1.1What are the three main purposes of an operating system?

(1)帮助执行用户程序(2)管理软硬件资源

（3）为用户提供操作接口（4）组织用户更好地使用计算机

1.2 List the four steps that are necessary to run a program on a completely dedicated machine.
Preprocessing > Processing > Linking > Executing.

答：（1）预约计算机时间（2)手动把程序加载到内存

（3）加载起始地址并开始执行 （4）在电脑的控制台监督和控制程序的执行

1.6 Define the essential properties of the following types of operating systems:

a. Batch
b. Interactive
c. Time sharing
d. Real time
e. Network
f. Distributed

答：A.批处理系统：成批处理作业，用户脱机工作，单、多道程序运行，适合处理需要很少交互的大型工件； B.交互式系统：交互性，及时性 C.分时系统：同时性，交互性，及时性，独立性 D.实时系统 ：对时间有严格要求，外部事件驱动方式，响应及时，容错-双机备份，可靠性高，通常为特殊用途提供专用系统 E、网络操作系统：网络通信，可实现无差错的数据传输；共享软硬件；网络管理（比如安全控制);网络服务 F、分布式系统：多台分散的计算机经互联网连接而成的系统，处理器不共享内存和一个时钟，每个处理器有自己的内存，它们通过总线互相交流。

1.7 We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the operating system to forsake this principle and to“waste” resources? Why is such a system not really wasteful?

答：单用户系统，它应该最大化地为用户使用，一个GUI（图形化用户接口）可能会浪费CPU周期，但是它优化了用户和系统的交互。

2.2 How does the distinction between monitor mode and user mode function as a rudimentary form of protection (security) system?

答：monitor mode(管理状态，也是特权状态，可以执行全部指令，包括特权指令和非特权指令，访问所有资源并且具有改变处理器状态的能力）

user mode （用户状态，也叫目态，只能执行非特权指令)

注解：rudimentary(基本的，初步的) distinction（区别）

特权指令(privileged instructions):只提供给操作系统的核心程序使用，不给用户提供。

2.3 What are the differences between a trap and an interrupt? What is the use of each function?

答：陷入（trap)是由处理器正在执行的指令导致的，一条指令执行期间允许响应陷入，通常陷入处理程序提供的服务是当前进程所需要的。一般发生在软件层。

中断（interrupt)是由于与现行指令无关的中断信号发出的，通常在两条机器指令之间才可以响应中断，一般来说，中断处理程序提供的服务不是当前进程所需要的。一般发生在硬件里。

陷入可以用来调用操作系统程序，寻找算术错误

中断用于标记一个I/O设备的完成，用来消除设备轮询。

2.5 Which of the following instructions should be privileged?

a. Set value of timer.
b. Read the clock.
c. Clear memory.
d. Turn off interrupts.
e. Switch from user to monitor mode.

答案：a,c,d,e

特权指令：（1）允许和禁止中断，控制中断禁止屏蔽位（2）在进程间切换处理（3）存取用于主存保护的寄存器（4）执行I/O操作(5)停止一个中央处理机的工作(6)清理内存(7)设置时钟（8）建立存储键（9）加载PSW

2.8 Protecting the operating system is crucial to ensuring that the computer system operates correctly. Provision of this protection is the reason behind dual-mode operation, memory protection, and the timer. To allow maximum flexibility, however, we would also like to place minimal constraints on the user.

The following is a list of operations that are normally protected. What is the *minimal* set of instructions that must be protected?

a. Change to user mode.
b. Change to monitor mode.
c. Read from monitor memory.
d. Write into monitor memory.
e. Fetch an instruction from monitor memory.
f. Turn on timer interrupt.
g. Turn off timer interrupt.

答案：b,c,d,g

3.6 List five services provided by an operating system. Explain how each provides convenience to the users. Explain also in which cases it would be impossible for user-level programs to provide these services.

