File Management

- File management is a major function of an operating system
- User need know how data is stored or retrieved from secondary storage

Computer Storage

- There are two levels of storage in a computer
  - Main memory – Temporary stores data and programs being executed - volatile
  - Secondary storage - stores data and programs permanently, for future use. Non-volatile

File System

- The file system is the most visible aspect of an operating system.
- It provides the mechanism for on-line storage of and access to both data and programs of the operating system and all the users of the computer system.
- The file system consists of two distinct parts: a collection of files, each storing related data, and a directory structure, which organizes and provides information about all the files in the system.

Minimum User Requirements

- View
- Read(open)
- Write(save)
- Edit
- Copy/Cut
- Move

Secondary storage devices

- Magnetic tapes
- Magnetic disks – Hard disc, zip disk
- Optical Disc – CD, DVD

Computer Files

- Data is stored in secondary storage in form of files.
- A file has two components
  - Name – uniquely identifies the file in secondary storage
  - Extension – denotes file format

Physical Vs Logical files

- For convenient use of the computer system, 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.

OS File Management Functions

- The operating system is responsible for the following activities in connection with file management:
  - The creation and deletion of files
  - The creation and deletion of directory
  - The mapping of files onto disk storage.
  - Backup of files on stable (non volatile) storage

File Control Block

- File control block
- storage structure consisting of information about a file.
- File attributes include:
  - **Name: The symbolic file name is the only information kept in humanreadable** 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 those systems that support different** types.
  - **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.

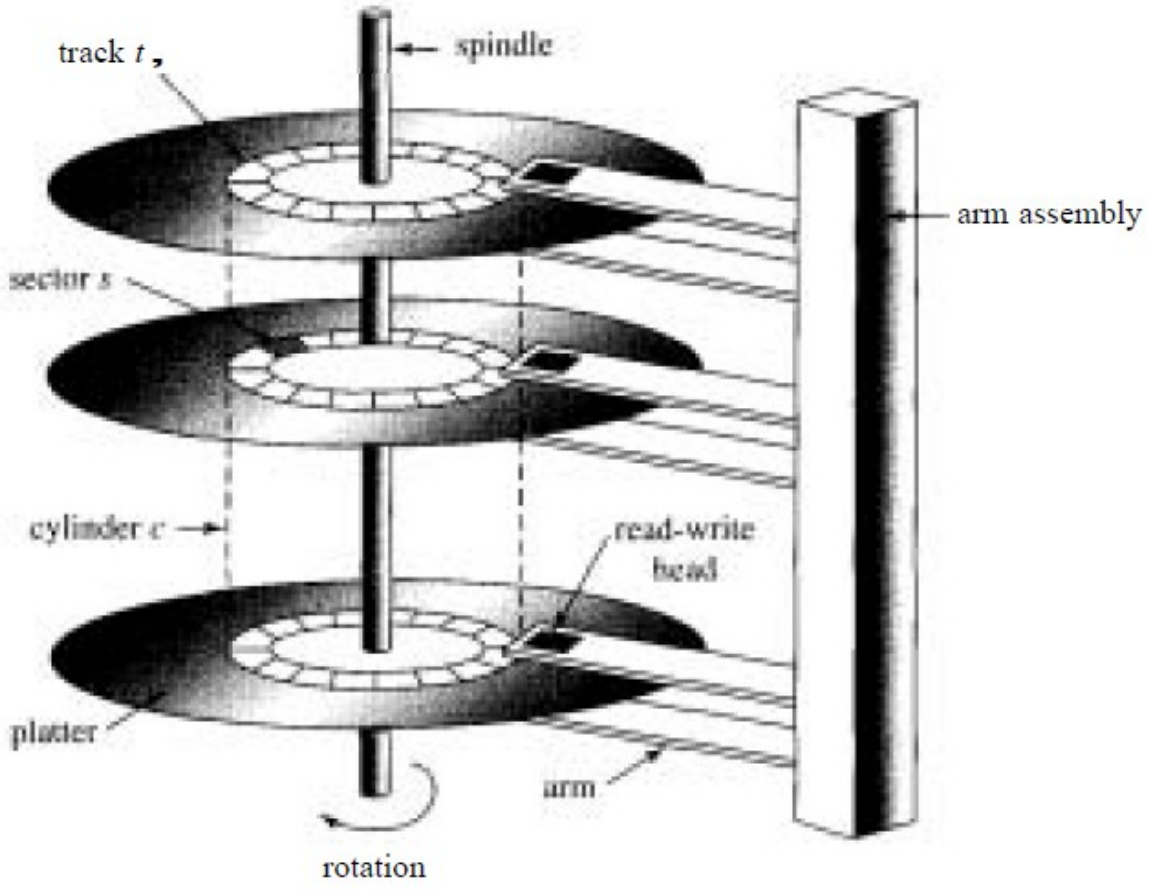
File operations

- **Creating a file: create a file space in the** file system must be found for the file. An entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the file system, and possibly other information.
- **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.

- **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.

- **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.

