until there are no more requests pending and then it reverses direction
- Also known as the Elevator algorithm
- In terms of improved throughput and mean response times, behaves like SSTF, however eliminates much of discrimination inherent in SSTF
- Causes the requests in direction opposite to the head movement to wait longer

SCAN

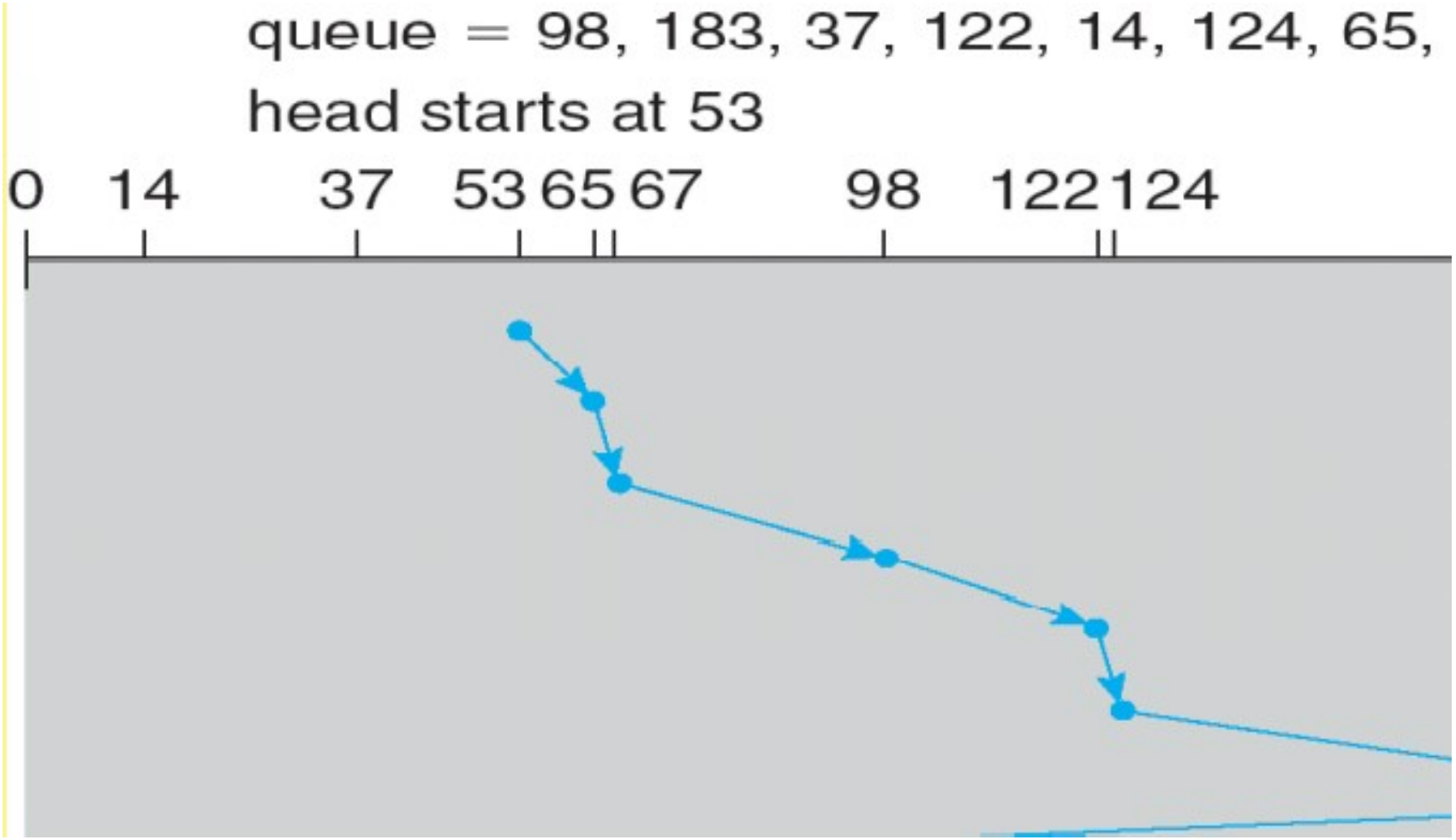


Total head movement is 208 cylinders

C-SCAN

- A variant of SCAN designed to provide a more uniform wait
- Like SCAN moves the head from one end of the disk to the other, servicing requests along the way. However, when the head reaches the end, it immediately returns to the beginning of the disk without servicing any requests on the round trip ; circular list
- At low loading, SCAN policy is best and under medium to heavy loading, C-SCAN yield the best results