答：（1)**执行程序**，操作系统加载文件的目录到内存中并开始执行。在用户程序不能合理分配CPU时间的情况下不能提供该项服务。

（2）**I/O操作**，程序运行过程中需要I/O设备上的数据时，可以通过I/O命令或I/O指令，请求操作系统的服务。

（3）**文件系统操作**，文件系统让用户按文件名创建，读写，修改，删除文件，使用方便，安全可靠。当设计多用户访问或共享文件时，操作系统还提供信息保护机制。

（4）**通信服务**，在很多情况下，一个进程要与另外的进程交换信息，进程通信可以借助共享内存实现，也可使用消息消息传送技术实现。

（5）**错误检测**，OS可以不做和处理各种硬件和软件造成的差错和异常，并让他们造成的影响缩小在最小范围内。

3.7 What is the purpose of system calls?

答:系统调用（system calls)的目的：扩充机器功能，增强系统能力，方便用户使用

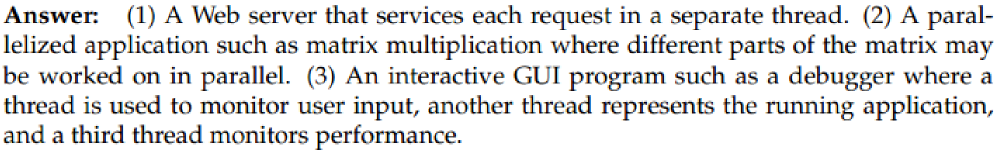
3.10 What is the purpose of system programs?

答：系统程序通过系统调用可以访问系统资源，调用操作系统。它提供基本的服务给用户，这样用不需要自己编写程序便可以解决问题。

4.1 MS-DOS provided no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.

答：（1）采取共享时间的方法允许每个进程都可访问系统。这种方法包括剥夺不主动放弃CPU 的进程和可重入的内核（2）进程和系统资源必须有保护，互相隔离开。每个进程可使用的内存有限，同时在设备上(比如磁盘)的操作也是有限的（3）关注内核，防止进程间出现死锁

5.1 Provide two programming examples of multithreading giving improve performance over a single-threaded solution.



Answer: (1) A Web server that services each request in a separate thread. (2) A parallelized application such as matrix multiplication where different parts of the matrix may be worked on in parallel. (3) An interactive GUI program such as a debugger where a thread is used to monitor user input, another thread represents the running application, and a third thread monitors performance.

5.3 What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

答：（1）在KLT(内核级线程）：和传统的基于进程操作系统中，大多数系统调用将阻塞进程，因此当线程执行一个系统调用是，不仅该线程被阻塞，而且进程内所有线程都会被阻塞，但是ULT中，可以选择另一个线程运行（2)在纯ULT中，多线程应用不能利用多重处理的优点，内核在一段时间里，分配一个进程进占用CPU。

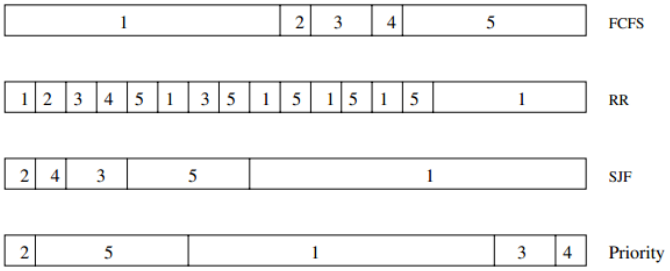
6.3 Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	Burst	Time Priority
*P*1	10	3
*P*2	1	1
*P*3	2	3
*P*4	1	4
*P*5	5	2

The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0.
a. Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling.
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?
c. What is the waiting time of each process for each of the scheduling algorithms in part a?
d. Which of the schedules in part a results in the minimal average waiting time (over all processes)?