Structure of a disk



- Formatting – Prepares the disk for data storage. Disk surface is divided into concentric circles known as tracks which are further divided into sectors
- Each sector has an address which uniquely identifies its location

Disk Access Times

- When computer is in use, files are transferred between main memory and secondary storage.
- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.
- Access time has two major components
  - *Seek time* is the time for the disk are to move the heads to the cylinder containing the desired sector.
  - *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head.
  - Head switching time – activate head for reading or writing
  - Data transfer
- Minimize seek time
  - Seek time seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Disk Scheduling

- Several algorithms exist to schedule the servicing of disk I/O requests. Examples
- FCFS
- SSTF
- SCAN
- CSCAN
- LOOK
- CLOOK

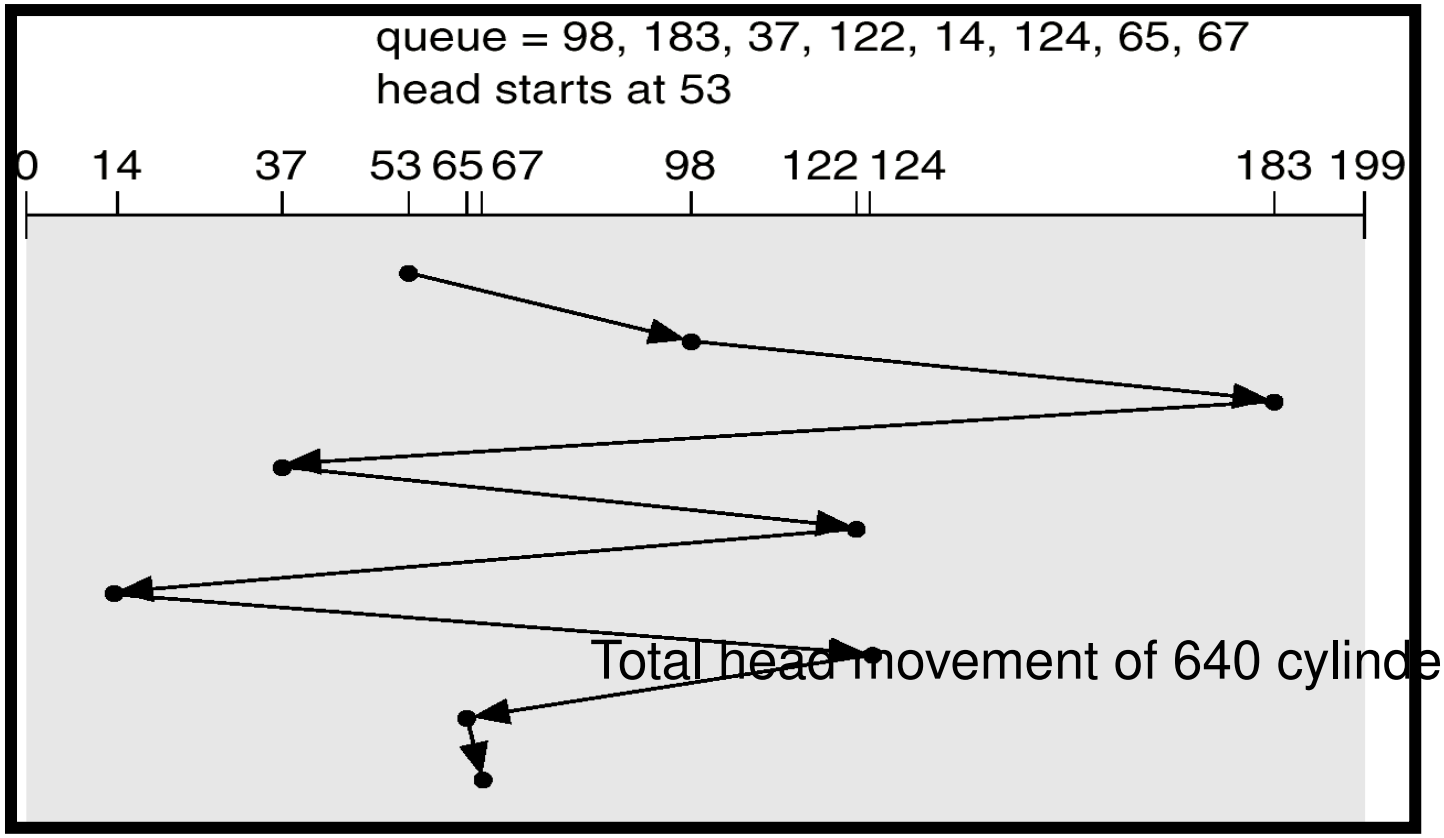
- We illustrate them with a request queue (0-199) of cylinders requested

