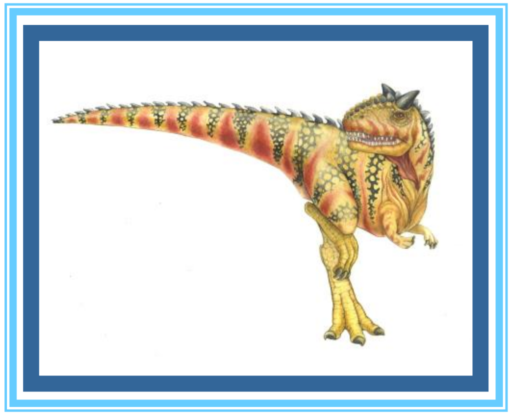
Chapter 12: Mass-Storage Systems

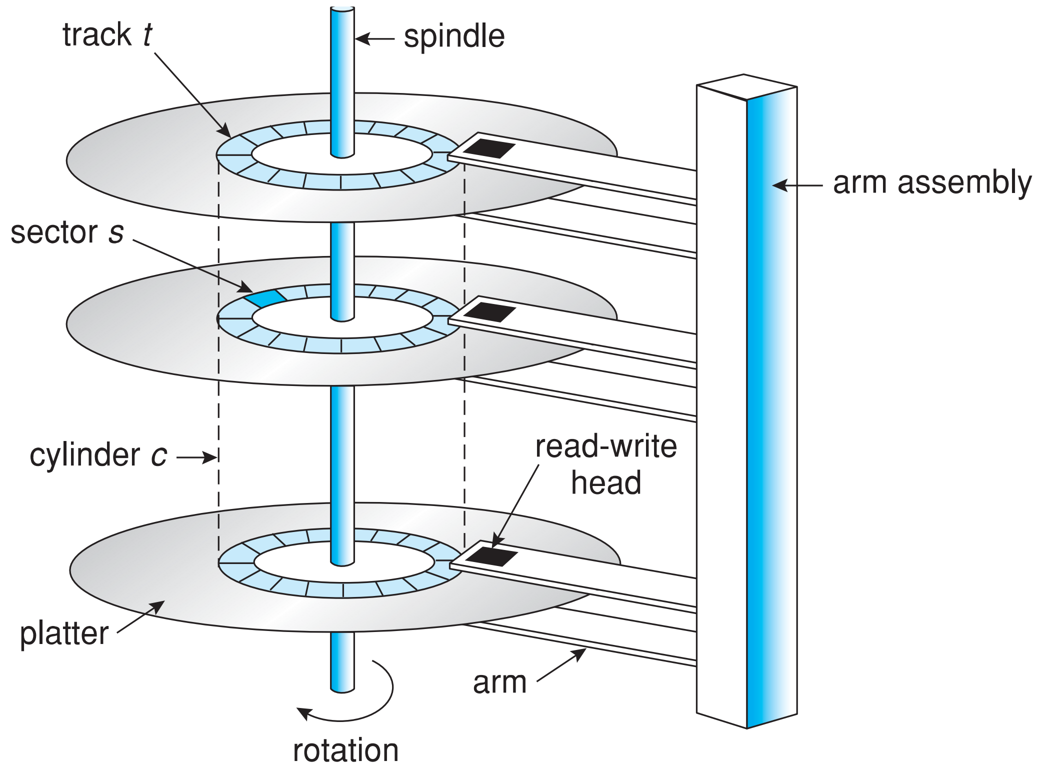


Chapter 12: Mass-Storage Systems

- Disk Structure
- Disk Scheduling
- Swap-Space Management

Disk Structure

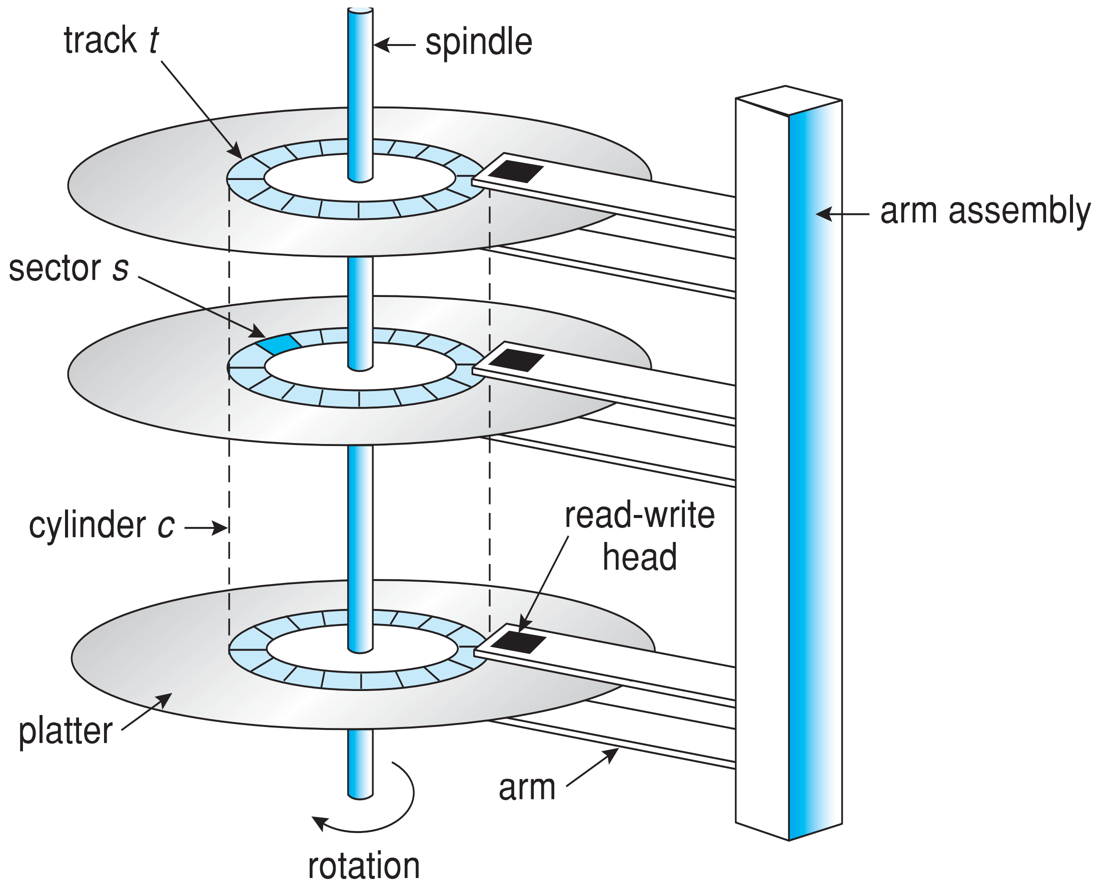
- A disk is a storage device that stores and retrieves data using magnetic or optical technology.
- It is a flat, circular plate made of a rigid material, such as metal or plastic, with a thin coating on its surface.
- Disks are commonly used in computers and other electronic devices for long-term data storage



Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Low-level formatting creates **logical blocks** size, such as **1,024** bytes on physical media
- 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
  - Logical to physical address that consists of a cylinder number, a track number within that cylinder, and a sector number within that track.

Moving-head Disk Mechanism



Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having **a fast access time** and **large disk bandwidth**

Access time has two major components:

- **Seek time-** time taken for the disk arm to move the heads to the cylinder containing the desired sector. It should be minimum

- **Rotational latency** is the additional time for the disk to rotate the desired sector to the disk head
- 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
- We can improve both the **access time and the bandwidth** by managing the order in which disk I/O requests are serviced.

Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address to transfer, memory address to transfer, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
- when one request is completed, the operating system chooses which pending request to service next.
- How does the operating system make this choice?