Answer:

a. The four Gantt charts are


1 | 2 | 3 | 4 | 5 FCFS
1 | 2 | 3 | 4 | 5 | 1 | 3 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 RR
2 | 4 | 3 | 5 | 1 SJF
2 | 5 | 1 | 3 | 4 Priority

b. Turnaround time

	FCFS	RR	SJF	Priority
P_1	10	19	19	16
P_2	11	2	1	1
P_3	13	7	4	18
P_4	14	4	2	19
P_5	19	14	9	6

c. Waiting time (turnaround time minus burst time)

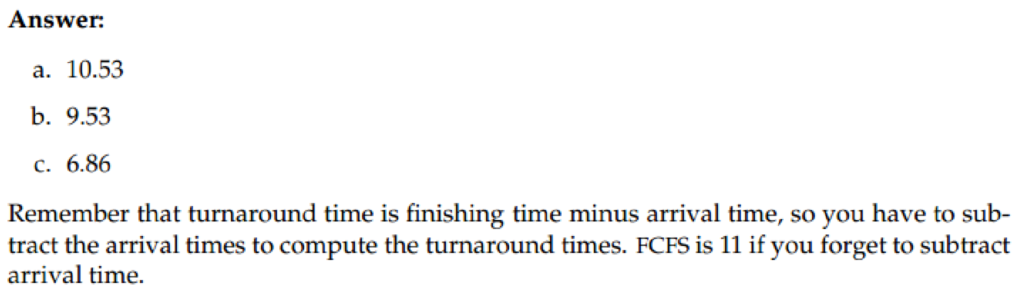
	FCFS	RR	SJF	Priority
P_1	0	9	9	6
P_2	10	1	0	0
P_3	11	5	2	16
P_4	13	3	1	18
P_5	14	9	4	1

d. Shortest Job First

6.4 Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time. In answering the questions, use nonpreemptive scheduling and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?
b. What is the average turnaround time for these processes with the SJF scheduling algorithm?
c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process *P*1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes *P*1 and *P*2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.



Answer:

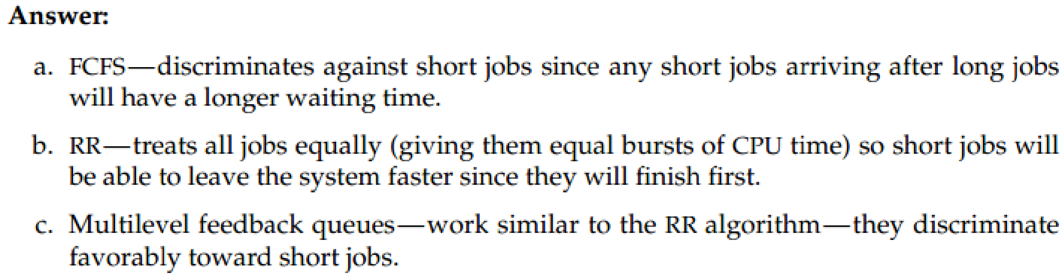
a. 10.53

b. 9.53

c. 6.86

Remember that turnaround time is finishing time minus arrival time, so you have to subtract the arrival times to compute the turnaround times. FCFS is 11 if you forget to subtract arrival time.

6.10 Explain the differences in the degree to which the following scheduling algorithms discriminate in favor of short processes:
a. FCFS
b. RR
c. Multilevel feedback queues



Answer:

a. FCFS—discriminates against short jobs since any short jobs arriving after long jobs will have a longer waiting time.

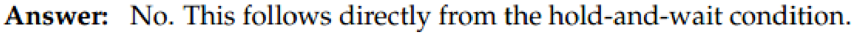
b. RR—treats all jobs equally (giving them equal bursts of CPU time) so short jobs will be able to leave the system faster since they will finish first.

c. Multilevel feedback queues—work similar to the RR algorithm—they discriminate favorably toward short jobs.

7.7 Show that, if the wait and signal operations are not executed atomically, then mutual exclusion may be violated.

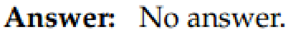
7.8 The Sleeping-Barber Problem. A barbershop consists of a waiting room with *n* chairs and the barber room containing the barber chair. If there are no customers to be served,the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop.If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.

8.2 Is it possible to have a deadlock involving only one single process? Explain your answer.



Answer: No. This follows directly from the hold-and-wait condition.