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

  Head pointer is at 53

FCFS

- Arrival order of service requests is used

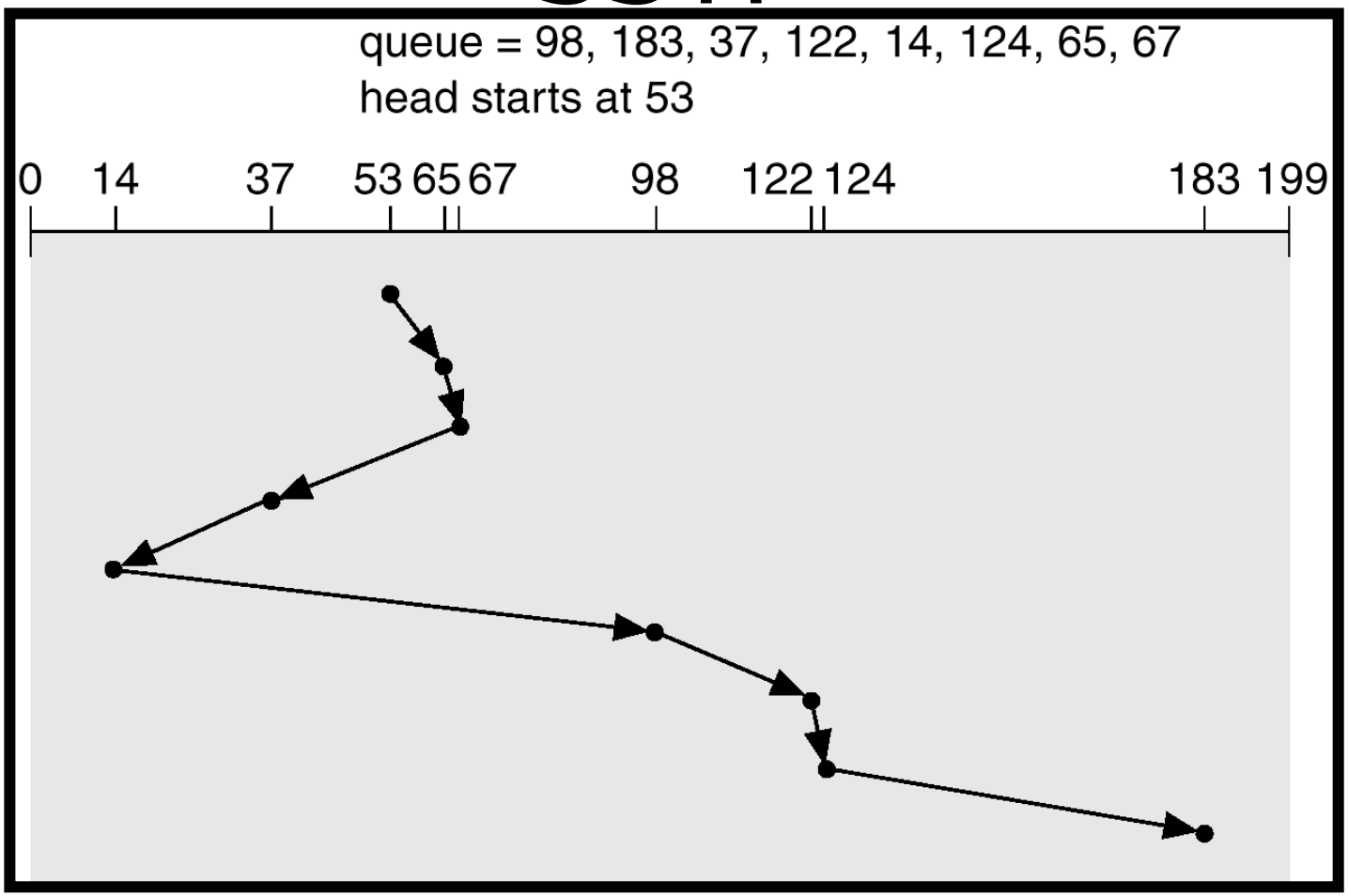


SSTF

- Selects the request with the minimum seek time from the current head position.

.

SSTF



Total head movement of 236 cylinders.

SCAN

- 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*.

.

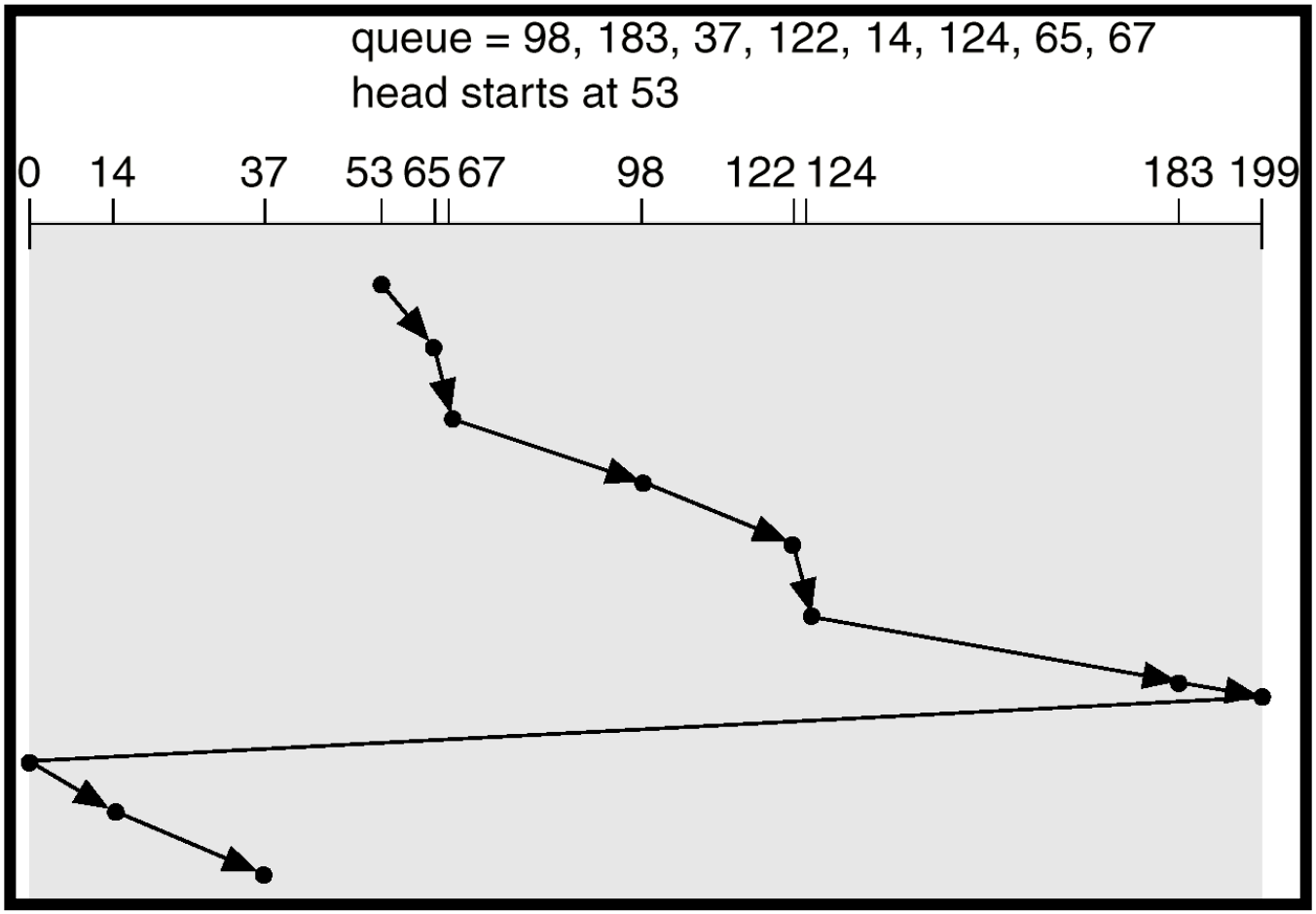
SCAN (Cont.)

