

目录 Contents

- 1 Overviews
- 2 Process Description and Control
- 3 Threads and Kernel Architecture
- 4 Concurrency: Mutual Exclusion and Synchronization
- 5 Concurrency: Deadlock and Starvation
- 6 Memory Management and Virtual Memory
- 7 Uniprocessor Scheduling
- 8 **I/O Management and Disk Scheduling**
- 9 File Management

1

I/O Management and Disk Scheduling

- 1 Principles of I/O Hardware
- 2 Principles of I/O Software
- 3 I/O Buffering
- 4 Disk Scheduling
- 5 RAID
- 6 Disk Cache

2

8.1 Principles of I/O Hardware

1 I/O Devices Overviews

- I/O device features: There are many types of devices with different characteristics and different operation modes.
- The operating system controls all input/output devices and implements the following I/O functions:
 - Issue commands to devices, catch interrupts and handle errors
 - providing an interface independent of the rest (device independent interface)

3

Differences in I/O Devices

➤ Data rate(数据速率)

- May be differences of several orders of magnitude between the data transfer rates

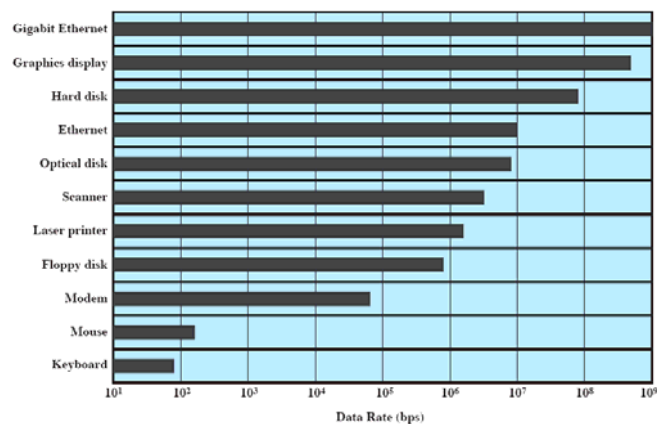


Figure 11.1 Typical I/O Device Data Rates

4

Differences in I/O Devices.

→ **Application(用途)**

- Disk used to store files requires file management software.
- Disk used to store virtual memory pages needs special hardware and software to support it
- Terminal used by system administrator may have a higher priority

→ **Complexity of control**

- the effect on the operating system is filtered by the complexity of the I/O module that controls the device

5

Differences in I/O Devices..

→ **Unit of transfer(传送单位)**

- Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk

→ **Data representation(数据表示)**

- Encoding schemes

→ **Error conditions(错误条件)**

- Devices respond to errors differently

6

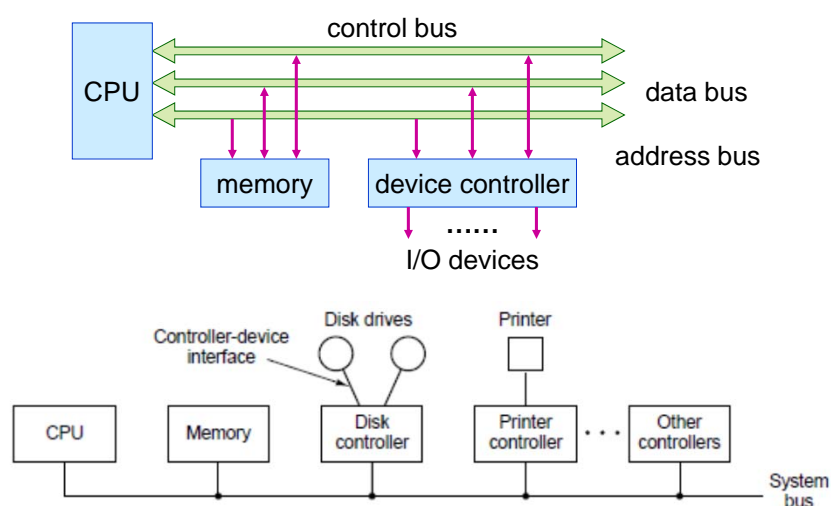
8.1 Principles of I/O Hardware

2 Organization of I/O

- ➔ **The host I/O system:** using specialized I/O computers called **I/O channels** taking some of the load off the main CPU.
- ➔ **Microcomputer I/O system:** most personal computers and servers use the bus model for communication between the CPU and the controllers. The I/O device is connected to the bus through the device controller. The CPU does not communicate directly with the I/O device, but with the device controller.

7

Organization of I/O.



8

Categories of I/O Devices

- ➔ External devices that engage in I/O with computer systems can be grouped into three categories:
- ➔ **Human readable:** suitable for communicating with the computer user
 - ➔ printers, terminals, video display, keyboard, mouse
- ➔ **Machine readable:** suitable for communicating with electronic equipment
 - ➔ disk drives, USB keys, sensors, controllers
- ➔ **Communication:** suitable for communicating with remote devices
 - ➔ modems, digital line drivers

9


Categories of I/O Devices.

- ➔ **Block devices:** the one that stores information in fixed-size blocks, each one with its own address. The essential property of a block device is that it is possible to read or write each block independently of all the other ones.
 - ➔ Common block sizes range from 512 bytes to 32768 bytes.
 - ➔ Disks are the most common block devices
- ➔ **Character 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(for pointing) and most other devices that are not disk-like can be seen as character devices

10


8.1 Principles of I/O Hardware

3 Evolution of the I/O Function

- ➔ Processor directly controls a peripheral device
 - ➔ Controller or I/O module is added
 - ➔ Processor uses programmed I/O without interrupts
 - ➔ Processor does not need to handle details of external devices
 - ➔ Controller or I/O module with interrupts
 - ➔ Processor does not spend time waiting for an I/O operation to be performed
- 

11

Evolution of the I/O Function.