8.4 Consider the traffic deadlock depicted in Figure 8.11.
a. Show that the four necessary conditions for deadlock indeed hold in this example.
b. State a simple rule that will avoid deadlocks in this system.

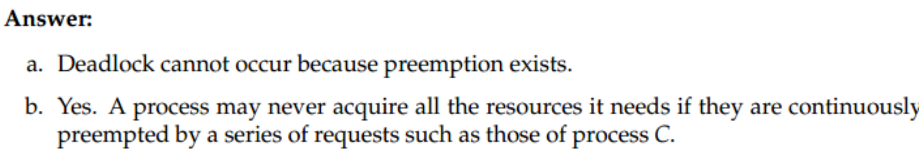


Answer: No answer.

8.13 Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:
a. What is the content of the matrix *Need*?
b. Is the system in a safe state?
c. If a request from process *P*1 arrives for (0,4,2,0), can the request be granted immediately?

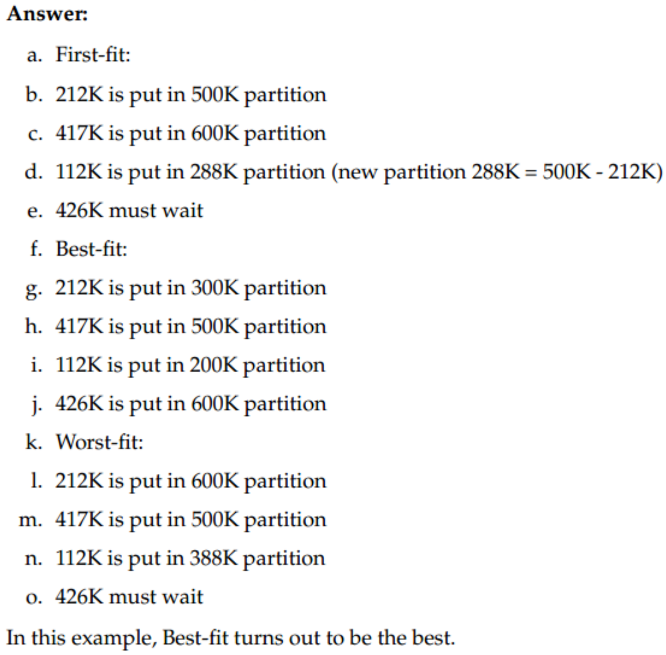


Answer:

a. Deadlock cannot occur because preemption exists.

b. Yes. A process may never acquire all the resources it needs if they are continuously preempted by a series of requests such as those of process *C*.

9.5 Given memory partitions of 100K, 500K, 200K, 300K, and 600K (in order), how would each
of the First-fit, Best-fit, and Worst-fit algorithms place processes of 212K, 417K, 112K, and 426K (in order)? Which algorithm makes the most efficient use of memory?



Answer:

a. First-fit:

b. 212K is put in 500K partition

c. 417K is put in 600K partition

d. 112K is put in 288K partition (new partition 288K = 500K - 212K)

e. 426K must wait

f. Best-fit:

g. 212K is put in 300K partition

h. 417K is put in 500K partition

i. 112K is put in 200K partition

j. 426K is put in 600K partition

k. Worst-fit:

l. 212K is put in 600K partition

m. 417K is put in 500K partition

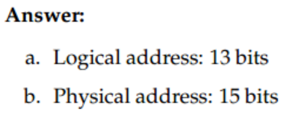
n. 112K is put in 388K partition

o. 426K must wait

In this example, Best-fit turns out to be the best.

9.8 Consider a logical address space of eight pages of 1024 words each, mapped onto a physical
memory of 32 frames.

a. How many bits are there in the logical address?
b. How many bits are there in the physical address?



