Operating System
Chapter 6: Input/Output

Prepared By:
Amit K. Shrivastava
Asst. Professor
Nepal College Of Information Technology

Introduction

- The two main jobs of a computer are I/O and processing. In many cases, the main job is I/O and the processing is merely incidental. For instance, when we browse a web page or edit a file, our immediate interest is to read or enter some information, not to compute an answer.
- It must issue commands to the devices, catch interrupts, and handle errors. It should also provide an interface between the devices and the rest of the system that is simple and easy to use. To the extent possible, the interface should be the same for all devices. The I/O code represents a significant fraction of the total operating system.

Principles of I/O hardware

- Different people look at I/O hardware in different ways. Electrical engineers look at in terms of chips, wires, power supplies, motors, and all the other physical components that make up the hardware.
- Programmers look at the interface presented to the software- the commands the hardware accepts, the functions it carries out, and the errors that can be reported back.
- Here we are concerned with programming I/O devices, not designing, building, or maintaining them.

I/O Devices

- I/O devices can be roughly divided into two categories: block devices and character devices.
- A block device is one that stores information in fixed- size blocks, each one with its own address. Common block sizes range from 512 bytes to 32,768 bytes. All transfers are in units of one or more entire blocks. The essential property of a block device is that it is possible to read or write each block independently of all the other ones. Hard disks, CD-Roms, and USB sticks are common block devices.
- A character device delivers or accepts a stream of characters, without regard to any block structure. It is not addressable and does not have any seek operation. Printers, network interfaces, mice and most other devices that are not disk- like can be seen as character device.

Device Controllers

- I/O units typically consist of a mechanical component and an electronic component. The electronic component is called the device controller or adapter. On personal computers, it often takes the form of a chip on the parent board or a printed circuit card that can be inserted into a (PCI) expansion slot.
- The controller card usually has a connector on it, into which a cable leading to the device itself can be plugged. Many controllers can handle two, four, or even eight identical devices.
- The interface between the controller and device is often a very low-level interface. What actually comes off the drive, however, is a serial bit stream and finally a checksum. The controller's job is to convert the serial bit stream into a block of bytes and perform any error correction necessary. The block of bytes is typically first assembled, bit by bit, in a buffer inside the controller. After its checksum has been verified and the block has been declared to be error free, it can be copied to main memory.