- ➔ Direct Memory Access
 - ➔ Blocks of data are moved into memory without involving the processor
 - ➔ Processor involved at beginning and end only
 - ➔ I/O module is a separate processor
 - ➔ I/O processor
 - ➔ I/O module has its own local memory
 - ➔ Its a computer in its own right
- 

12

8.1 Principles of I/O Hardware

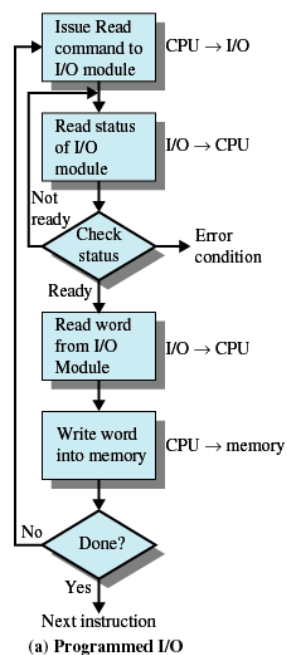
4 I/O Communication Techniques

- ➔ Three techniques are possible for I/O operations:
 - ➔ Programmed I/O
 - ➔ Interrupt-driven I/O
 - ➔ Direct memory access (DMA)

13

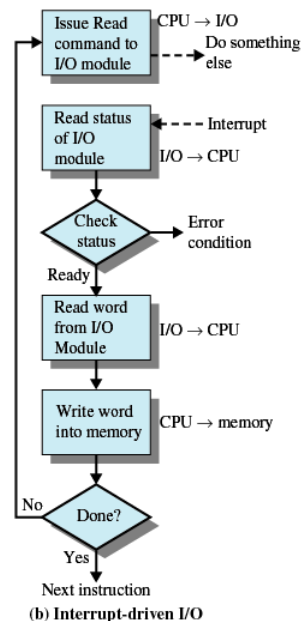
Programmed I/O

- ➔ Programmed I/O
 - ➔ **I/O module** performs the action, not the processor
 - ➔ Sets appropriate bits in the I/O status register
 - ➔ Processor checks status until operation is complete
 - ➔ Consumes a lot of processor time because every word read or written passes through the processor
- ➔ Process is **busy-waiting** for the operation to complete



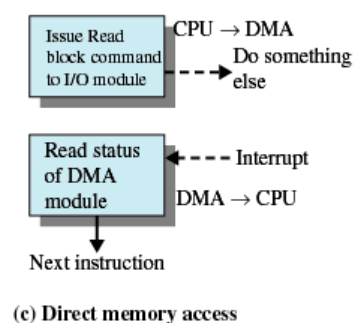
Interrupt-Driven I/O

- ➔ Interrupt-Driven I/O
 - ➔ I/O command is issued
 - I/O module issues interrupt request to processor when it is ready to exchange data
 - ➔ Processor continues executing instructions
 - Processor is interrupted
 - Processor saves **context of program executing** and begins executing **interrupt-handler**
 - ➔ I/O module sends an interrupt when done
- ➔ No needless waiting

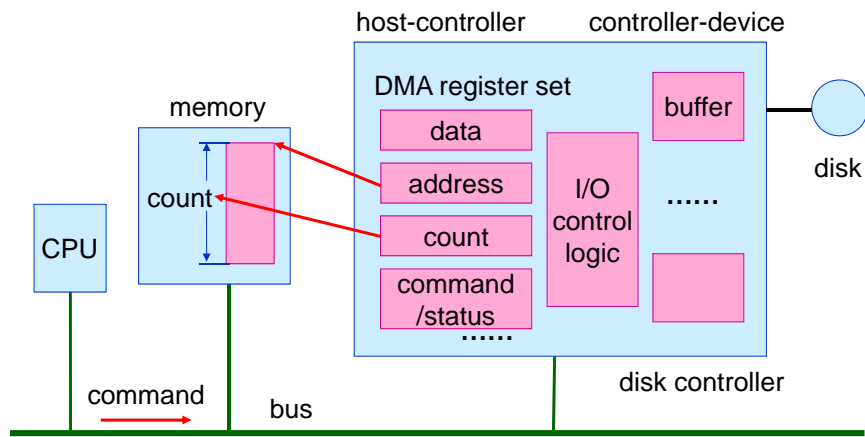


Direct Memory Access (DMA)

- ➔ Direct Memory Access (DMA)
 - ➔ I/O exchanges occur **directly** with memory
 - ➔ Processor grants I/O module **authority** to read from or write to memory
 - ➔ **Relieves** the processor responsibility for the exchange
- ➔ DMA module controls exchange of data between main memory and the I/O device
- ➔ Processor interrupted only after entire block has been transferred



Direct Memory Access (DMA).