Disk Scheduling (Cont.)

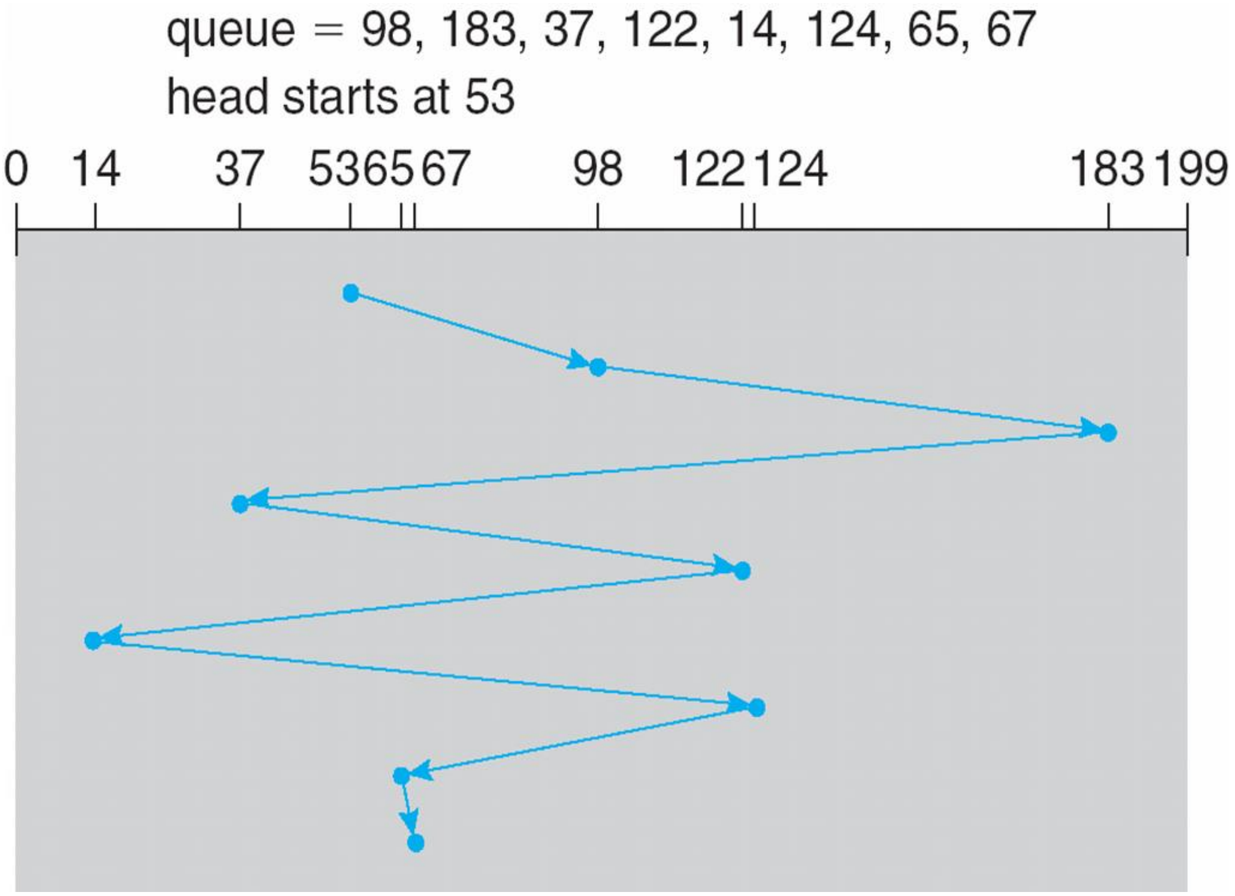
- Note that drive controllers have small buffers and can manage a queue of I/O requests
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request for I/O to blocks on cylinders