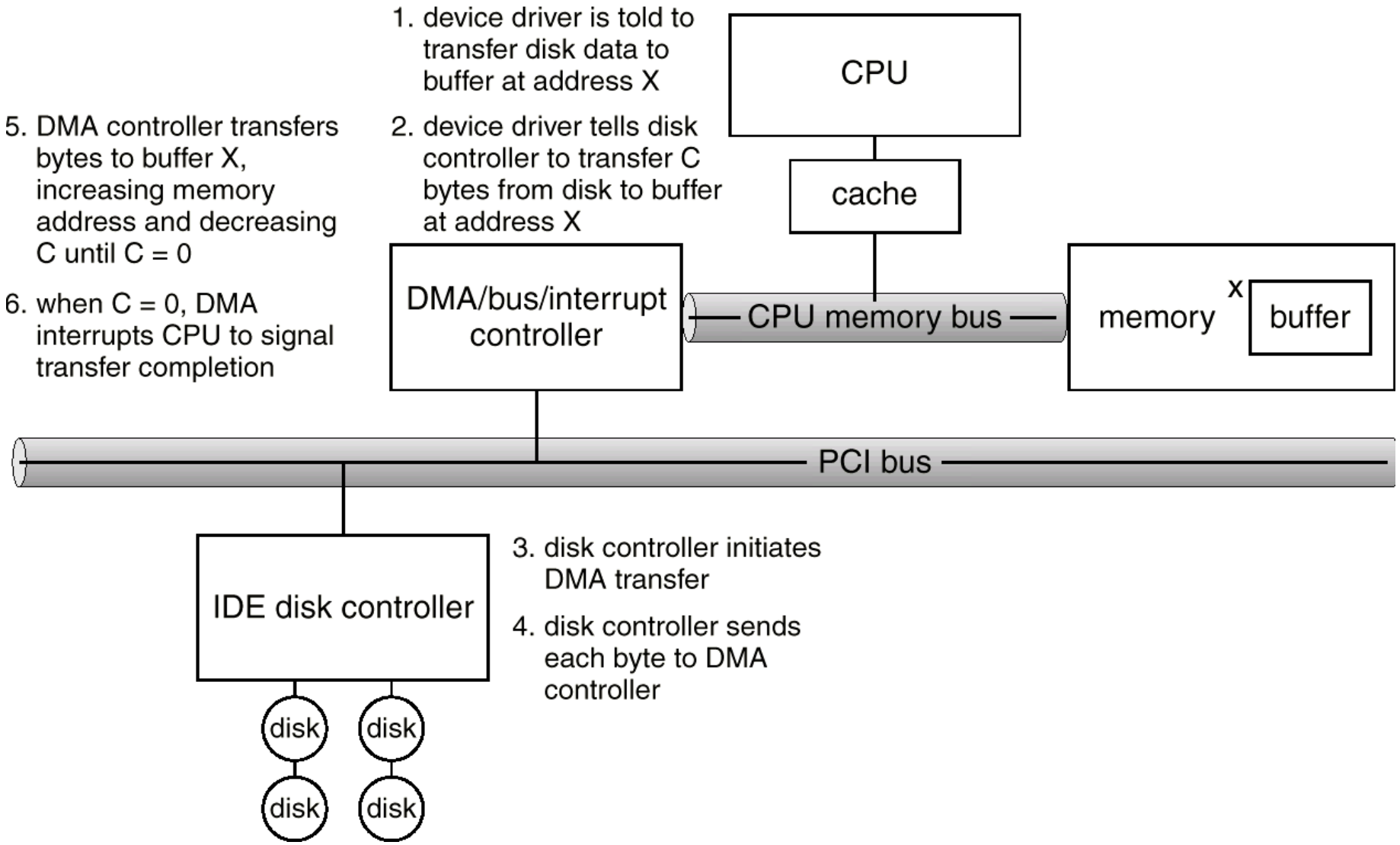
Memory –Mapped I/O

- Each controller has a few registers that are used for communicating with the CPU. By writing into these registers, the operating system can command the device to deliver data, accept data, switch itself on or off, or otherwise perform some action. By reading from these registers, the operating system can learn what the device state is, whether it is prepared to accept a new command, and so on.
- The issue thus arises of how the CPU communicates with the control registers and the device data buffers. For this we map all the control registers into memory space. Each control register is assigned a unique memory address to which no memory is assigned. This system is called **memory-mapped I/O**. Usually, the assigned address are at the top of the address space.

DMA(Direct Memory Access)

- Many computers avoid burdening the main CPU with **PIO** by offloading some of this work to a special-purpose processor called a directmemory-access **(DMA**) controller. To initiate a DMA transfer, the host writes a DMA command block into memory. This block contains a pointer to the source of a transfer, a pointer to the destination of the transfer, and a count of the number of bytes to be transferred.
- The CPU writes the address of this command block to the DMA controller, then goes on with other work. The DMA controller proceeds to operate the memory bus directly, placing addresses on the bus to perform transfers without the help of the main CPU. A simple DMA controller is a standard component in PCs,and bus-mastering I/O boards for the PC usually contain their own high-speed DMA hardware.

Six Step Process to Perform DMA Transfer



Goals of the I/O Software

- A key concept in the design of I/O software is known as **device independence.** It means that is should be possible to write programs that can access any I/O device without having to specify the device in advance. For example, a program that reads a file as input should be able to read a file on hard disk, a CD-ROM, a DVD, or a USB stick without having to modify the programs for each different device.
- Closely related to device independence is the goal of **uniform naming.** The name of a file or a device should simply be a string or an integer and not depend on the device in any way.

Goals of the I/O Software(contd...)

- Another important issue for I/O software is **error handling.** In general, errors should be handled as close to the hardware as possible. If the controller discovers a read error, it should try to correct the error itself if it can. If it cannot, then the device driver should handle it, perhaps by just trying to read the block again.
- Another key issue is that **synchronous**(blocking) versus **asynchronous**(interrupt-driven) transfers. Most physical I/O is asynchronous- the CPU starts the transfer and goes off to do something else until the interrupt arrives.
- Another issue for the I/O software is **buffering.** Often data that come off a device cannot be stored directly in its final destination.For example, when a packet comes in off the network, the operating system does not know where to put it until it has stored the packet somewhere and examined it.