17

Typical DMA Block

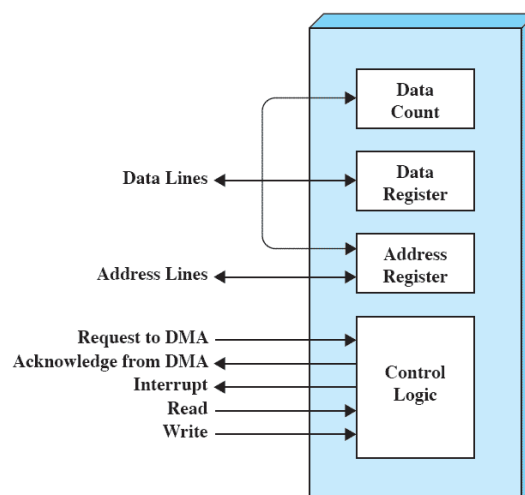
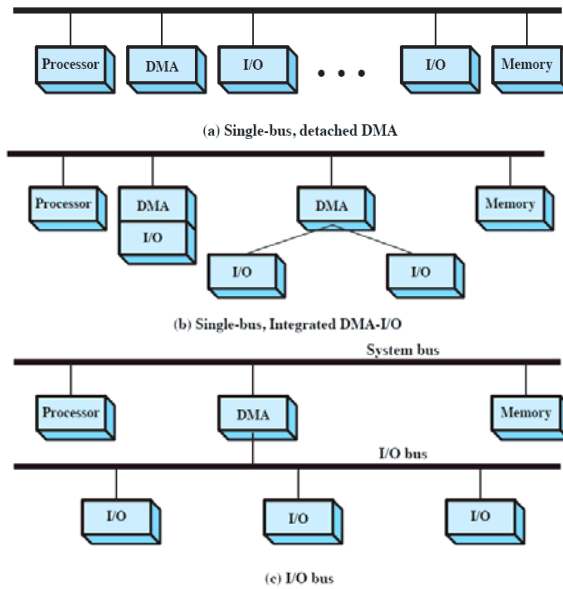


Figure 11.2 Typical DMA Block Diagram

18

DMA Alternative Configurations



19

Relationship Among Techniques

Table 11.1 I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

20

8.2 Principles of I/O Software

1 Design Objectives

→ Efficiency

- Most I/O devices extremely slow compared to main memory
- I/O cannot keep up with processor speed
 - Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
 - Swapping is used to bring in additional Ready processes which is an I/O operation

21

Design Objectives.

→ Generality(通用性)

- Desirable to handle all I/O devices in a uniform manner
- Hide most of the details of device I/O in lower-level routines so that processes and upper levels see devices in general terms such as read, write, open, close, lock, unlock

22

8.2 Principles of I/O Software

2 Logical Structure of I/O Function

→ Hierarchical philosophy: Layer

→ Logical I/O (逻辑I/O)

→ Deal with the device as a logical resource

→ Device I/O (设备I/O)

→ Requested operations are converted into appropriate sequences of I/O instructions, channel commands and controller orders

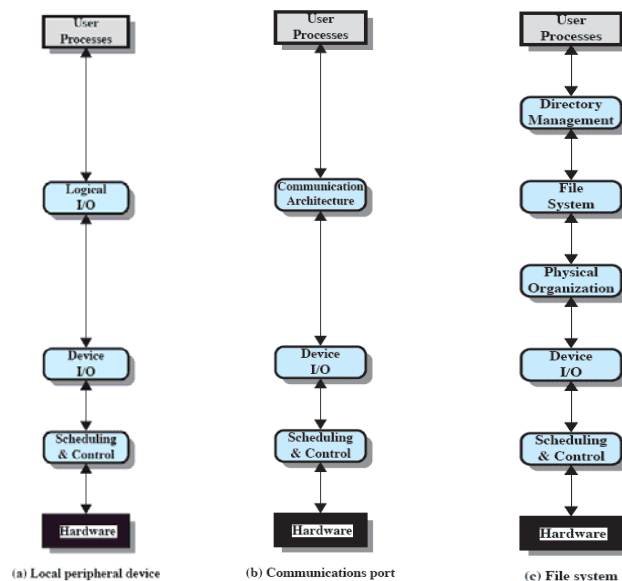
→ Scheduling and control (调度和控制)

→ Actual queuing and scheduling of I/O operations

→ Control of the operations

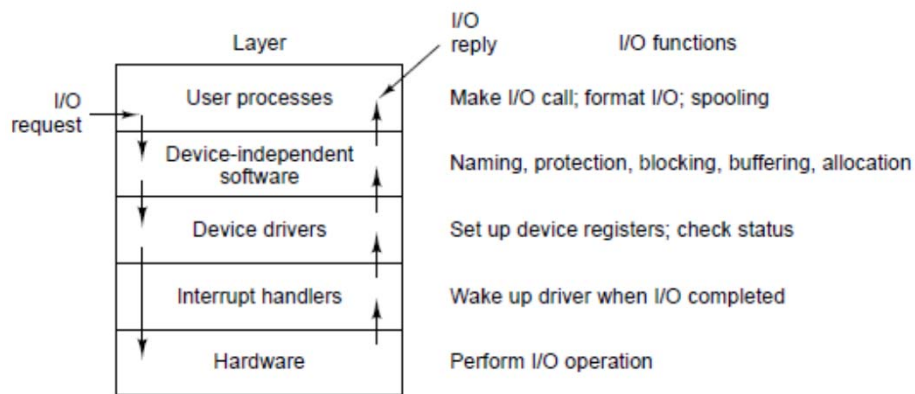
23

Model of I/O Organization



24

Layers of The I/O Software System



25

8.3 I/O Buffering

1 Cache Memory (高速缓存)

- ➔ intended to provide memory access time approaching that of the fastest memories available, and support a large memory size that the price of less expensive types of the semiconductor memories
 - ➔ **Invisible** to operating system
 - ➔ Increase the speed of memory
 - ➔ Processor speed is faster than memory speed
 - ➔ Exploit the principle of locality