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

C-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.

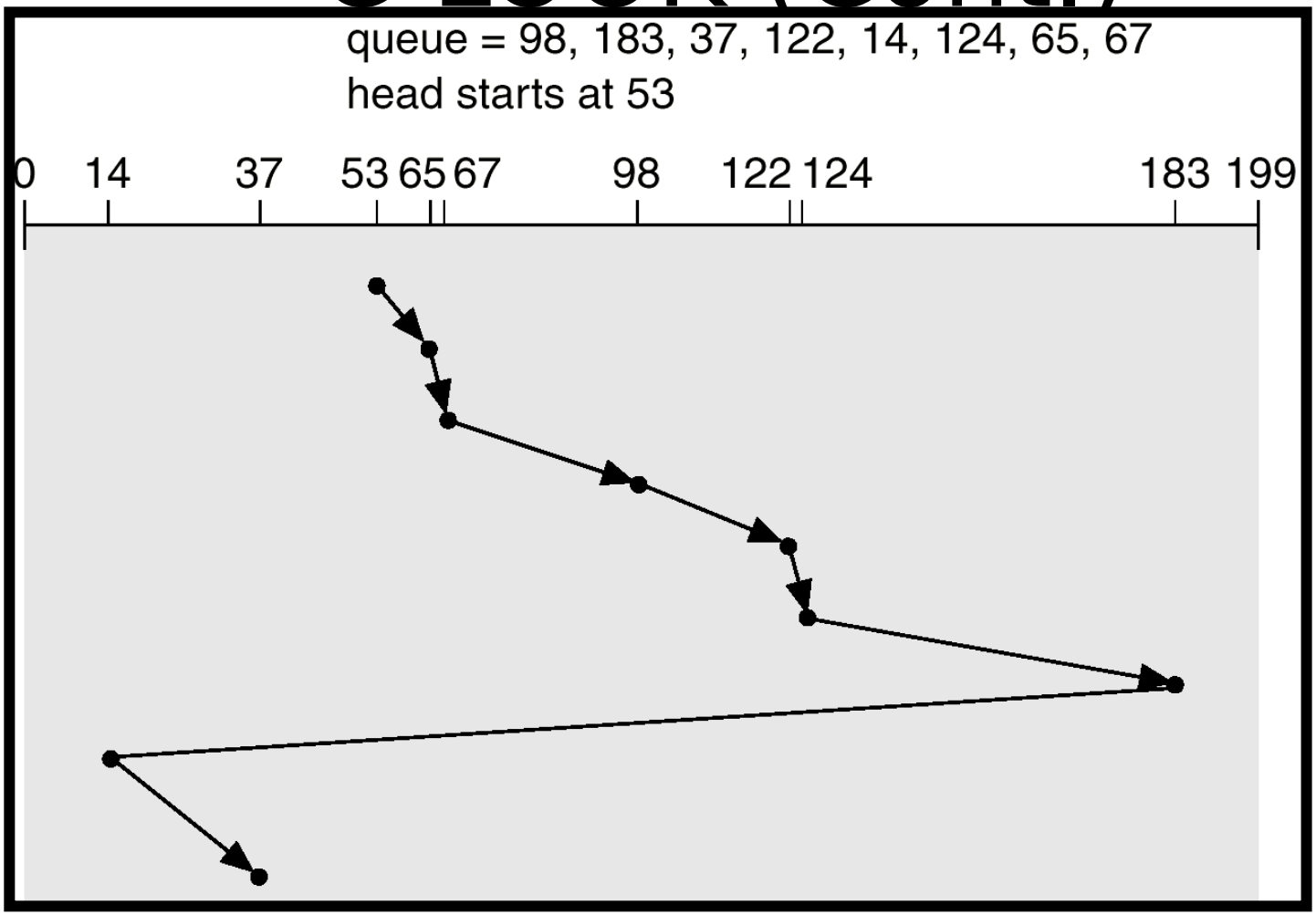
C-SCAN (Cont.)


