Chapter 8 Main Memory (P. 310)

8.3

8.5

8.9

8.12

Thinking about:

8.1

8.10

8.11

8.13

8.1 Explain the difference between internal and external fragmentation.

Answer:

Internal Fragmentation is the area in a region or a page that is not used by the job occupying that region or page. This space is unavailable for use by the system until that job is finished and the page or region is released.

8.3 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?

8.3 Answer:

a. First-fit:

1) 212K is put in 500K partition
2) 417K is put in 600K partition
3) 112K is put in 288K partition (new partition 288K = 500K - 212K)
4) 426K must wait

b. Best-fit:

1) 212K is put in 300K partition
2) 417K is put in 500K partition
3) 112K is put in 200K partition
4) 426K is put in 600K partition

c. Worst-fit:

1) 212K is put in 600K partition
2) 417K is put in 500K partition
3) 112K is put in 388K partition
4) 1 426K must wait

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

8.5

	External fragmentation	Internal fragmentation	Ability to share code across processes
Contiguous	Yes	No	Difficult
Pure segmentation	Yes	No	Easy most
Pure paging	No	Yes	easy

8.9 Answer:

a. **400 nanoseconds; 200 nanoseconds to access the page table and 200 nanoseconds to access the word in memory.**

b. **Effective access time = 0.75 × (200 nanoseconds) + 0.25× (400 nanoseconds) = 250 nanoseconds.**

8.11 Explain why it is easier to share a reentrant module using segmentation than it is to do so when pure paging is used.

Answer:

Since segmentation is based on a logical division of memory rather than a physical one, segments of any size can be shared with only one entry in the segment tables of each user.

With paging there must be a common entry in the page tables for each page that is shared.

8.12

- 219+430=649
- 2300+10=2310
- Error
- 1327+400=1727
- error

Chapter 9 Virtual Memory (P. 366)

9.4 9.5 9.11 9.21(experiment 3)

1. Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.
2. 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 three, or four frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.
   1. LRU replacement
   2. FIFO replacement
   3. Optimal replacement

9.4 answer

- Virtual memory space: 2^{32}
- Physical memory: 2^{18} B
- Page size: 4096B=2^{12} B
- Logical address: 11123456
- Translation:
  - 11123456/4096: S=2715   R=2816
  - Page number p=2715   offset d=2816
  - Page table[2715]=frame number F
  - Physical address: F d

9.5 Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds.

Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

Answer:

0.2μsec = (1-p)*0.1μsec + (0.3p)*8 millisec +(0.7p)*20 millisec

0.1= -0.1p+2400p+14000p

0.1 ≈ 16,400p

P ≈ 0.000006

9.11 (P.340)

- **Page fault 1** 把指令load [M]所在页调入内存
- **Page fault 2** 把地址M所在页面调入内存，取出其内容a
- **Page fault 3** 将地址a对应的页面调入内存，从中取出内容

- **Thrashing**

1. **Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.**

   Answer:

   A page fault occurs when an access to a page that has not been brought into main memory takes place.

   The operating system verifies the memory access,

   aborting the program if it is invalid.

   If it is valid, a free frame is located and I/O is requested to read the needed page into the free frame. Upon completion of I/O, the process table and page table are updated and the instruction is restarted.

Answer for 3 frames

FIFO

16 page faults   page fault rate: 16/20=80%

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
-	1	1	1	4		4	4	6	6	6		3	3	3		2	2		2	6
-	-	2	2	2		1	1	1	2	2		2	7	7		7	1		1	1
-	-	-	3	3		3	5	5	5	1		1	1	6		6	6		3	3
	F	F	F	F		F	F	F	F	F		F	F	F		F	F		F	F

Answer for 3 frames

LRU

15 page faults   page fault rate: 15/20=75%

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
-	1	1	1	4		4	5	5	5	1		1	7	7		2	2			2
-	-	2	2	2		2	2	6	6	6		3	3	3		3	3			3
-	-	-	3	3		1	1	1	2	2		2	2	6		6	1			6
	F	F	F	F		F	F	F	F	F		F	F	F		F	F			F

Answer for 3 frames

OPT

11 page faults   page fault rate: 11/20=55%

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
-	1	1	1	1			1	1				3	3			3	3			3
-	-	2	2	2			2	2				2	7			2	2			2
-	-	-	3	4			5	6				6	6			6	1			6
	F	F	F	F			F	F				F	F			F	F			F