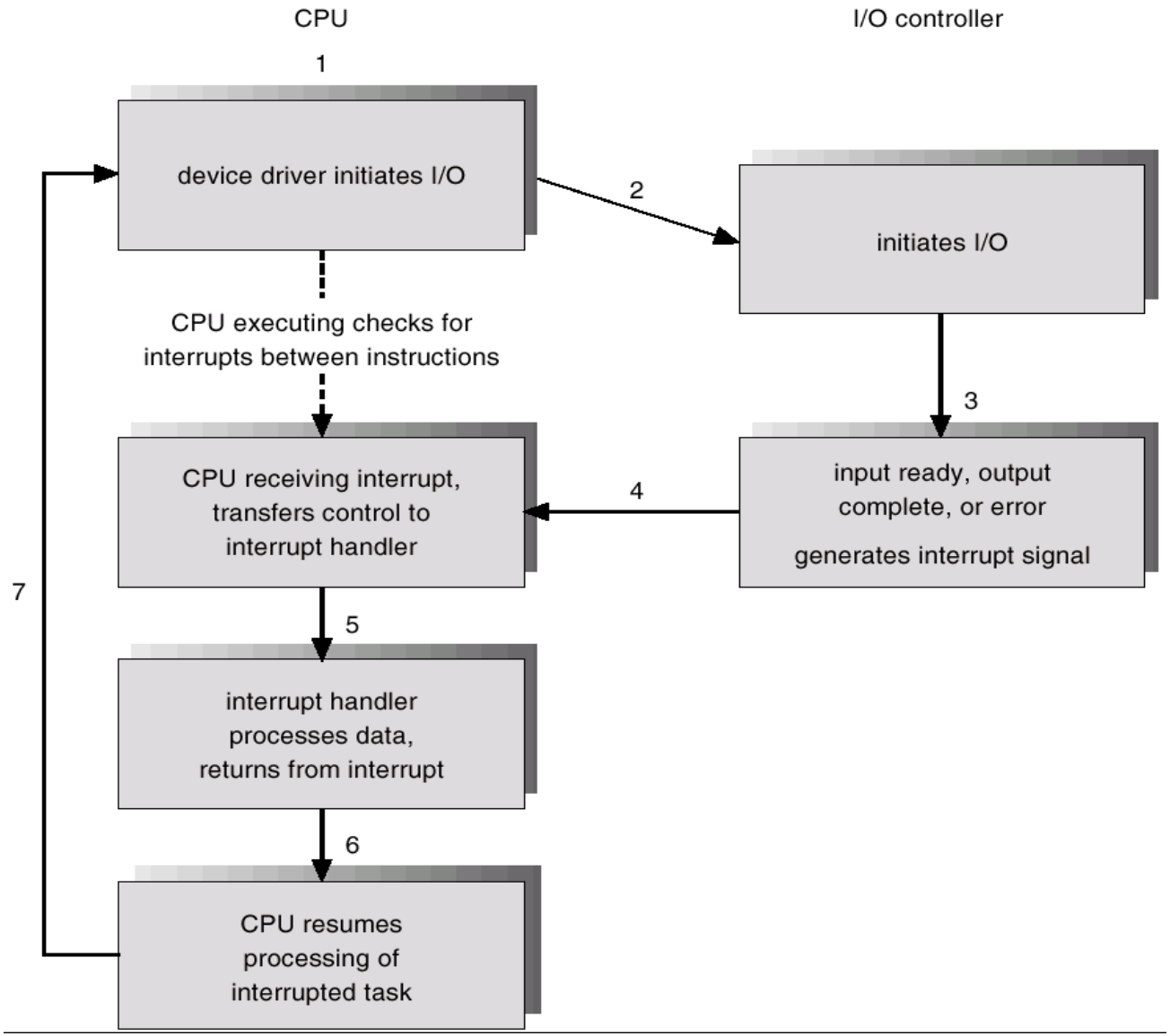
Polled I/O

- The simplest form form of I/O is to have the CPU do all the work. This method is called **programmed I/O.** The action followed by operating system are summarized in following manner. First the data are copied to the kernel. Then the operating systems enters a tight loop outputting the characters one at a time. The essential aspect of programmed I/O is that after outputting a character, the CPU continuously polls the device to see if it is ready to accept another one. This behavior is often called **polling** or **busy waiting.**

Interrupt - Driven I/O

- The basic interrupt mechanism works as follows. The CPU hardware has a wire called the interrupt-request line that the CPU senses after executing every instruction. When the CPU detects that a controller has asserted a signal on the interrupt request line, the CPU saves a small amount of state, such as the current value of the instruction pointer, and jumps to the interrupt-handler routine at a fixed address in memory. The interrupt handler determines the cause of the interrupt, performs the necessary processing, and executes a return from interrupt instruction to return the CPU to the execution state prior to the interrupt. We say that the device controller *raises an interrupt by asserting a* signal on the interrupt request line, the CPU *catches the interrupt and dispatches* to the interrupt handler, and the handler *clears the interrupt by servicing the* device.

Interrupt-Driven I/O Cycle



RAID

- Disk drives have continued to get smaller and cheaper, so it is now economically feasible to attach a large number of disks to a computer system. Having a large number of disks in a system presents opportunities for improving the rate at which data can be read or written, if the disks are operated in parallel. Furthermore, this setup offers the potential for improving the reliability of data storage, because redundant information can be stored on multiple disks. Thus, failure of one disk does not lead to loss of data. A variety of disk-organization techniques, collectively called redundant arrays of inexpensive disks (RAID), are commonly used to address the performance and reliability issues.

RAID LEVELS

RAID level 0 – Striping:

- In a RAID 0 system data are split up into blocks that get written across all the drives in the array. By using multiple disks (at least 2) at the same time, this offers superior I/O performance. This performance can be enhanced further by using multiple controllers, ideally one controller per disk.

Advantages:

- RAID 0 offers great performance, both in read and write operations. There is no overhead caused by parity controls.
- All storage capacity is used, there is no overhead.
- The technology is easy to implement.

Disadvantages:

- RAID 0 is not fault-tolerant. If one drive fails, all data in the RAID 0 array are lost. It should not be used for mission-critical systems.

RAID LEVELS

RAID level 1 – Mirroring:

- Data are stored twice by writing them to both the data drive (or set of data drives) and a mirror drive (or set of drives). If a drive fails, the controller uses either the data drive or the mirror drive for data recovery and continues operation. You need at least 2 drives for a RAID 1 array.

Advantages

- RAID 1 offers excellent read speed and a write-speed that is comparable to that of a single drive.
- In case a drive fails, data do not have to be rebuild, they just have to be copied to the replacement drive.
- RAID 1 is a very simple technology.

Disadvantages

- The main disadvantage is that the effective storage capacity is only half of the total drive capacity because all data get written twice.

RAID LEVELS

RAID level 5:

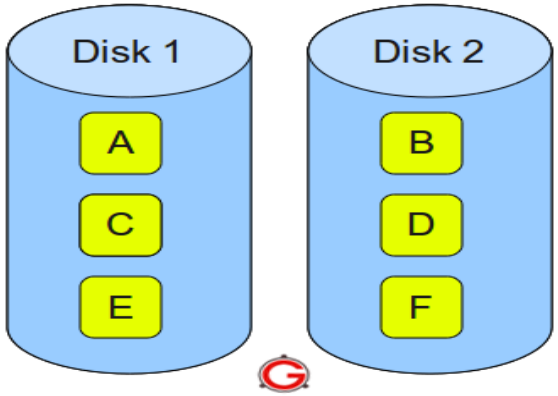
- RAID 5 is the most common secure RAID level. It requires at least 3 drives but can work with up to 16. Data blocks are striped across the drives and on one drive a parity checksum of all the block data is written. The parity data are not written to a fixed drive, they are spread across all drives, as the drawing below shows. Using the parity data, the computer can recalculate the data of one of the other data blocks, should those data no longer be available. That means a RAID 5 array can withstand a single drive failure without losing data or access to data.