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

  Head pointer 53

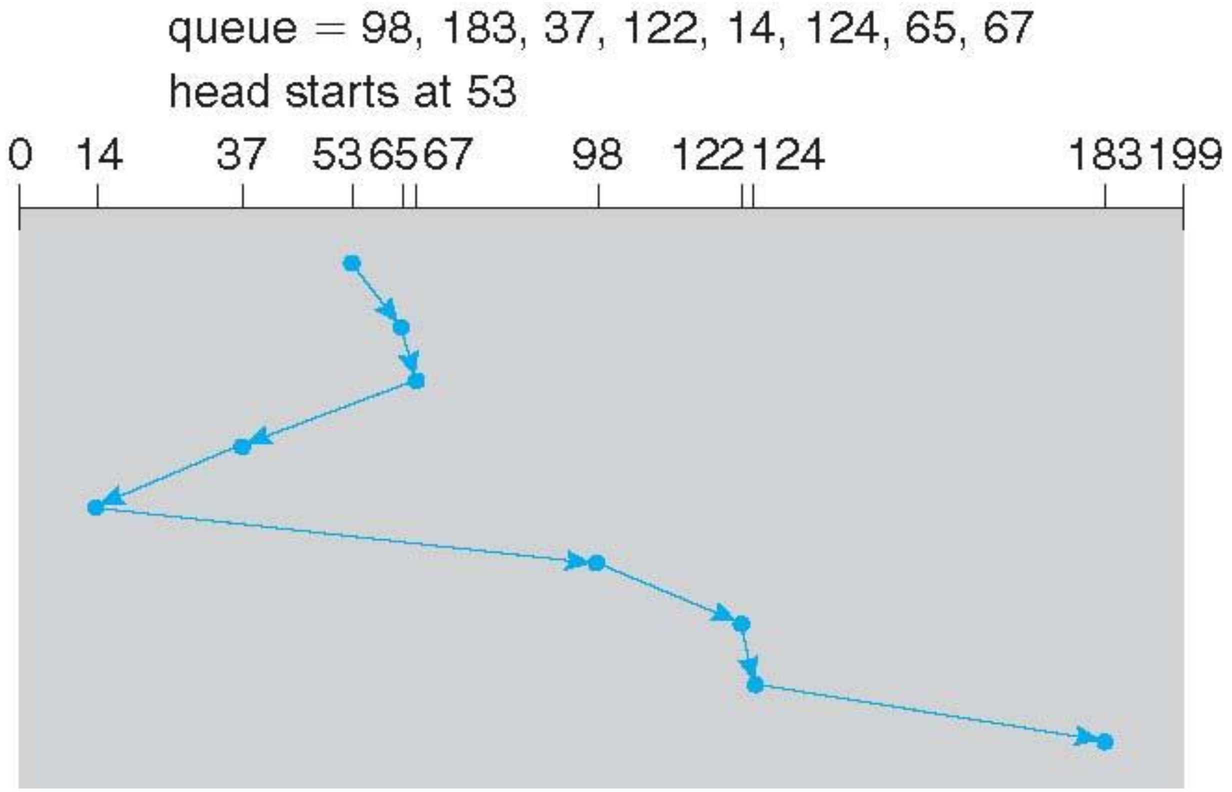
FCFS

Illustration shows total head movement of 640 cylinders



SSTF

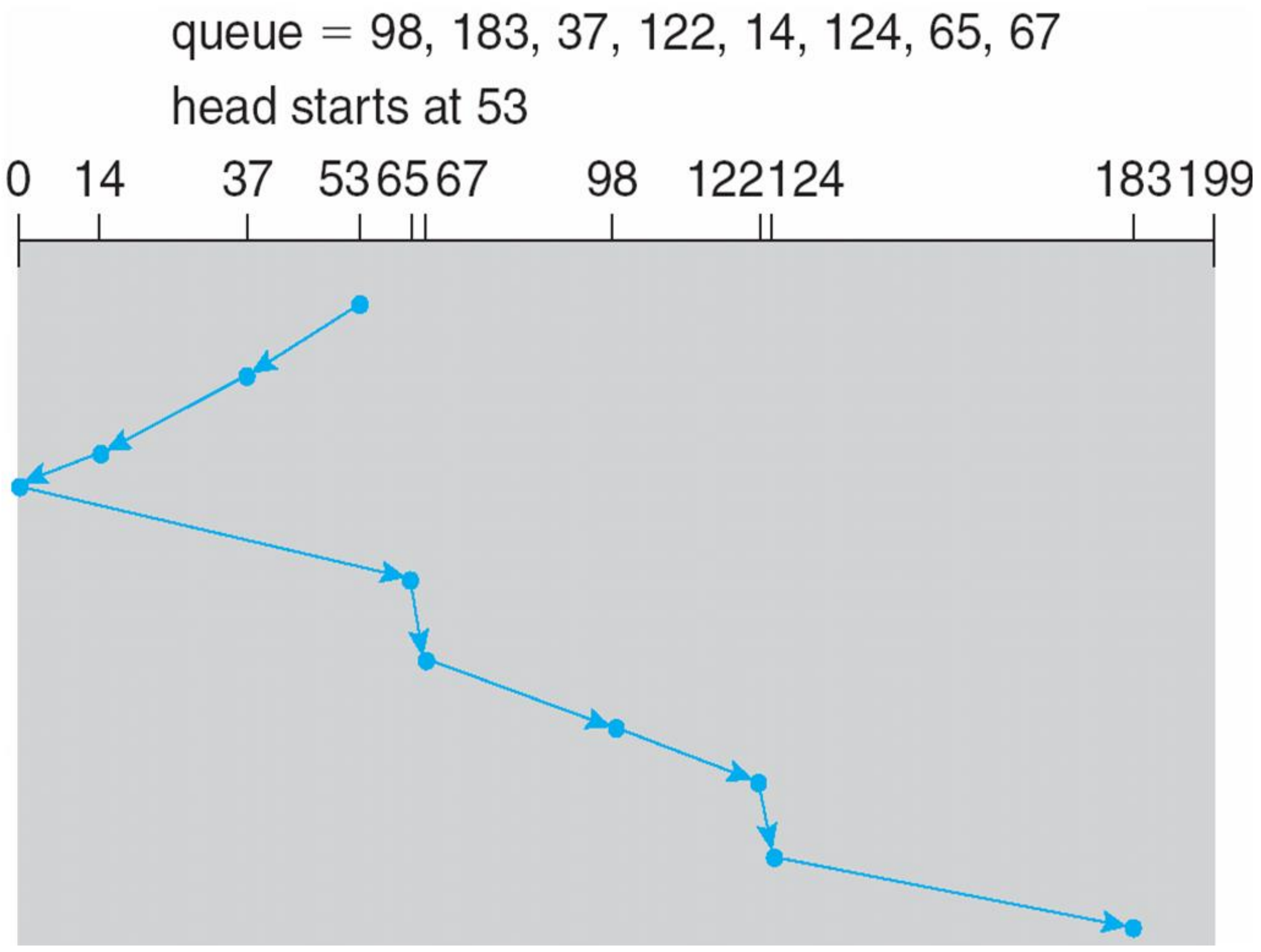
- Shortest Seek Time First selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders



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.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- Illustration shows total head movement of 236 cylinders
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