Answer for 4 frames

LRU

10 page faults, page fault rate: 10/20=50%

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
-	1	1	1	1			1	1				1	1	6			6			
-	-	2	2	2			2	2				2	2	2			2			
-	-	-	3	3			5	5				3	3	3			3			
-	-	-	-	4			4	6				6	7	7			1			
	F	F	F	F			F	F				F	F	F			F			

Answer for 4 frames

FIFO

14 page faults, page fault rate: 14/20=70%

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
-	1	1	1	1			5	5	5	5		3	3	3		3	1		1	
-	-	2	2	2			2	6	6	6		6	7	7		7	7		3	
-	-	-	3	3			3	3	2	2		2	2	6		6	6		6	
-	-	-	-	4			4	4	4	1		1	1	1		2	2		2	
	F	F	F	F			F	F	F	F		F	F	F		F	F		F	

Answer for 4 frames

OPT

8 page faults, page fault rate: 8/20=40%

	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
-	1	1	1	1			1	1					7				1			
-	-	2	2	2			2	2					2				2			
-	-	-	3	3			3	3					3				3			
-	-	-	-	4			5	6					6				6			
	F	F	F	F			F	F					F				F			

Chapter 10 File-System Interface (P. 408)

10.1

10.2

10.9

10.1

- Links may refer to this new file
- When the file is deleted, links to it are all deleted together.

10.2

- Maintain one table that contains references to files that are being accessed by all users at the current time.
- One entry with shared counter.

10.9

- One copy: 便于保持数据的一致性和完整性，操作互斥
- More copy: 维护数据的一致性比较困难

Chapter 11 File-System Implementation (P. 447)

11.6

Thinking about: 11.1 11.2 11.3

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 contiguous-allocation 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.

11.1

– 分配快，无外部碎片，有内部碎片
– 分配慢，有外部碎片，无内部碎片
– 介于a,b之间

11.2：支持随机访问

11.3 Consider a system where free space is kept in a free-space list.

a. Suppose that the pointer to the free-space list is lost. Can the system reconstruct the free-space list? Explain your answer.

c. Suggest a scheme to ensure that the pointer is never lost as a result of memory failure.

Answer:

a. In order to reconstruct the free list, it would be necessary to perform “garbage collection.” This would entail searching the entire directory structure to determine which pages were already allocated to jobs. Those remaining unallocated pages could be relinked as the free-space list.

c. The free-space list pointer could be stored on the disk, perhaps in several places.

11.6 Answer:

Let *Z* be the starting file address (block number).

a. Contiguous. Divide the logical address by 512 with *X* and *Y* the resulting quotient and remainder respectively.

i. Add *X* to *Z* to obtain the physical block number. *Y* is the displacement into that block.

ii. 1

b. Linked. Divide the logical physical address by 511 with *X* and *Y* the resulting quotient and remainder respectively.

i. Chase down the linked list (getting *X* + 1 blocks). *Y* + 1 is the displacement into the last physical block.

ii. 4

c. Indexed. Divide the logical address by 512, with *X* and *Y* the resulting quotient and remainder, respectively.

i. Get the index block into memory. Physical block address is contained in the index block at location *X*. *Y* is the displacement into the desired physical block.

ii. 2

3. 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

Chapter 12 Mass-Storage Structure (P. 489)

12.1

12.2

12.1 Answer:

- a. New requests for the track over which the head currently resides can theoretically arrive as quickly as these requests are being serviced.
- b. All requests older than some predetermined age could be "forced" to the top of the queue, and an associated bit for each could be set to indicate that no new request could be moved ahead of these requests. For SSTF, the rest of the queue would have to be reorganized with respect to the last of these "old" requests.
- c. To prevent unusually long response times.
- d. Paging and swapping should take priority over user requests. It may be desirable for other kernel-initiated I/O, such as the writing of file system metadata, to take precedence over user I/O. If the kernel supports real-time process priorities, the I/O requests of those processes should be favored.

12.2 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 9985.

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

Chapter 13 I/O Systems (P. 526)

13.6 Answer:

Generally, blocking I/O is appropriate when the process will only be waiting for one specific event. Examples include a disk, tape, or keyboard read by an application program. Non-blocking I/O is useful when I/O may come from more than one source and the order of the I/O arrival is not predetermined. Examples include network daemons listening to more than one network socket, window managers that accept mouse movement as well as keyboard input, and I/O-management programs, such as a copy command that copies data between I/O devices. In the last case, the program could optimize its performance by buffering the input and output and using non-blocking I/O to keep both devices fully occupied.

Non-blocking I/O is more complicated for programmers, because of the asynchronous rendezvous that is needed when an I/O occurs. Also, busy waiting is less efficient than interrupt-driven I/O so the overall system performance would decrease.