26

Cache and Main Memory

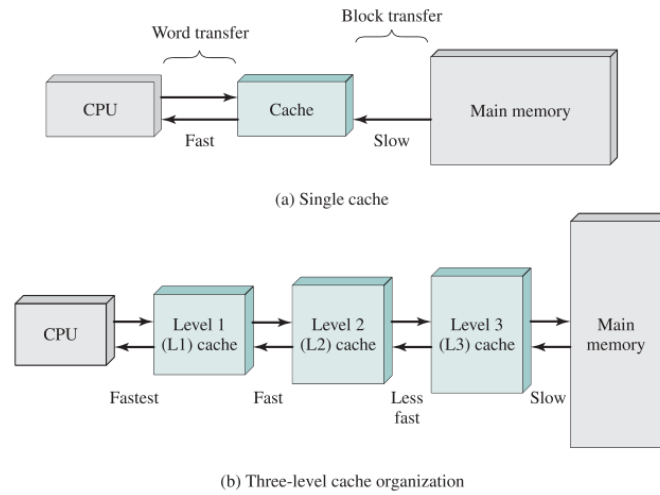


Figure 1.16 Cache and Main Memory

27

Cache/Main Memory Structure

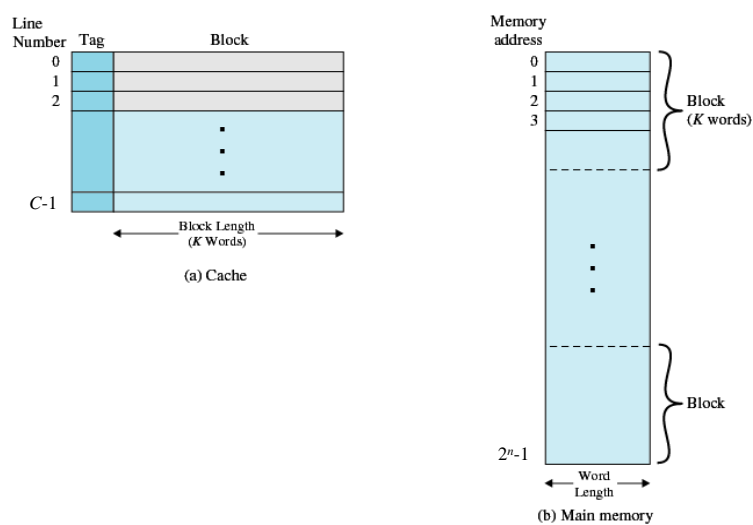


Figure 1.17 Cache/Main-Memory Structure

28

Cache Read Operation

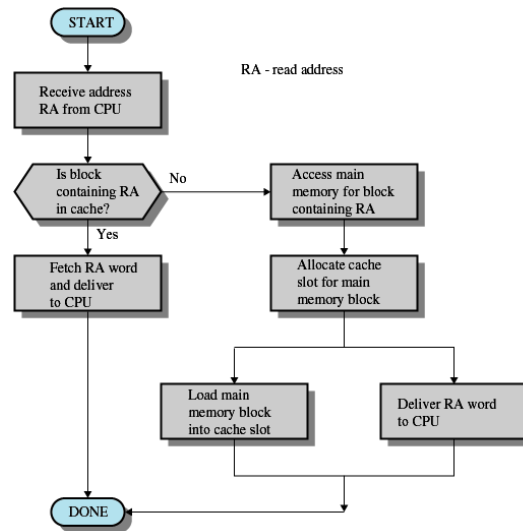


Figure 1.18 Cache Read Operation

29

Cache Design

- ➔ Cache size
 - ➔ Small caches have a significant impact on performance
- ➔ Block size: The unit of data exchanged between cache and main memory
 - ➔ Larger block size more hits until probability of using newly fetched data becomes less than the probability of reusing data that have to be moved out of cache
- ➔ Mapping function: determines which cache location the block will occupy

30

Cache Design.

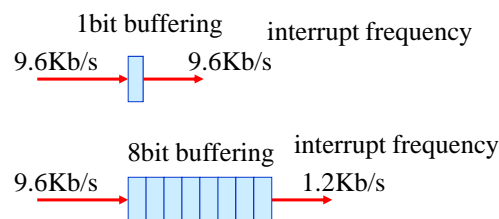
- ➔ Replacement algorithm: determines which block to replace
 - ➔ Least-Recently-Used (LRU) algorithm
- ➔ Write policy: when the memory write operation takes place
 - ➔ Can occur every time block is updated
 - ➔ Can occur only when block is replaced
 - ➔ Minimizes memory write operations
 - ➔ Leaves main memory in an obsolete state

31

8.3 I/O Buffering

2 I/O Buffering

- ➔ Reasons for buffering
 - ➔ Processes must wait for I/O to complete before proceeding
 - ➔ Avoid continuous use of the bus
 - ➔ Certain pages must remain in main memory during I/O



32

I/O Buffering.

➔ Block-oriented

- ➔ Information is stored in fixed sized blocks
- ➔ Transfers are made a block at a time
- ➔ Used for disks and tapes

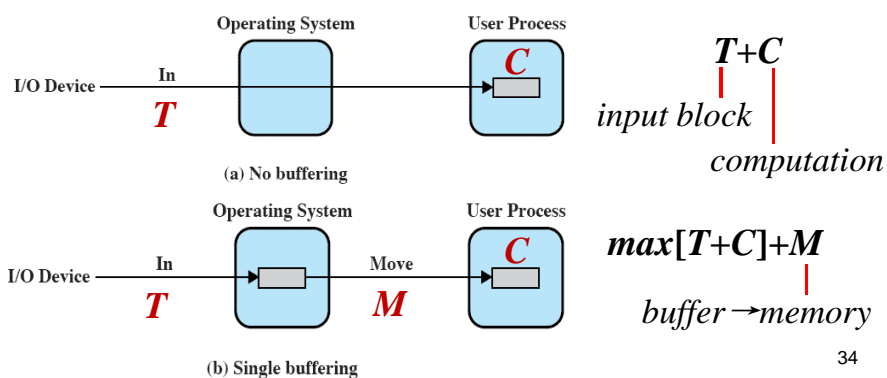
➔ Stream-oriented

- ➔ Transfer information as a stream of bytes
- ➔ Used for terminals, printers, communication ports, mouse and other pointing devices, and most other devices that are not secondary storage

33

Single Buffer

- ➔ No Buffer: without a buffer, the OS directly accesses the device when it needs
- ➔ Single Buffer: Operating system assigns a buffer in main memory for an I/O request