Answer:

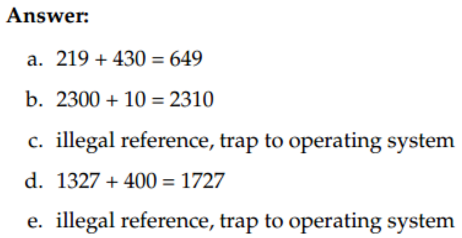
a. Logical address: 13 bits

b. Physical address: 15 bits

9.16 Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?
a. 0,430
b. 1,10
c. 2,500
d. 3,400
e. 4,112



Answer:

a. 219 + 430 = 649

b. 2300 + 10 = 2310

c. illegal reference, trap to operating system

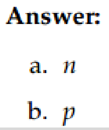
d. 1327 + 400 = 1727

e. illegal reference, trap to operating system

10.2 Assume that you have a page reference string for a process with *m* frames (initially all empty). The page reference string has length *p* with *n* distinct page numbers occur in it. For any page-replacement algorithms,

a. What is a lower bound on the number of page faults?

b. What is an upper bound on the number of page faults?



Answer:

a. *n*

b. *p*

10.11 Consider the following page reference string:

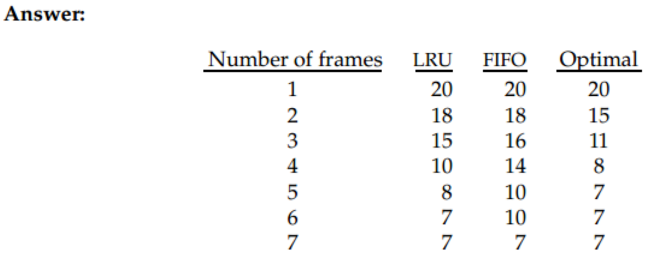
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

LRU replacement

FIFO replacement

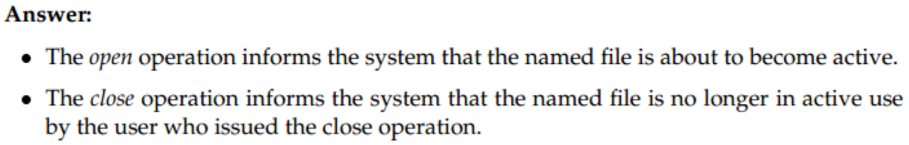
Optimal replacement



Answer:

Number of frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	8
5	8	10	7
6	7	10	7
7	7	7	7

11.7 Explain the purpose of the open and close operations.



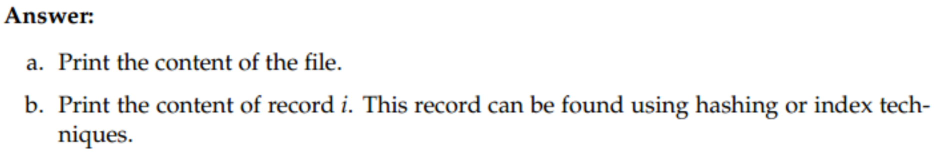
Answer:

- The *open* operation informs the system that the named file is about to become active.
- The *close* operation informs the system that the named file is no longer in active use by the user who issued the close operation.

11.9 Give an example of an application in which data in a file should be accessed in the following order:

a. Sequentially

b. Randomly



Answer:

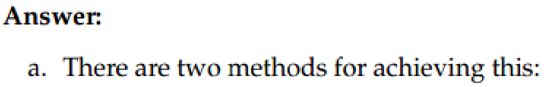
a. Print the content of the file.

b. Print the content of record *i*. This record can be found using hashing or index techniques.

11.12 Consider a system that supports 5000 users. Suppose that you want to allow 4990 of these users to be able to access one file.

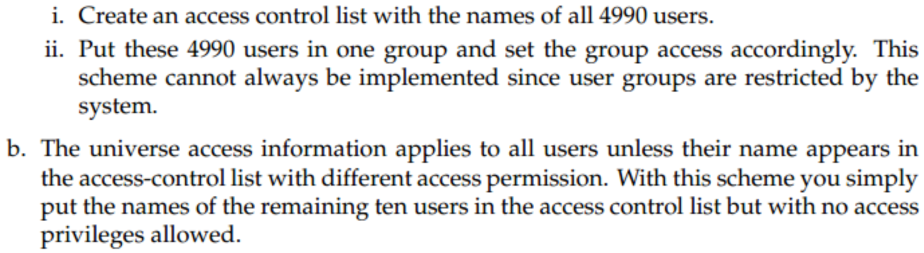
a. How would you specify this protection scheme in UNIX?

b. Could you suggest another protection scheme that can be used more effectively for this purpose than the scheme provided by UNIX?