Advantages:

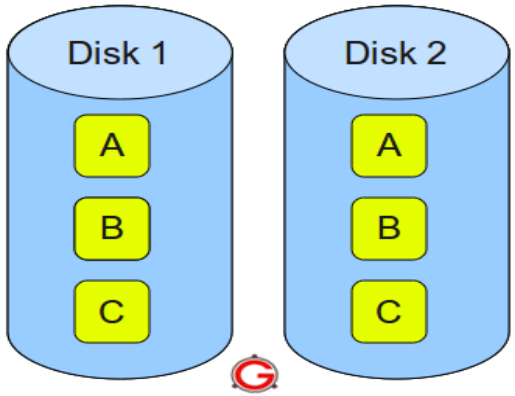
- Read data transactions are very fast while write data transactions are somewhat slower (due to the parity that has to be calculated).
- If a drive fails, you still have access to all data, even while the failed drive is being replaced and the storage controller rebuilds the data on the new drive.

Disadvantages:

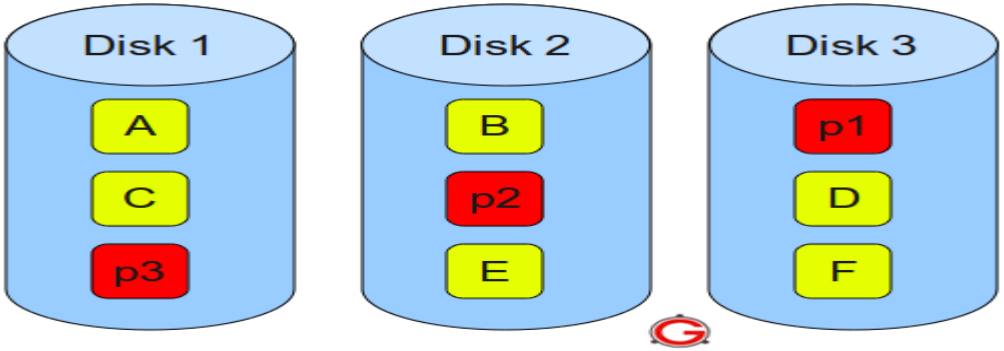
- Drive failures have an effect on throughput, although this is still acceptable.



RAID 0 – Blocks Striped. No Mirror. No Parity.



RAID 1 – Blocks Mirrored. No Stripe. No parity.



RAID 5 – Blocks Striped. Distributed Parity.

Device Drivers

- We know that device controller has some device registers used to give it commands or some device registers used to read out its status or both. The number of device registers and the nature of the commands vary radically from device to device. For example, a mouse driver has to accept information from the mouse telling how far it has moved and which buttons are currently depressed. In contrast, a disk driver may have to know all about sectors, tracks, cylinders, heads, arm motion, motor drives, and all the other mechanics of making disk work properly. Obviously, these drivers will be very different.
- As a consequence, each I/O device attached to a computer needs some device-specific code for controlling it. This code, called the **device driver,** is generally written by device's manufacturer and delivered along with the device. Since each operating system needs its own drivers, device manufacturers commonly supply drivers for several popular operating systems.

Device Drivers(contd.)