34

Block-oriented Single Buffer

- ➔ Input transfers made to buffer
- ➔ Block moved to user space when needed
- ➔ Another block is moved into the buffer
 - ➔ Read ahead
- ➔ User process can process one block of data while next block is read in
- ➔ Swapping can occur since input is taking place in system memory, not user memory
- ➔ Operating system keeps track of assignment of system buffers to user processes

35

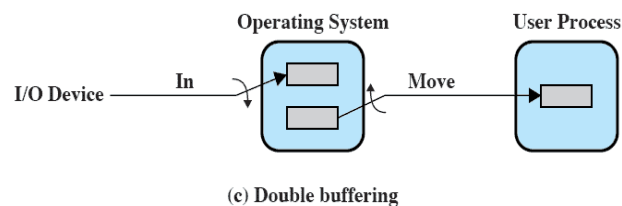
Stream-oriented Single Buffer

- ➔ Line-at-a-time operation: Used a line at time
 - ➔ appropriate for scroll-mode terminals (also called: dumb terminals 哑终端)
 - ➔ user input is one line at a time with a carriage return signaling the end of a line
 - ➔ output to the terminal is similarly one line at a time
- ➔ Byte-at-a-time operation
 - ➔ used on forms-mode terminals(表格终端), or other peripherals such as sensors and controllers
 - ➔ when each keystroke is significant

36

Double Buffer

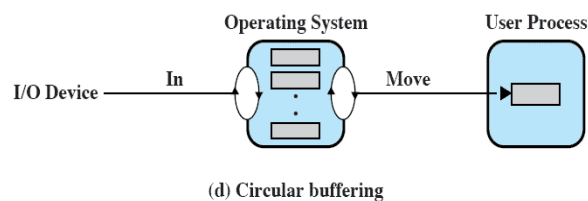
- ➔ Use two system buffers instead of one
- ➔ A process can transfer data to or from one buffer while the operating system empties or fills the other buffer
- ➔ Also known as buffer swapping



37

Circular Buffer

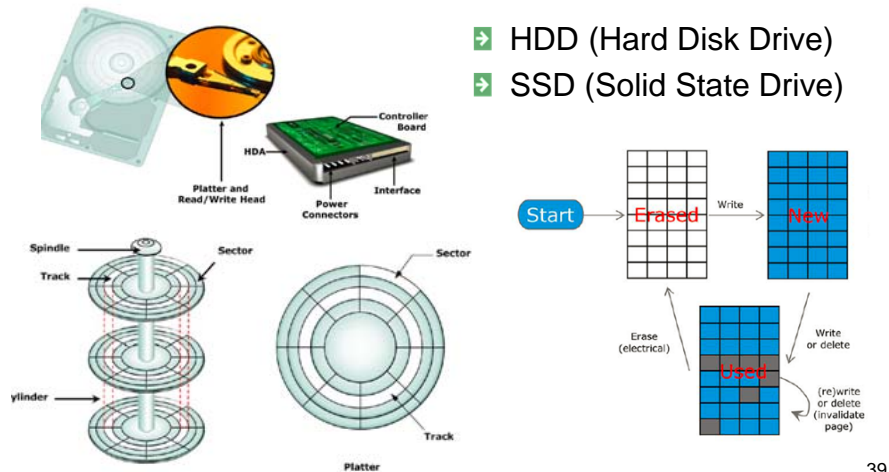
- ➔ More than two buffers are used
- ➔ Each individual buffer is one unit in a circular buffer
 - ➔ Used when I/O operation must keep up with process



38

8.4 Disk Scheduling

1 Disk Performance Parameters



39

Disk Performance Parameters.

- **Seek time(寻道时间)**: Time it takes to position the head at the desired track
- **Rotational delay** or **rotational latency(旋转延迟)**: Time it takes for the beginning of the sector to reach the head
- **Access time**: Sum of seek time and rotational delay
 - The time it takes to get in position to read or write

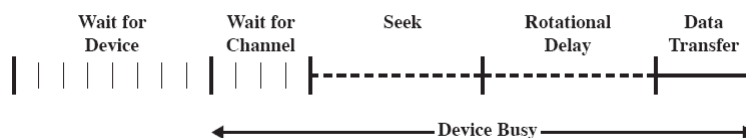


Figure 11.6 Timing of a Disk I/O Transfer

40

Time of Disk I/O Transfer

→ The total average access time can be expressed as:

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

seek time

Rotational delay
r: rotation speed(rps)

data transfer time
b: number of bytes to be transferrd
N: number of bytes on a track

41

8.4 Disk Scheduling

2 Disk Scheduling Policies

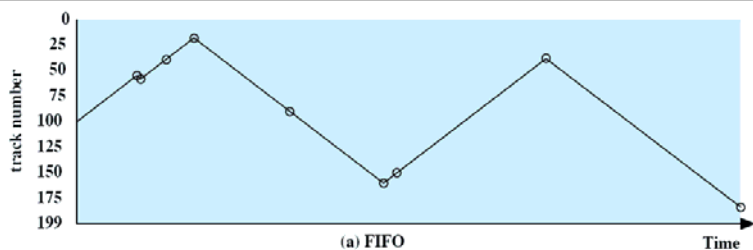
- **Seek time** is the reason for differences in performance
- For a single disk there will be a number of I/O requests, if requests are selected randomly, we will poor performance

42

First-in, First-out (FIFO)

- Process request sequentially
- Fair to all processes
- Approaches random scheduling(随机调度) in performance if there are many processes

→ Assume the disk head initially located at track 100, and a disk with 200 tracks, the requests tracks, in the order are: 55, 58, 39, 18, 90, 160, 150, 38, 184.