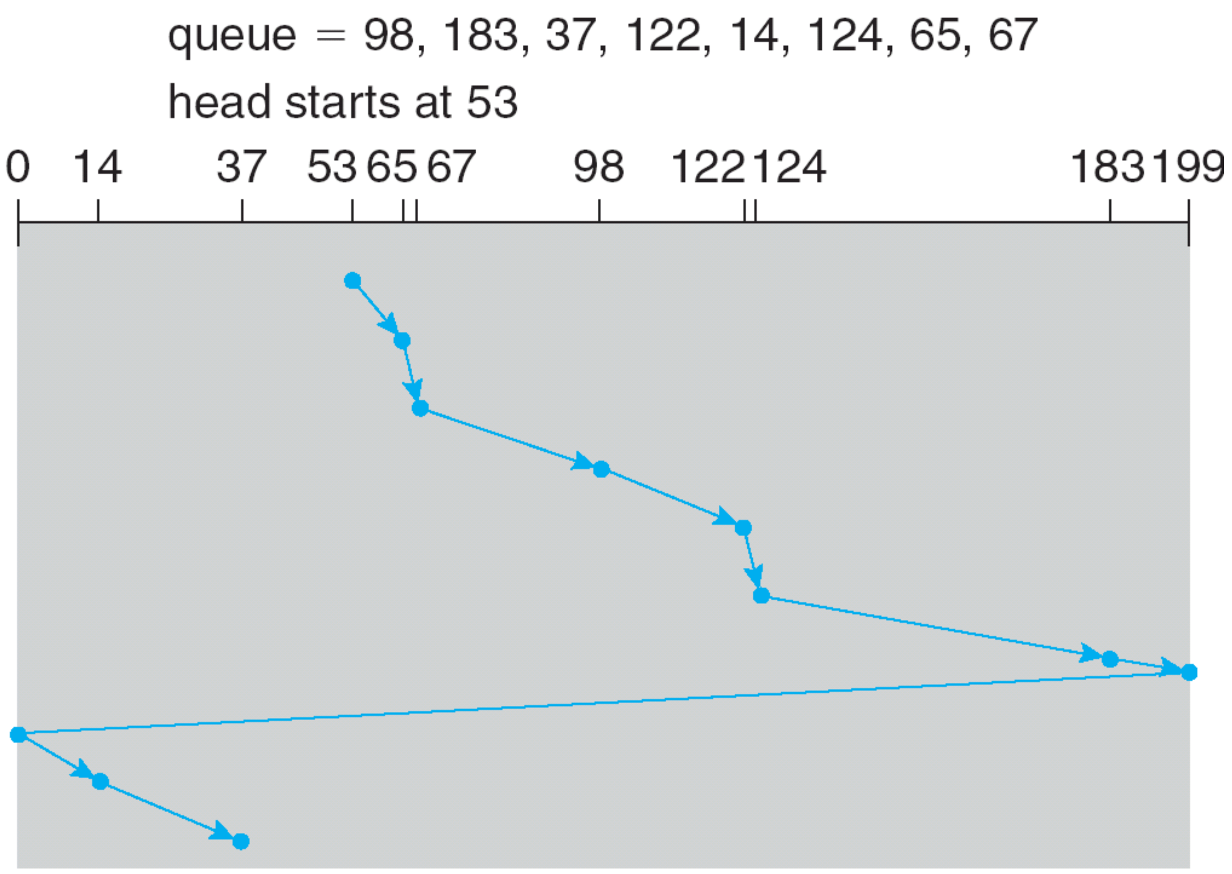
SCAN (Cont.)


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

- **Circular SCAN (C-SCAN) scheduling** is a variant of SCAN designed to provide a more uniform wait time.
- The head moves from one end of the disk to the other, servicing requests as it goes
    - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