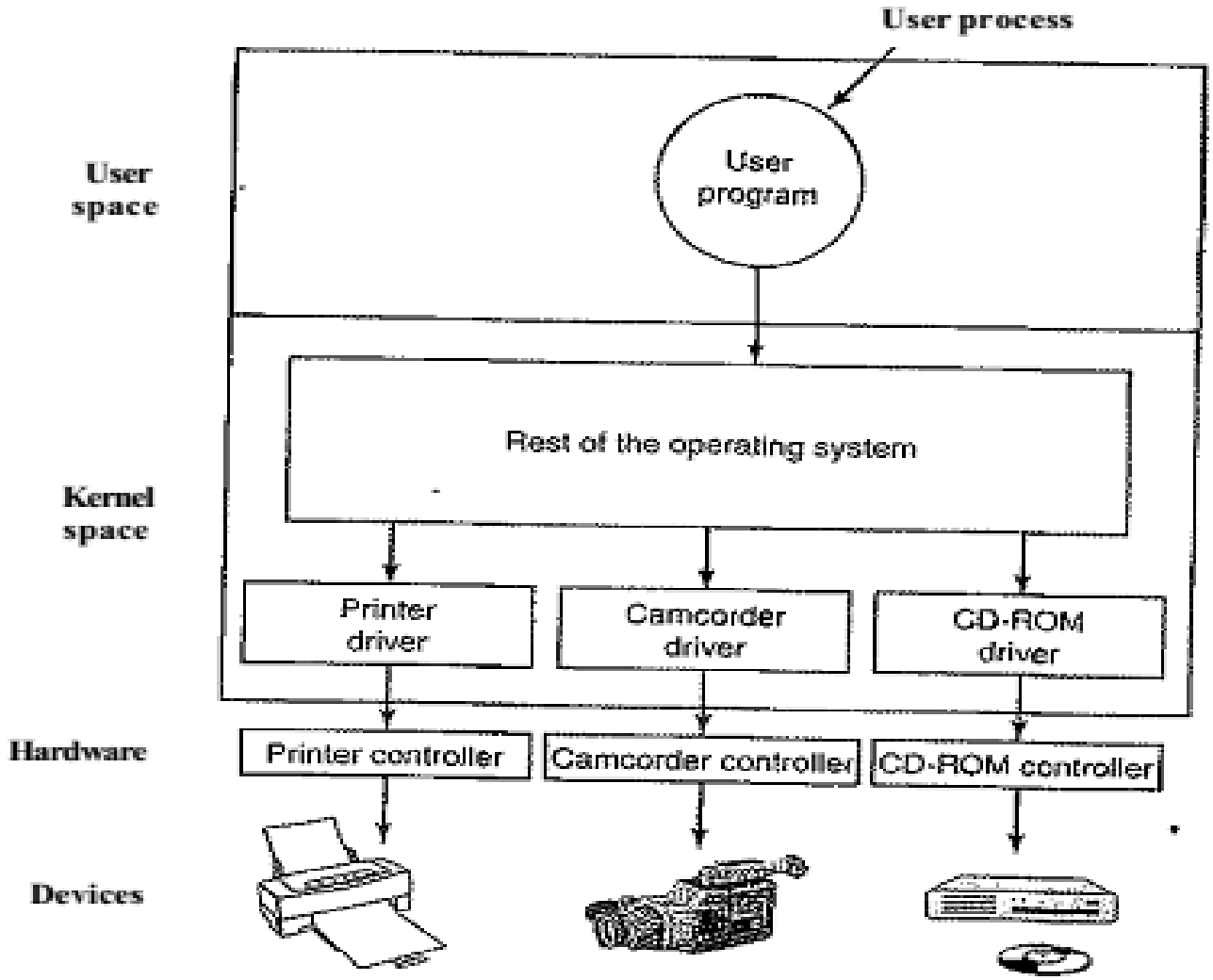


Figure 5-12. Logical positioning of device drivers. In reality all communication between drivers and device controllers goes over the bus.

Device-Independent I/O Software

- Although some of the I/O software is device specific, other parts of it are device independent. The exact boundary between the drivers and the device- independent software is system (and device) dependent, because some functions that could be done in a device-independent way may actually be done in the drivers, for efficiency or other reasons. The functions shown in Fig. below are typically done in the device-independent software.

Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

- The basic function of the device-independent software is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user-level software.

User Space I/O Software

- Although most of the I/O software is within the operating system, a small portion of it consists of libraries linked together with user programs, and even whole programs running outside the kernel. System calls, including the I/O system calls, are normally made by library procedures. When a C program contains the call

  count = write(fd, buffer, nbytes);

  the library procedure *write will be linked with the program and contained in the* binary program present in memory at run time. The collection of all these library procedures is clearly part of the I/O system.

Disk Hardware

- Disks come in a variety of types. The most common ones are the magnetic disks (hard disks and floppy disks). They are characterized by the fact that reads and writes are equally fast, which makes them ideal as secondary memory (paging, file systems, etc.). Arrays of these disks are sometimes used to provide highly reliable storage. For distribution of programs, data, and movies, various kinds of optical disks (CD-ROMs, CD-Recordables, and DVDs) are also important.

Magnetic Disks

- Magnetic disks are organized into cylinders, each one containing as many tracks as there are heads stacked vertically. The tracks are divided into sectors, with the number of sectors around the circumference typically being 8 to 32 on floppy disks, and up to several hundred on hard disks. The number of heads varies from 1 to about 16.
- Older disks have littie electronics and just deliver a simple serial bit stream. On these disks, the controller does most of the work. On other disks, in particular, **IDE (Integrated Drive Electronics)** and **SAT A (Serial ATA)** disks, the disk drive itself contains a microcontroller that does considerable work and allows the real controller to issue a set of higher-level commands. The controller often does track caching, bad block remapping, and much more.

Disk Scheduling

- 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.
- 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 Arm Scheduling Algorithm

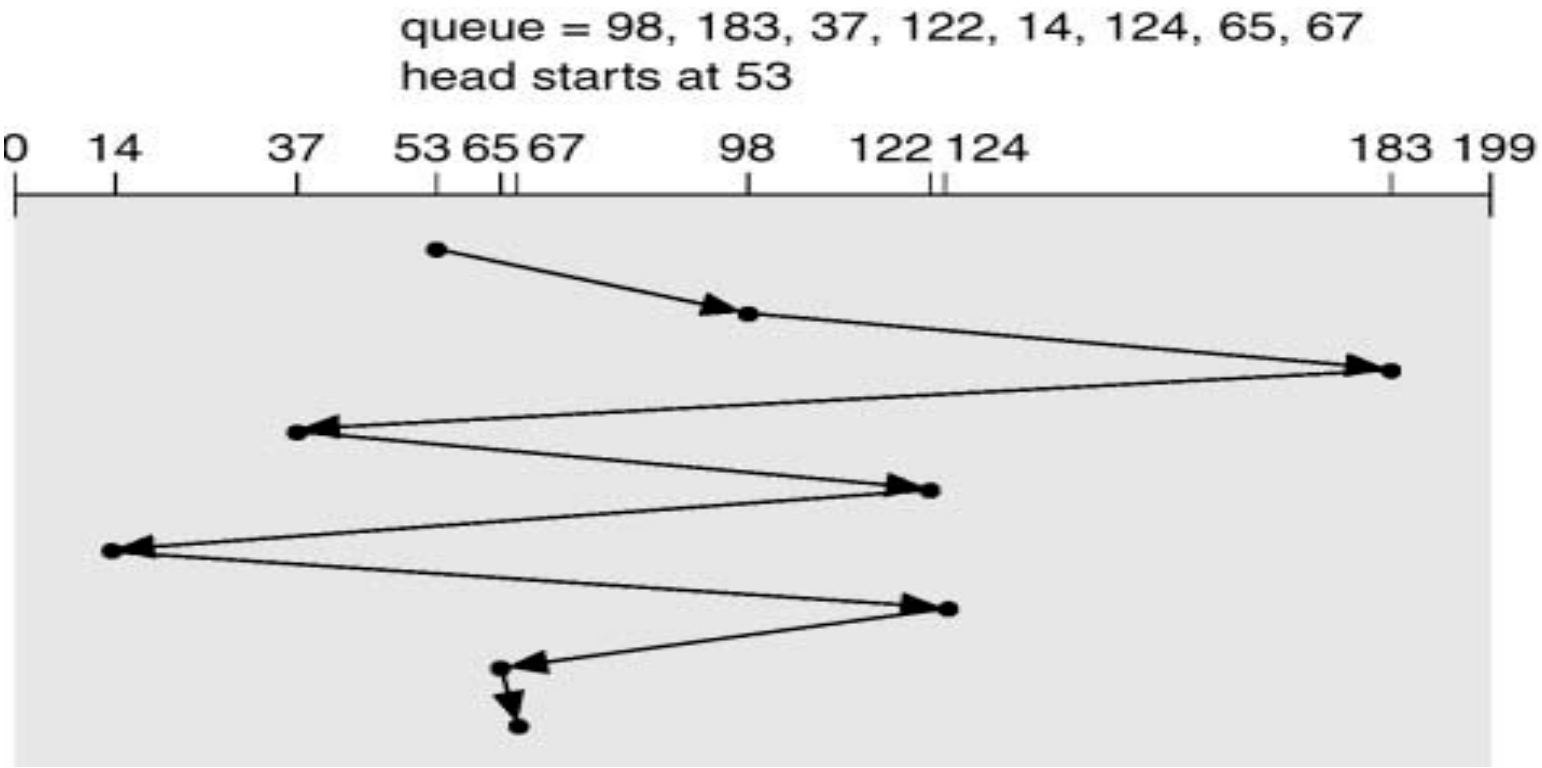
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