43

Priority

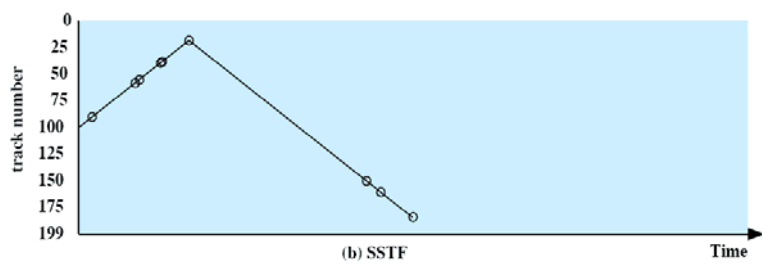
- Goal is not to optimize **disk use** but to meet other objectives
- Short batch jobs and interactive jobs may have higher priority
 - Provide good interactive response time
- Control of the scheduling is outside the control of disk management software

44

Shortest Service Time First (SSTF)

- Select the disk I/O request that requires the least movement of the disk arm from its current position
- Always choose the minimum Seek time
- Starvation

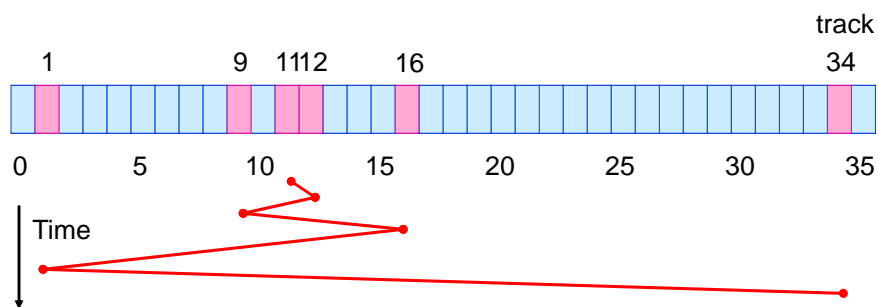
➤ 100 : 55, 58, 39, 18, 90, 160, 150, 38, 184.



45

Starvation

➤ 10(increasing) : 11, 1, 16, 34, 9, 12

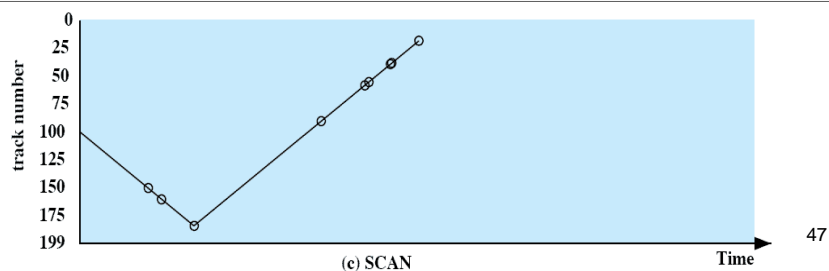


46

SCAN

- Disk arm moves **in one direction only**, satisfying all outstanding requests until it reaches the last track in that direction, then the direction is reversed.
- Also known as the elevator algorithm.
- Favors jobs whose requests are for tracks nearest to both innermost and outermost tracks, and the latest-arriving jobs.

➤ 100(increasing) : 55, 58, 39, 18, 90, 160, 150, 38, 184.

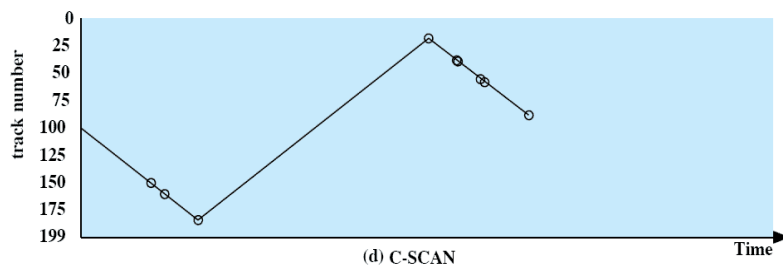


47

C-SCAN (Circular SCAN)

- Restricts scanning to one direction only
- When the last track has been visited in one direction, the arm is returned to the **opposite end** of the disk and the scan begins again

➤ 100(increasing) : 55, 58, 39, 18, 90, 160, 150, 38, 184.



48

N-Step-SCAN

- To avoid "**arm stickiness**"
- Segments the disk request queue into subqueues of length N
- Subqueues are processed one at a time, using SCAN
- While a queue is being processed new requests must be added to some other queue
- If fewer than N requests are available at the end of a scan, all of them are processed with the next scan

FSCAN

- Two queues
- One queue is empty for new requests

49

Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	Average seek length	27.5	Average seek length	27.8	Average seek length	35.8

8.5 RAID

➔ **RAID—Redundant Array of Independent Disks(独立磁盘冗余阵列)**, a standardized scheme for multiple-disk database design.

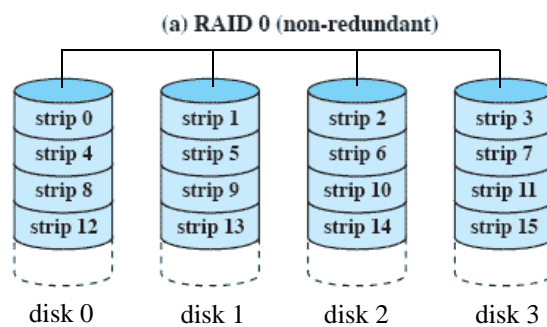
➔ **Characteristics**

- ➔ Set of physical disk drives viewed by the operating system as a single logical drive
- ➔ Data are distributed across the physical drives of an array
- ➔ Redundant disk capacity is used to store parity(奇偶校验) information

51

RAID 0 (Non-redundant)