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

LOOK and C-LOOK

- Arm only goes as far as the last request in each direction instead of till the last cylinder
- LOOK and C-LOOK are obtained from SCAN and C-SCAN by using this idea

C-LOOK (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

Selecting a Disk-Scheduling Algorithm

- 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 depends on the number and types of requests.
- Requests for disk service can be influenced by the file allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

Directory Structure

- file systems may consist of millions of files
- Typically, OS defines
  - Containers that can store files
    - Called partitions, volumes, etc
    - May span one or more devices
  - A Directory provides a logical grouping of files

Information in a Device Directory

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed
- Date last updated
- Owner ID
- Protection information

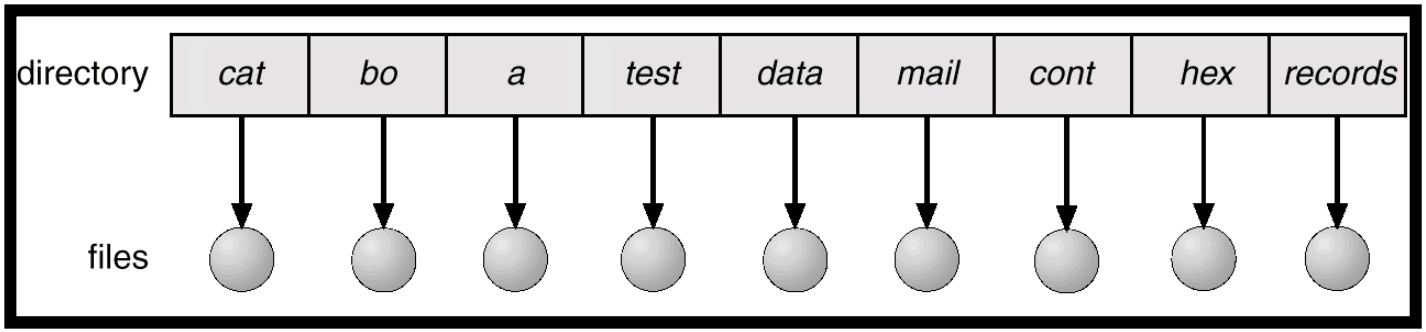
Organize the Directory (Logically) to Obtain

- **Efficiency** – locating a file quickly.
- **Naming** – convenient to 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, ...)

Evolution of Directory Structures

Single-Level Directory

- A single directory for all users.