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

Head pointer 53

FCFS

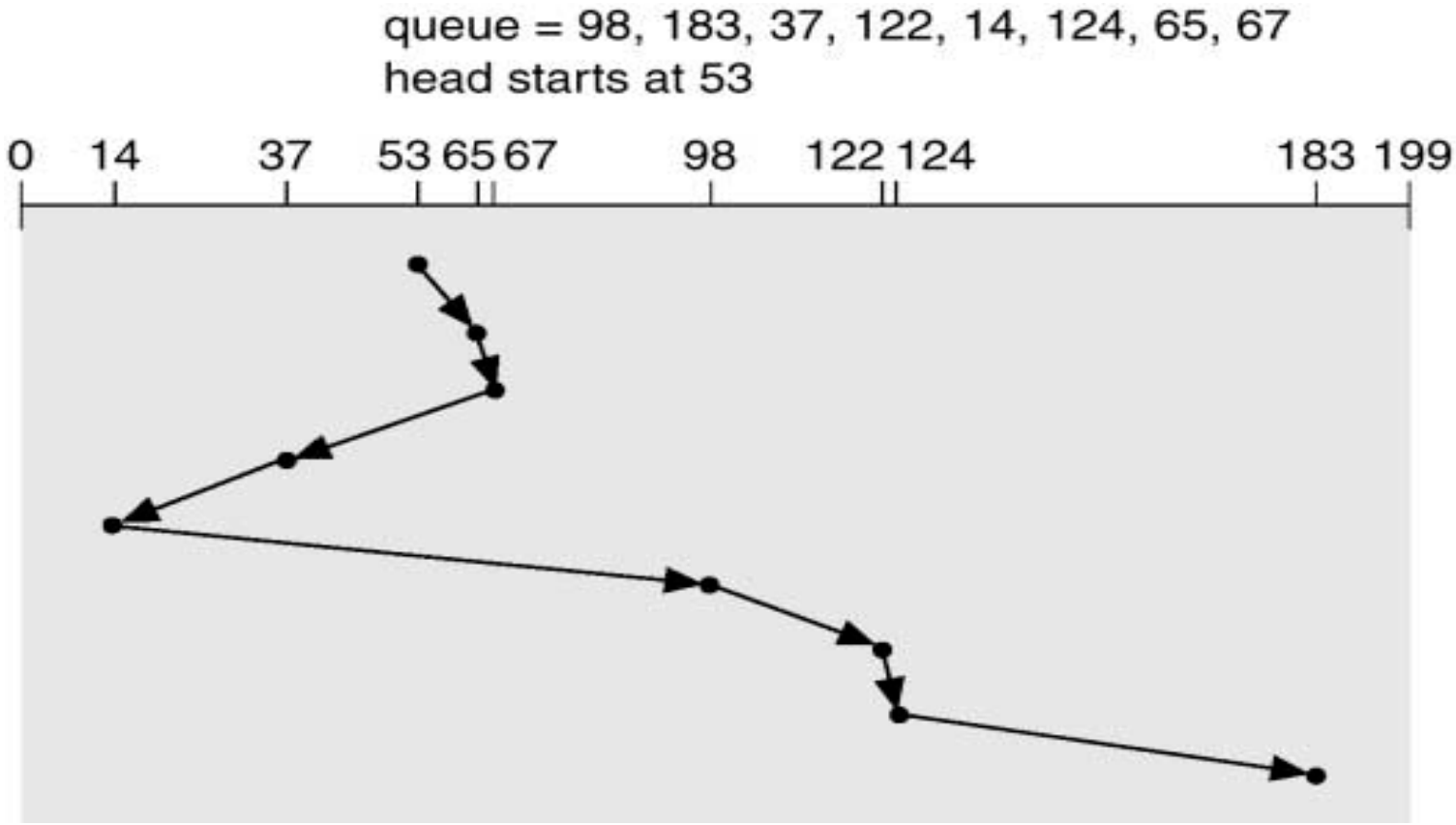
Illustration shows total head movement of 640 cylinders.