- ➔ Logical disk is divided into strips(条带)
- ➔ User and system data are distributed across all of the disks in the array
- ➔ Not a true RAID because it does not include redundancy to improve performance or provide data protection

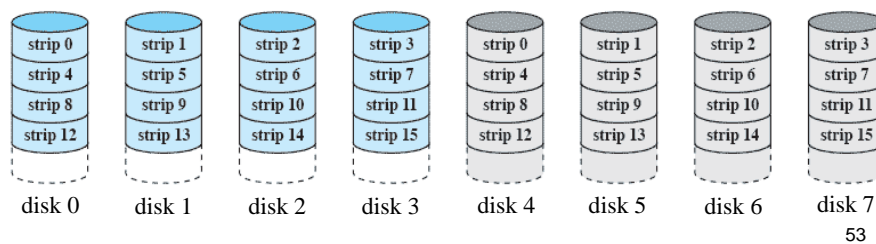


52

RAID 1 (Mirrored)

- Redundancy is achieved by the simple expedient of duplicating all the data
- There is no “write penalty”(写损失)
- When a drive fails the data may still be accessed from the second drive
- Principal disadvantage is the cost

(b) RAID 1 (mirrored)

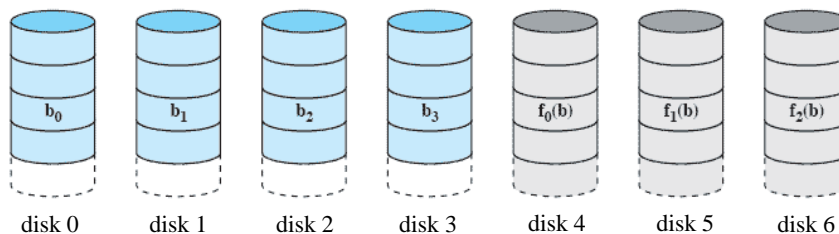


53

RAID 2 (Redundancy through Hamming code)

- Makes use of a parallel access technique
- Data striping (small) is used
- Typically a Hamming code is used
- Effective choice in an environment in which many disk errors occur

(c) RAID 2 (redundancy through Hamming code)

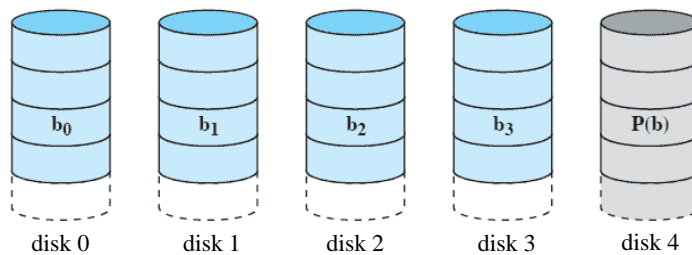


54

RAID 3 (Bit-interleaved parity)

- Requires only a single redundant disk, no matter how large the disk array
- Employs parallel access, with data distributed in small strips
- Can achieve very high data transfer rates

(d) RAID 3 (bit-interleaved parity)

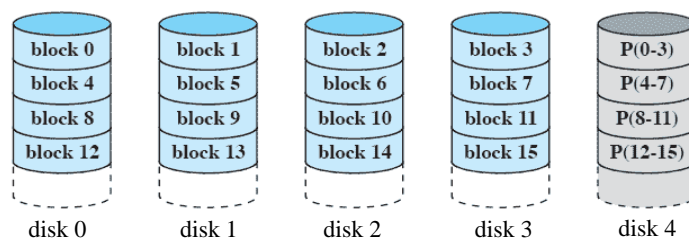


55

RAID 4 (Block-level parity)

- Makes use of an independent access technique
- A bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip on the parity disk
- Involves a write penalty when an I/O write request of small size is performed

(e) RAID 4 (block-level parity)

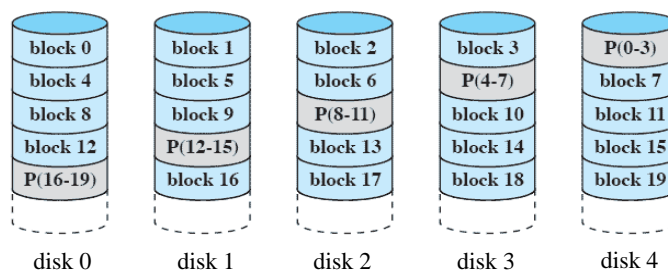


56

RAID 5 (Block-level distributed parity)

- Similar to RAID-4 but distributes the parity bits across all disks
- Typical allocation is a round-robin scheme
- Has the characteristic that the loss of any one disk does not result in data loss

(f) RAID 5 (block-level distributed parity)

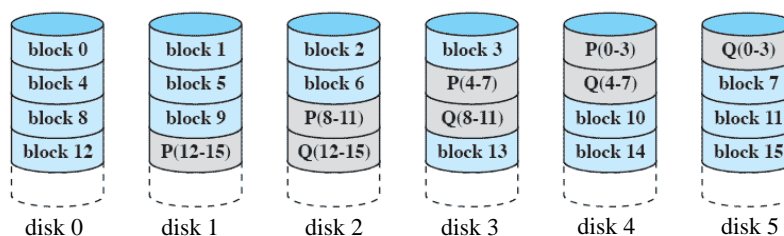


57

RAID 6 (Dual redundancy)

- Two different parity calculations are carried out and stored in separate blocks on different disks
- Provides extremely high data availability
- Incurs a substantial write penalty because each write affects two parity blocks

(g) RAID 6 (dual redundancy)



58

8.6 Disk Cache

- ➔ **Cache memory**: is used to apply to a memory that is smaller and faster than main memory and that is interposed between main memory and the processor
 - ➔ Reduces average memory access time by exploiting the principle of locality
- ➔ **Disk cache**: is a buffer in main memory for disk sectors
 - ➔ Contains a copy of some of the sectors on the disk

59

Disk Cache.