Answer:

a. There are two methods for achieving this:



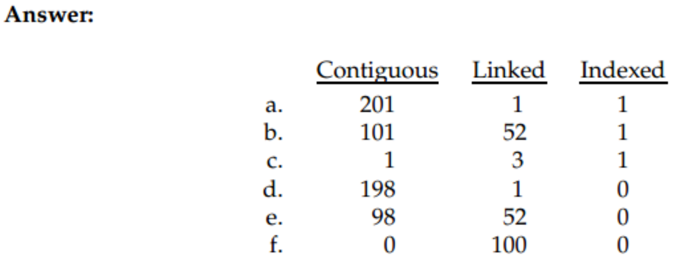
i. Create an access control list with the names of all 4990 users.

ii. Put these 4990 users in one group and set the group access accordingly. This scheme cannot always be implemented since user groups are restricted by the system.

b. The universe access information applies to all users unless their name appears in the access-control list with different access permission. With this scheme you simply put the names of the remaining ten users in the access control list but with no access privileges allowed.

12.1 Consider a file currently consisting of 100 blocks. Assume that the file control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block, the following conditions hold. In the contiguousallocation case, assume that there is no room to grow in the beginning, but there is room to grow in the end. Assume that the block information to be added is stored in memory.

a. The block is added at the beginning.
b. The block is added in the middle.
c. The block is added at the end.
d. The block is removed from the beginning.
e. The block is removed from the middle.
f. The block is removed from the end.



Answer:

	Contiguous	Linked	Indexed
a.	201	1	1
b.	101	52	1
c.	1	3	1
d.	198	1	0
e.	98	52	0
f.	0	100	0

13.2 Consider the following I/O scenarios on a single-user PC.

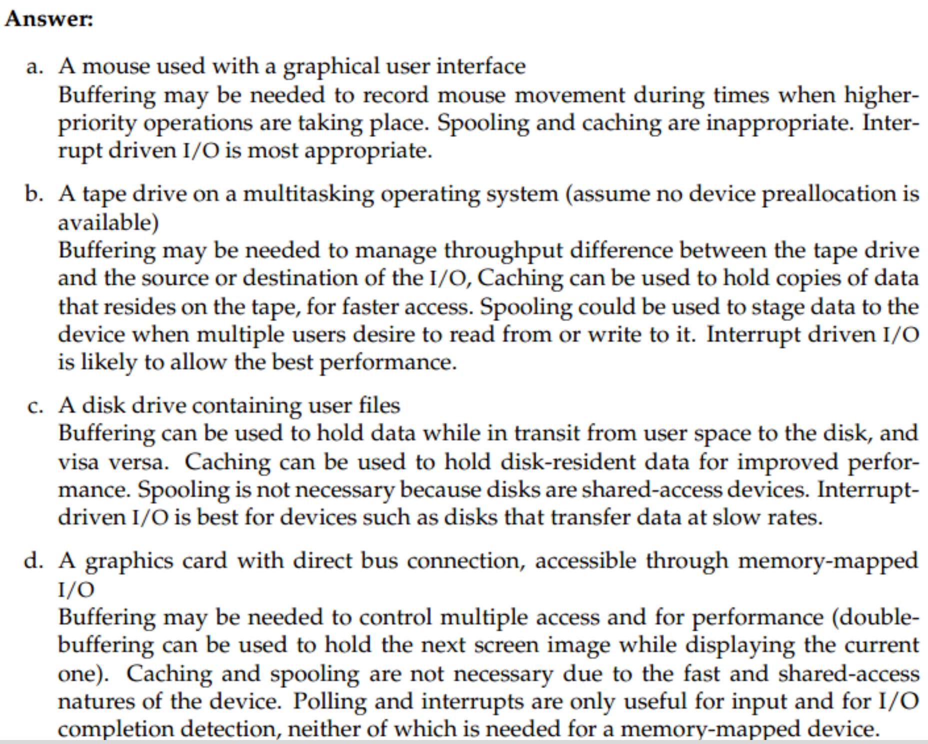
a. A mouse used with a graphical user interface

b. A tape drive on a multitasking operating system (assume no device preallocation is available)

c. A disk drive containing user files

d. A graphics card with direct bus connection, accessible through memory-mapped I/O