Shortest Seek First(SSF)

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

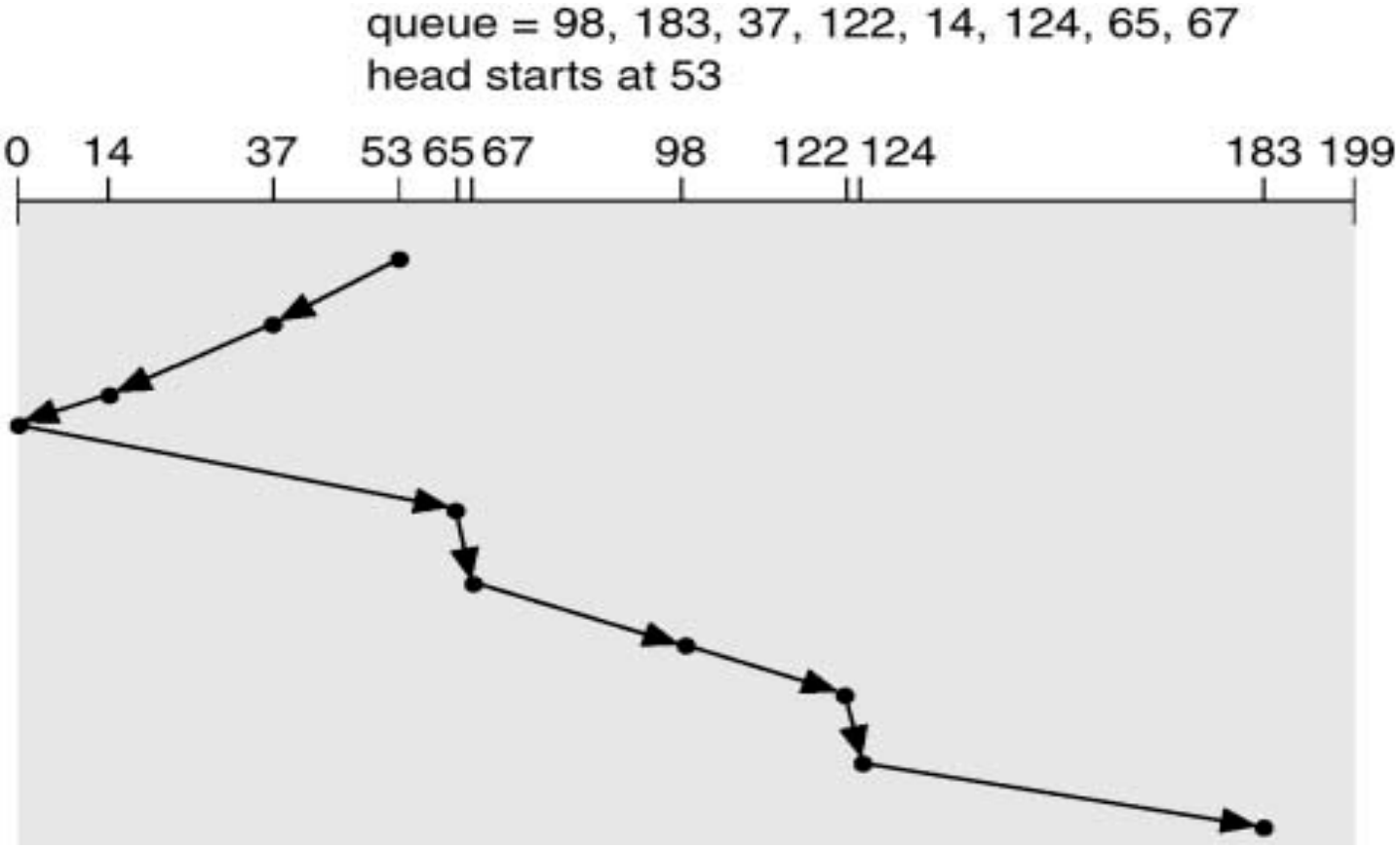
SSF (Cont.)


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

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.*
- Illustration shows total head movement of 208 cylinders.