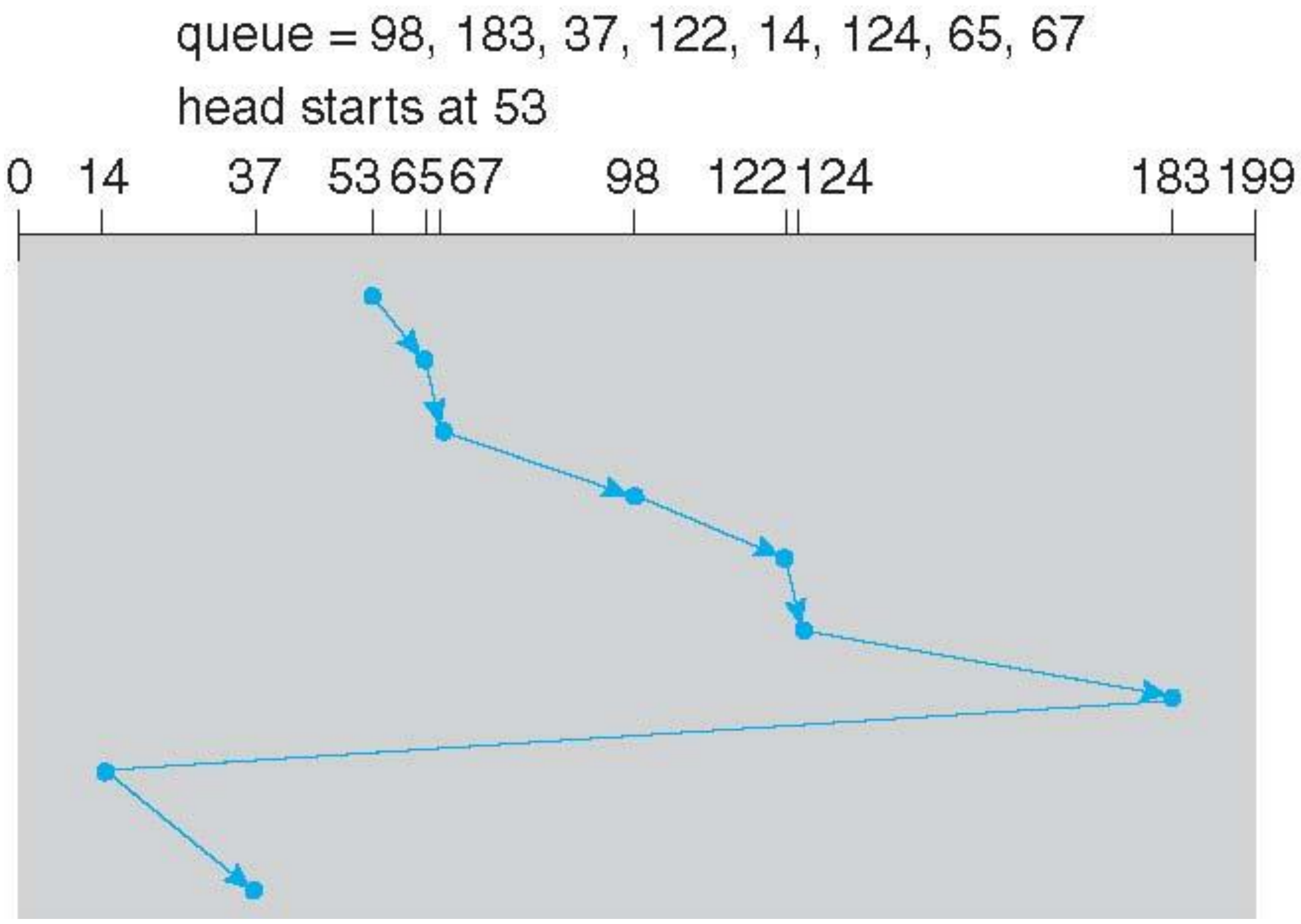
C-SCAN (Cont.)


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

C-LOOK

- Both SCAN and C-SCAN move the disk arm across the full width of the disk.
- In practice, neither algorithm is often implemented this way.
- More commonly, the arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to the end of the disk.
- Versions of SCAN and C-SCAN that follow this pattern are called **LOOK**

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
  - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
  - A program reading contiguously allocated file will generate several requests that are close together on the disk, resulting in limited head movement.
  - A linked or indexed file, in contrast, may include blocks that are widely scattered on the disk, resulting in greater head movement.