For each of these I/O scenarios, would you design the operating system to use buffering, spooling, caching, or a combination? Would you use polled I/O, or interrupt-driven I/O? Give reasons for your choices.



Answer:

a. A mouse used with a graphical user interface
Buffering may be needed to record mouse movement during times when higher-priority operations are taking place. Spooling and caching are inappropriate. Interrupt driven I/O is most appropriate.

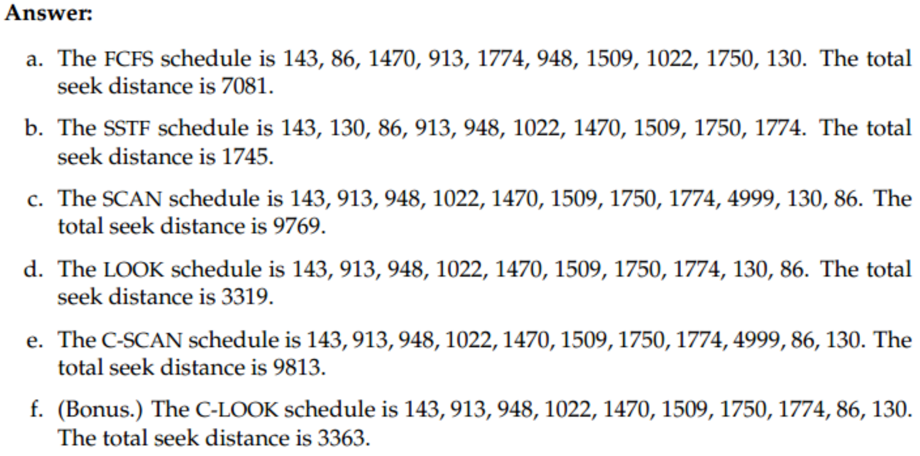
b. A tape drive on a multitasking operating system (assume no device preallocation is available)
Buffering may be needed to manage throughput difference between the tape drive and the source or destination of the I/O, Caching can be used to hold copies of data that resides on the tape, for faster access. Spooling could be used to stage data to the device when multiple users desire to read from or write to it. Interrupt driven I/O is likely to allow the best performance.

c. A disk drive containing user files
Buffering can be used to hold data while in transit from user space to the disk, and visa versa. Caching can be used to hold disk-resident data for improved performance. Spooling is not necessary because disks are shared-access devices. Interrupt-driven I/O is best for devices such as disks that transfer data at slow rates.

d. A graphics card with direct bus connection, accessible through memory-mapped I/O
Buffering may be needed to control multiple access and for performance (double-buffering can be used to hold the next screen image while displaying the current one). Caching and spooling are not necessary due to the fast and shared-access natures of the device. Polling and interrupts are only useful for input and for I/O completion detection, neither of which is needed for a memory-mapped device.

14.2 Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently
serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is
86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130
Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following diskscheduling
algorithms?

a. FCFS
b. SSTF
c. SCAN
d. LOOK
e. C-SCAN



Answer:

a. The FCFS schedule is 143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. The total seek distance is 7081.

b. The SSTF schedule is 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774. The total seek distance is 1745.

c. The SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 130, 86. The total seek distance is 9769.

d. The LOOK schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 130, 86. The total seek distance is 3319.

e. The C-SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 4999, 86, 130. The total seek distance is 9813.

f. (Bonus.) The C-LOOK schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774, 86, 130. The total seek distance is 3363.

1.1 1.6 2.3 2.5 3.7 6.3 6。4 7.8 8.13 9.5 9.8 10.11 12.1 14.2