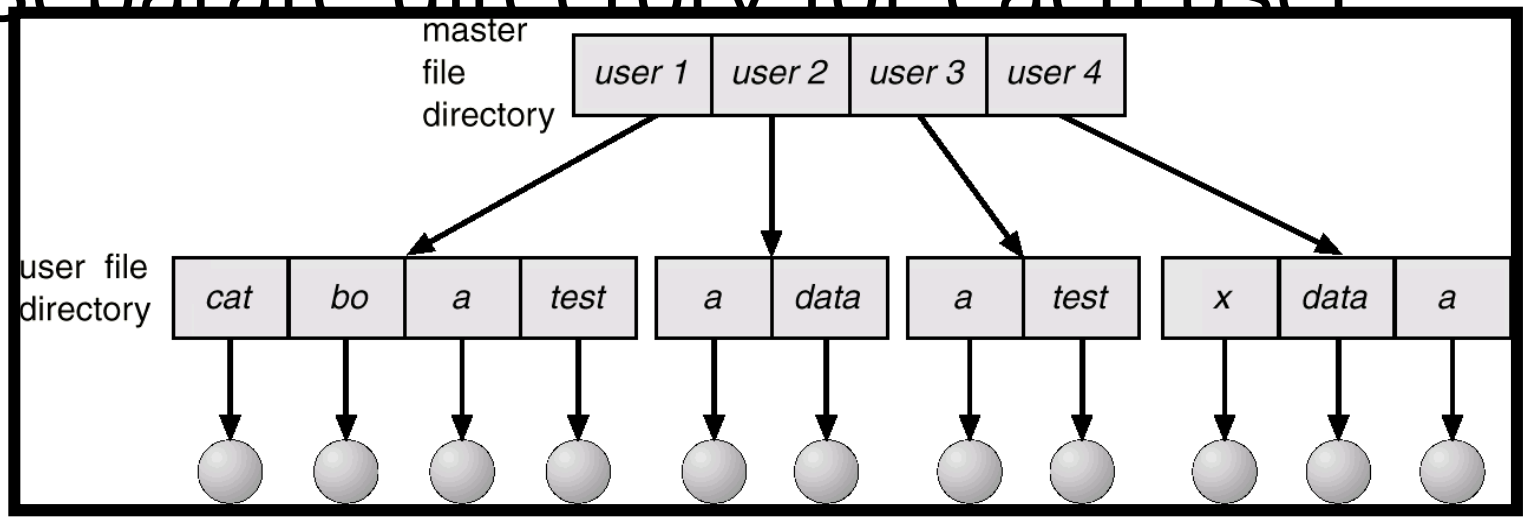


Naming problem

Grouping problem

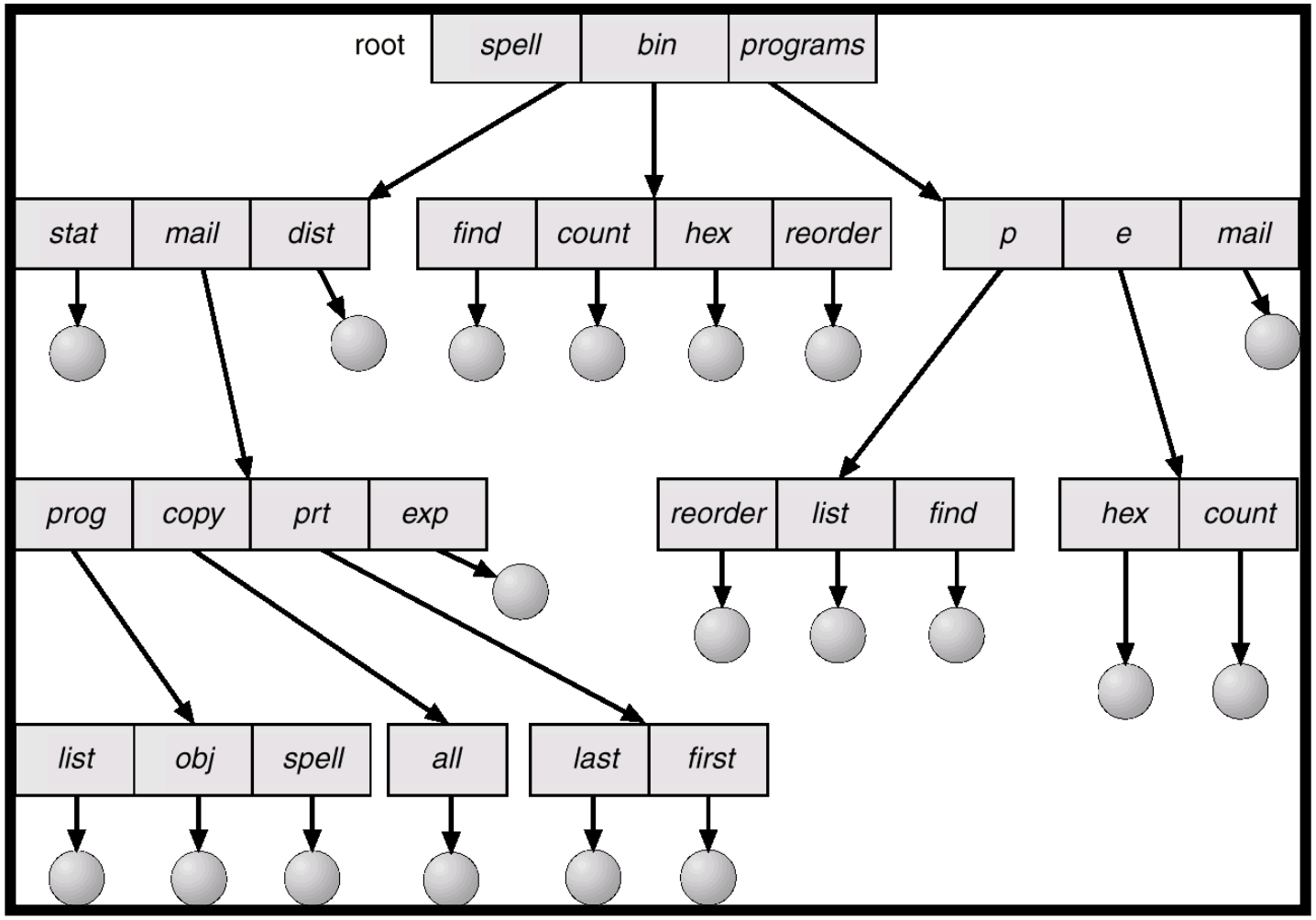
Two-Level Directory

- Separate directory for each user



- Path name
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree-Structured Directories