C-SCAN


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

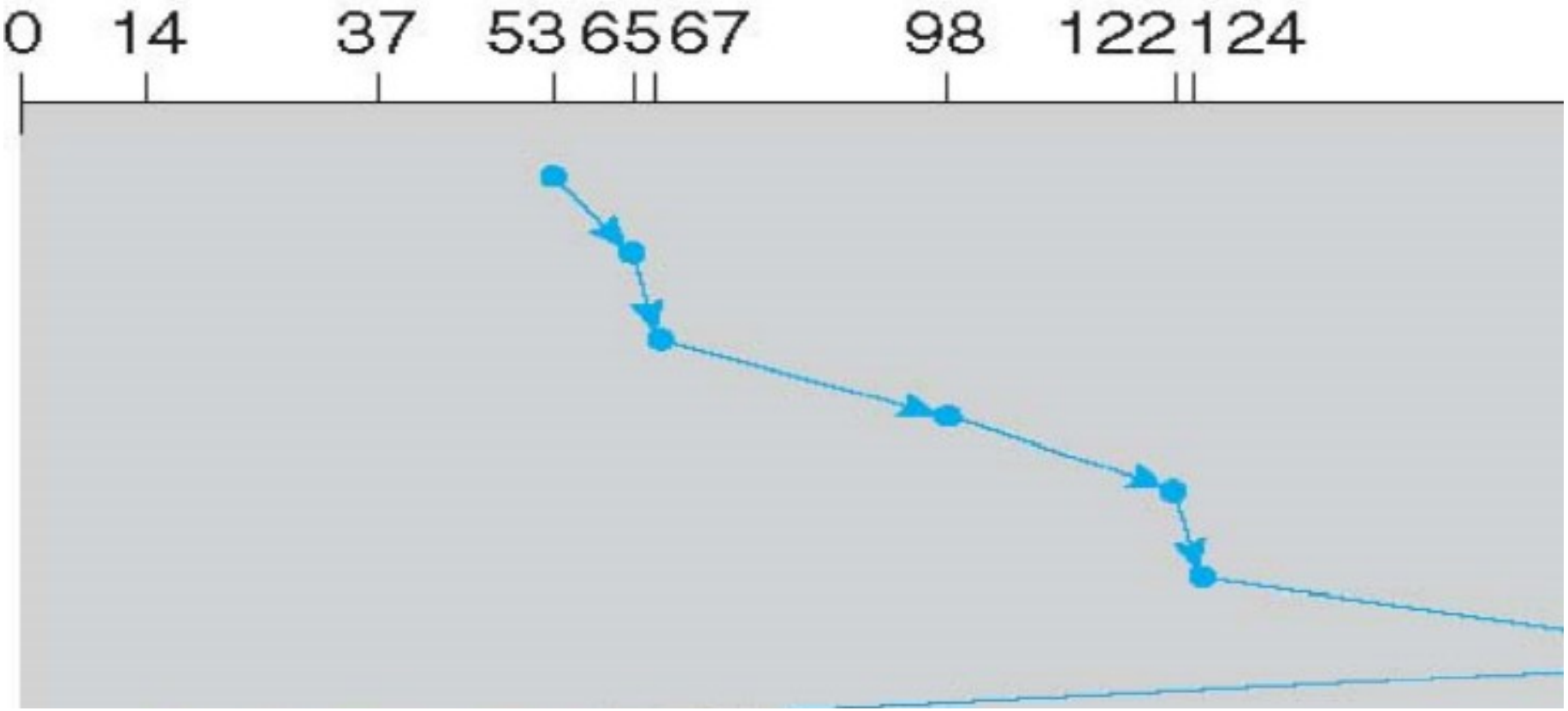
LOOK/C-LOOK Scheduling

- SCAN and C-SCAN move the disk arm across full width of the disk
- In practice, the arm goes as far as the final request in each direction. Then, it reverses the direction immediately, without going all the way to the end of the disk. i.e it looks for a request before continuing to move in a given direction

C-LOOK Scheduling Example

queue = 98, 183, 37, 122, 14, 124, 65, 6

head starts at 53



Selection of Algorithms

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
- Performance dependent on number and types of requests (with single request in queue, all will give same performance)
- Disk service request is greatly influenced by the file allocation method
- The location of directories and index box is also important; caching helps in reducing the arm movement

Rotational Optimization

- Also known as Sector Queuing
- The concept is to reduce the latency time by minimizing the disk rotations
- Paralleling the SSTF strategy of seek optimization is the SLTF i,.e Shortest Latency Time First
- Once the disk arm arrives at a particular cylinder, there may be many requests pending on the various tracks of that cylinder; the SLTF strategy examines all these requests and service the one with the shortest delay first
- In order to reduce the rotation of the disk, requests are queued by sector position around the disk and the nearest sectors are serviced first

Example

- Suppose head of a disk, with 200 tracks numbering 0 to 199, is currently serving a request at track 140. The queue of pending requests, in FIFO order is

  84, 147, 91,177,94,150,102,175,130

  Assuming earlier direction to be towards zero, calculate total head movements for following disk scheduling algorithms :

  - FCFS
  - SSTF
  - SCAN
  - LOOK