- ➔ When an I/O request is made for a particular sector, a check is made to determine if the sector is in the disk cache
 - ➔ **yes**: the request is satisfied via the cache
 - ➔ **No**: the requested sector is read into the disk cache from the disk

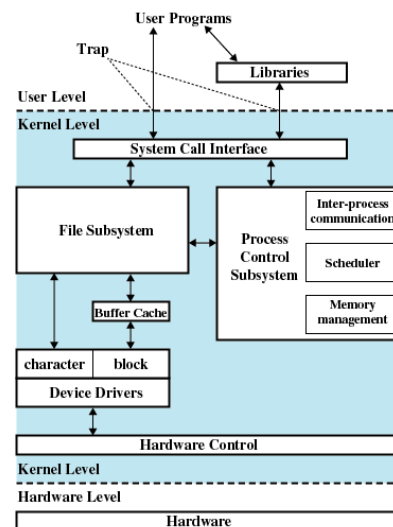


Figure 2.15 Traditional UNIX Kernel [BACH86]

Least Recently Used

- ➔ The block that has been in the cache the longest with no reference to it is replaced
- ➔ The cache consists of a stack of blocks and a stack of pointers is used
 - ➔ Most recently referenced block is on the top of the stack
 - ➔ When a block is referenced or brought into the cache, it is placed on the top of the stack
 - ➔ The block on the bottom of the stack is removed when a new block is brought in
- ➔ Blocks don't actually move around in main memory

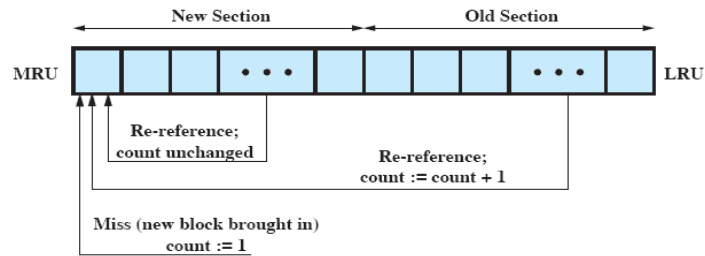
61

Least Frequently Used

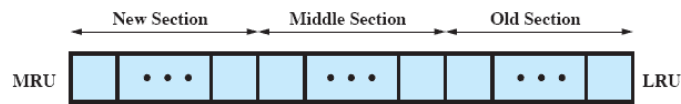
- ➔ The block that has experienced the fewest references is replaced
- ➔ A counter is associated with each block
 - ➔ Counter is incremented each time block accessed
 - ➔ Block with smallest count is selected for replacement
- ➔ Some blocks may be referenced many times in a short period of time and the reference count is misleading

62

Frequency-Based Replacement



(a) FIFO



(b) Use of three sections

63

Disk Cache Performance

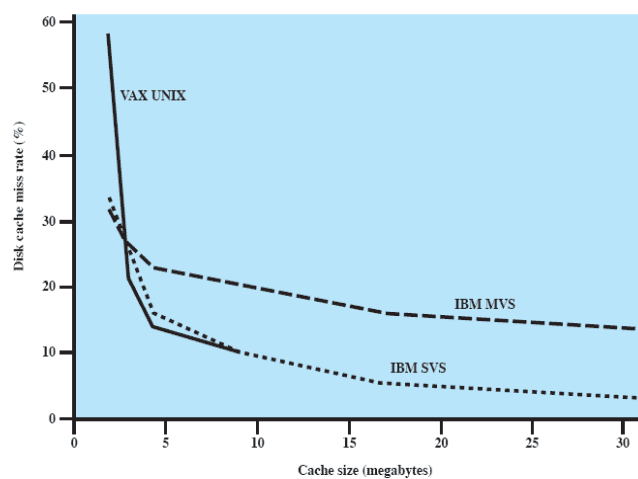


Figure 11.10 Some Disk Cache Performance Results Using LRU

64

Disk Cache Performance.

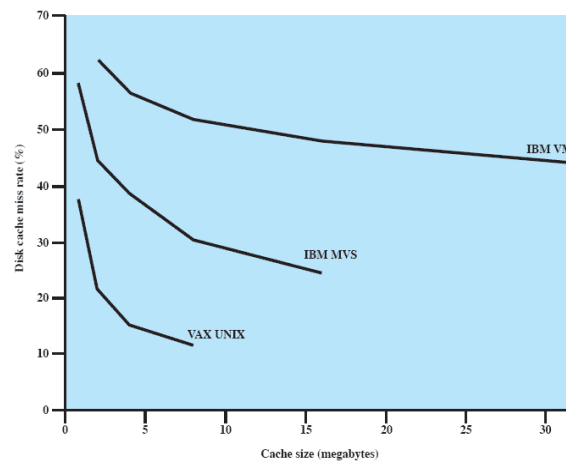


Figure 11.11 Disk Cache Performance Using Frequency-Based Replacement [ROBI90]

65

Terminology

- ➔ I/O Communication Techniques
 - ➔ Programmed I/O
 - ➔ Interrupt-driven I/O
 - ➔ Direct memory access (DMA)
- ➔ I/O channel
- ➔ logical I/O; device I/O
- ➔ I/O buffering
 - ➔ single buffer
 - ➔ double buffer
 - ➔ circular buffer
- ➔ seek time; rotational delay; access time; transfer time

66

Terminology

- Disk Scheduling Policies
 - first-in, first-out (FIFO)
 - priority
 - shortest seek time first (SSTF)
 - scan
 - c-scan
 - N-step-scan; F-scan
- Redundant Array of Independent Disks(RAID)
- disk cache