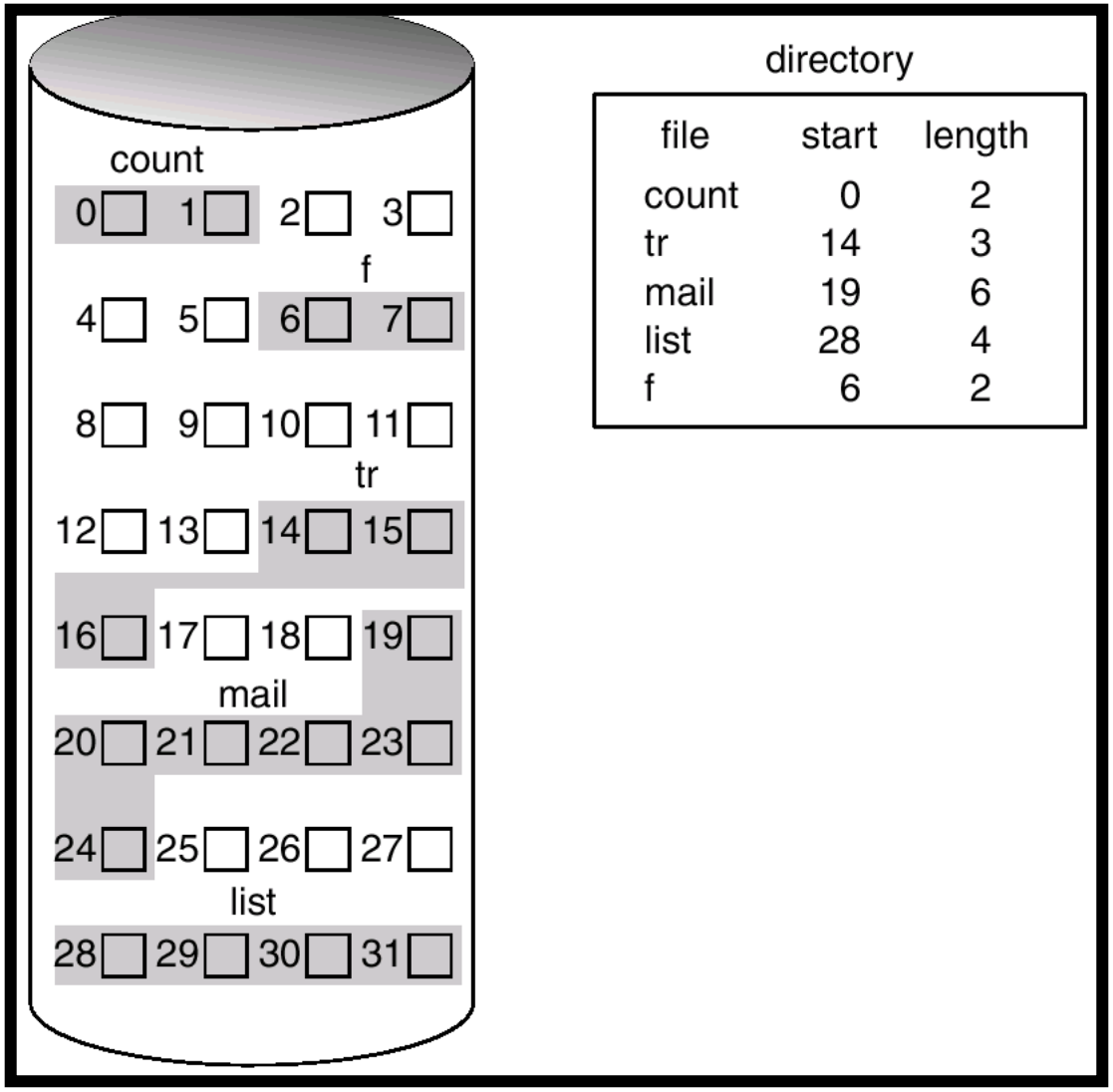
Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation

Contiguous Allocation

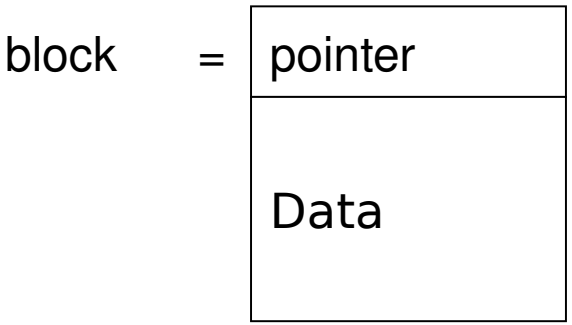
- Each file occupies a set of contiguous blocks on the disk.
- Simple – only starting location (block #) and length (number of blocks) are required.
- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.