Selecting a Disk-Scheduling Algorithm

- 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

Swap-Space Management

- **Swapping** is a memory management technique used in multi-programming for removing a process from the main memory and storing it into secondary memory, and then bringing it back into the main memory for continued execution
- **Swap-Space :**
  The area on the disk where the swapped-out processes are stored is called swap space.
- **Swap-space management:**

➢ Swap-Space management is another low-level task of the operating system to optimize memory usage and improve system performance.

➢ The goal of this swap-space implementation is to provide the virtual memory the best throughput.

➢ The systems which are implementing **swapping** may use swap space to hold the entire process which may include image, code and data segments.

➢ **Paging systems** may simply store **pages** that have been pushed out of the main memory.

Swap-Space Management

- **Swap-Space Location :**
  A swap space can reside in one of the two places –

1) Normal file system :

✓ swap-space is simply a large file within the file system.

✓ To create it, name it and allocate its space **normal filesystem** routines can be used.

✓ This approach, through easy to implement, is inefficient.

✓ Navigating the directory structures and the disk-allocation data structures takes **time and extra disk access**.

2) Separate disk partition :

✓ Swap space can be created in a separate **raw** partition

✓ A swap space **storage manager** is used to allocate and de-allocate the blocks from the raw partition.

✓ the access time of swap space is shorter than the file system

✓ Raw partition approach creates fixed amount of swap space in case of the **disk partitioning**.

Data Structures for Swapping on Linux Systems

- The traditional UNIX kernel started with an implementation of swapping that copied entire process between contiguous disk regions and memory.

- UNIX later evolve to a combination of swapping and paging as paging hardware became available

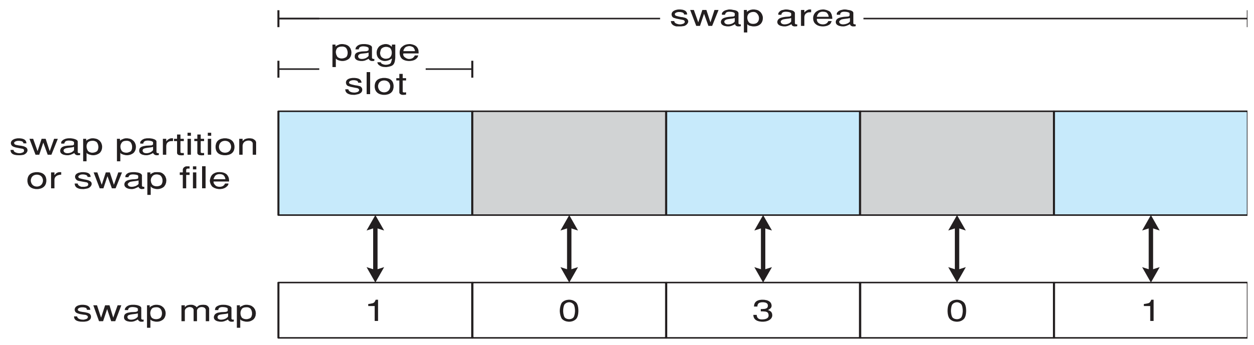
- In Solaris Swap space is only used as a backing store for pages of anonymous memory, which includes memory allocated for the stack, heap, and uninitialized data of a process

- Linux is almost similar to Solaris system. In both the systems the swap space is used only for anonymous memory.

- Linux allows one or more swap areas to be established.

Data Structures for Swapping on Linux Systems

- Each swap area consists of a series of 4-KB **page slots**, which are used to hold swapped pages.
- Associated with each swap area is a **swap map**—an array of integer counters, each corresponding to a page slot in the swap area.
- If the value of a counter is 0, the corresponding page slot is available.
- Values greater than 0 indicate that the page slot is occupied by a swapped page.
- The value of the counter indicates the number of mappings to the swapped page.
- For example, a value of 3 indicates that the swapped page is mapped to three different processes



End of Chapter 10