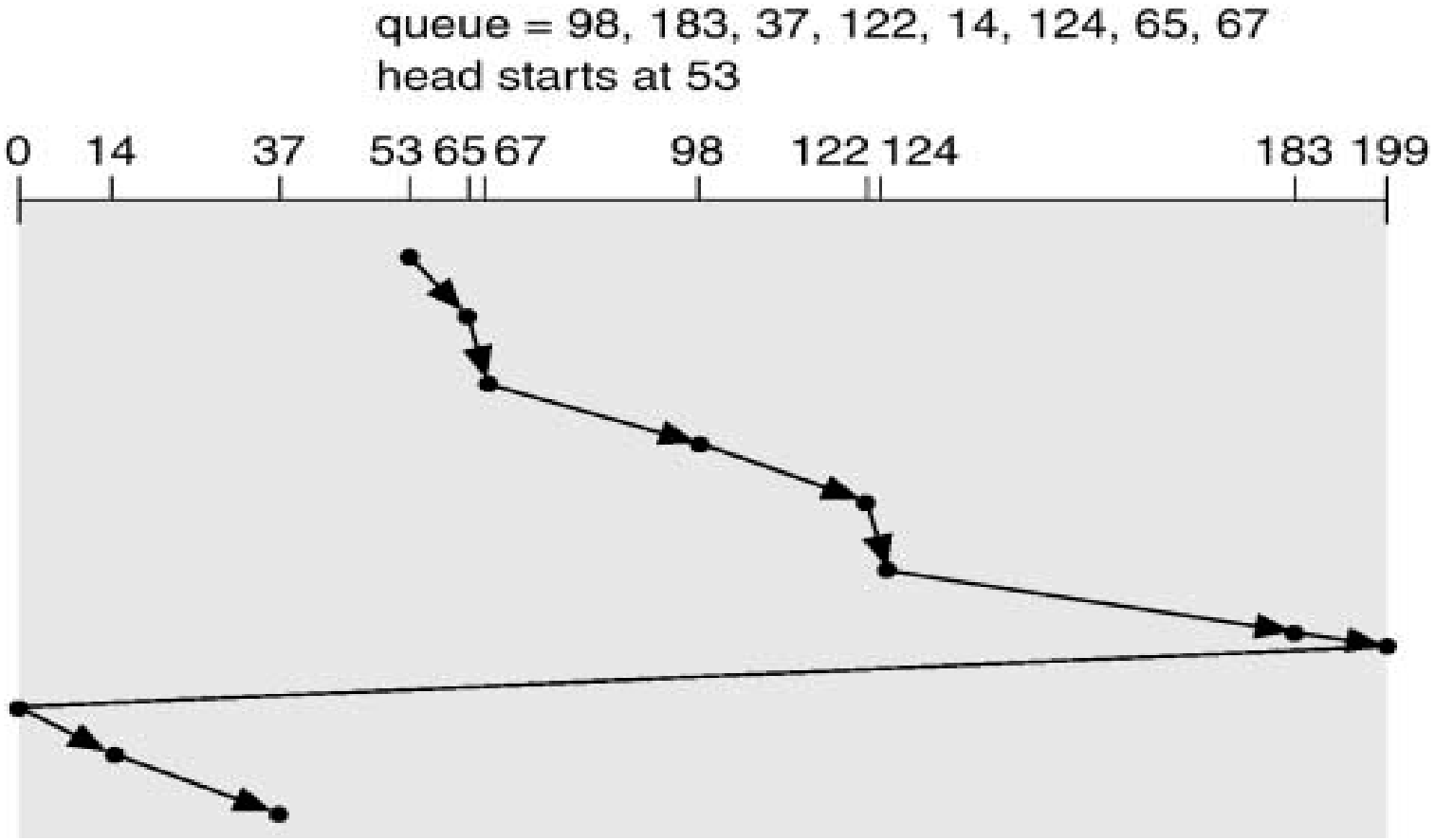
SCAN (Cont.)


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

C-SCAN

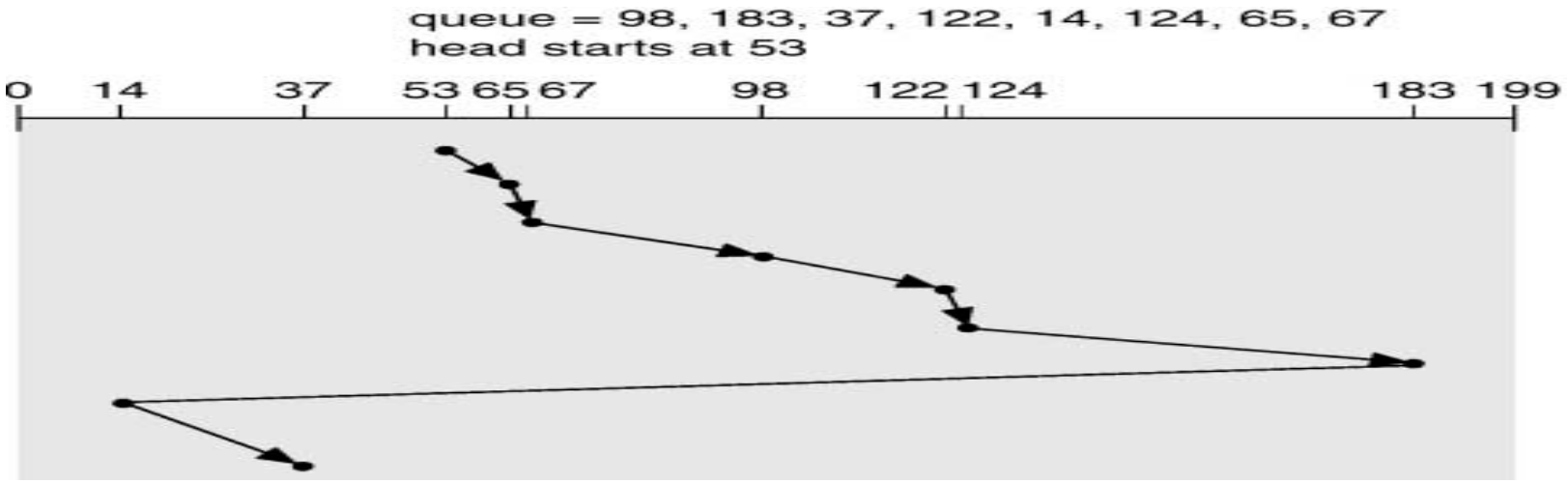
- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders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

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

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.



RAM Disks

- Refers to RAM that has been configured to simulate a disk drive. You can access files on a RAM disk as you would access files on a real disk. RAM disks, however, are approximately a thousand times faster than hard disk drives. They are particularly useful, therefore, for applications that require frequent disk accesses.
- Because they are made of normal RAM, RAM disks lose their contents once the computer is turned off. To use a RAM disk, therefore, you need to copy files from a real hard disk at the beginning of the session and then copy the files back to the hard disk before you turn the computer off. Note that if there is a power failure, you will lose whatever data is on the RAM disk. (Some RAM disks come with a battery backup to make them more stable.)
- A RAM disk is also called a *virtual disk* or a *RAM drive.*