Contiguous Allocation of Disk Space

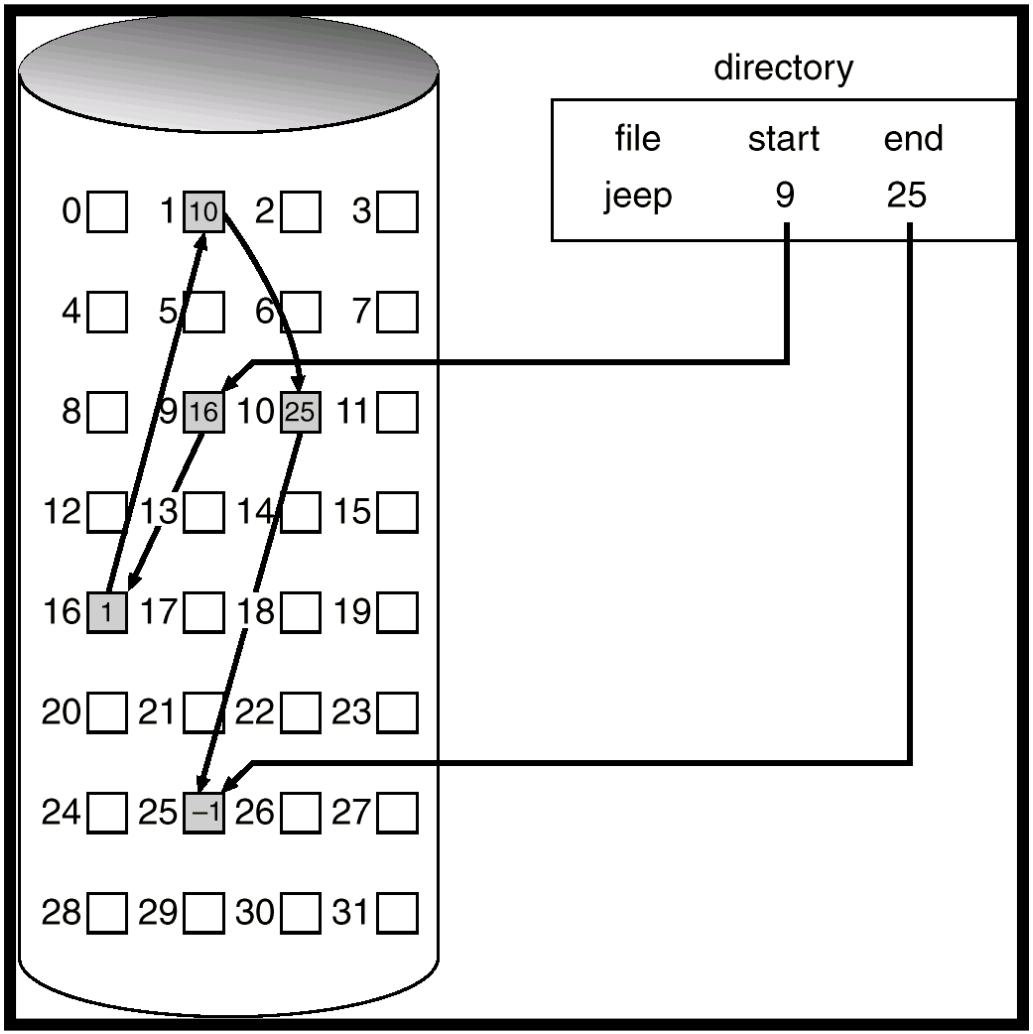


Linked Allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.

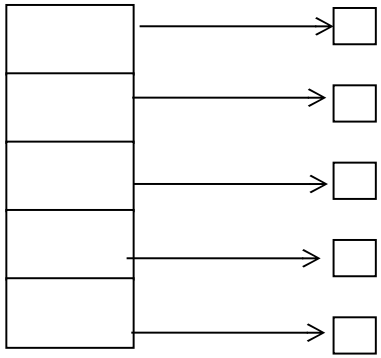


Linked Allocation



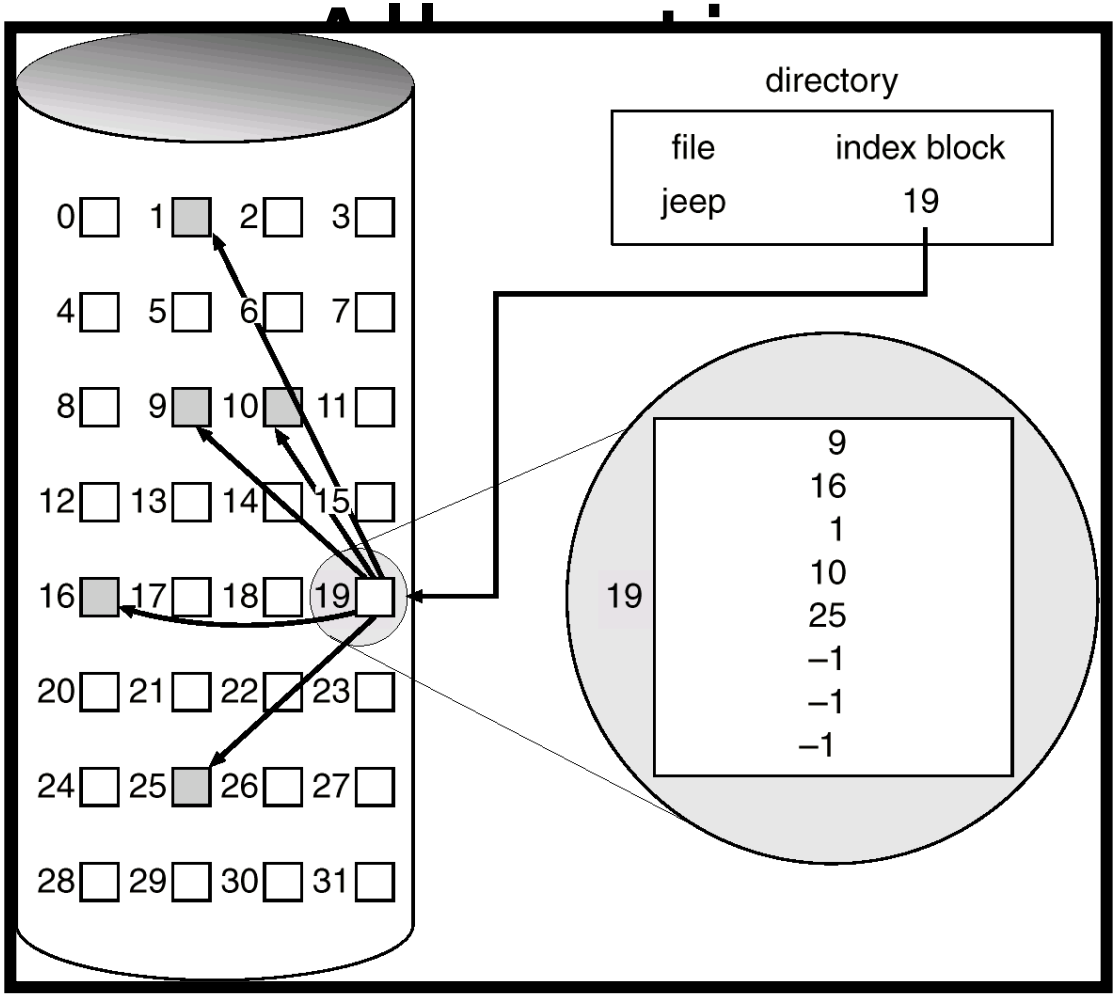
Indexed Allocation

- Brings all pointers together into the *index block.*
- Logical view.



index table

Example of Indexed Allocation



END