CHAPTER 8

8.5 Compare the main memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:

a. external fragmentation
b. internal fragmentation
c. ability to share code across processes

Hint/Ans:
Contiguous memory allocation: external fragmentation, no sharing
Pure segmentation: external fragmentation
Pure paging: internal fragmentation

8.8 Although Android does not support swapping on the boot disk, it is possible to set up a swap space using a separate SD nonvolatile memory card. Why would Android disallow swapping on its boot disk yet allow it on a secondary disk?

Hint/Ans:
Probable reasons are as follows: 1) Boot disk is not spacious enough to accommodate the swap space. 2) Instead of swapping, Android opts to kill the process in case of space shortage. 3) Wear-out problem that flash memory shows

8.12 Assuming a 1 KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

a. 3085 b. 42095 c. 215201 d. 650000 e. 2000001

Hint/Ans:
The address and page number starts at 0.
b. page = 41; offset = 111

8.16 Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following?

a. A conventional single-level page table?
b. An inverted page table?

Hint/Ans:
a. 2^{20} b. 2^{17}

8.19 Explain why sharing a reentrant module is easier when segmentation is used than when pure paging is used.

Hint/Ans:
segmentation: only one segment
paging: one or more pages to be shared

8.24 Consider the Intel address-translation scheme shown in Figure 8.22.

a. Describe all the steps taken by the Intel Pentium in translating a logical address into a physical address.
b. What are the advantages to the operating system of hardware that provides such complicated memory translation?
c. Are there any disadvantages to this address-translation system? If so, what are they? If not, why is

this scheme not used by every manufacturer?

Hint/Ans:

a. See Section 8.7.1.2 (IA-32 paging)
b. help operating systems to implement their memory scheme in hardware. hardware → more efficient, and simpler kernel
c. multiple table lookups

CHAPTER 9

9.2 A simplified view of thread states is **Ready**, **Running**, and **Blocked**, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O). This is illustrated in Figure 9.31. Assuming a thread is in the Running state, answer the following questions, and explain your answer:

a. Will the thread change state if it incurs a page fault? If so, to what new state?
b. Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what new state?
c. Will the thread change state if an address reference is resolved in the page table? If so, to what new state?

Hint/Ans:

- a. Yes. b. No. c. No.

9.4 What is the copy-on-write feature, and under what circumstances is it beneficial? What hardware support is required to implement this feature?

Hint/Ans:

Read only data. The hardware support: write-protected bit in the page table entry. When trying to write a write-protected page, an exception occurs to tell OS.

9.5 A certain computer provides its users with a virtual-memory space of 2^{32} bytes. The computer has 2^{18} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

Hint/Ans:

The virtual address: 0001 0001 0001 0010 0011 0100 0101 0110
Translation process: hardware-implemented

9.6 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?

Hint/Ans:

See Section 9.2.2 (Performance of demand paging)
$P \simeq 0.000006$

9.8 Consider the following page reference string:

7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0 , 1.

Assuming demand paging with three frames, how many page faults would occur for the following

replacement algorithms?

- LRU replacement
- FIFO replacement
- Optimal replacement

Hint/Ans:

- 18
- 17
- 13

9.17 A page-replacement algorithm should minimize the number of page faults. We can achieve this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages associated with that frame. Then, to replace a page, we can search for the page frame with the smallest counter.

a. Define a page-replacement algorithm using this basic idea. Specifically address these problems:
   i. What the initial value of the counters
   is ii. When counters are increased
   iii. When counters are decreased
   iv. How the page to be replaced is selected

Hint/Ans:

a. i. 0.
   ii. whenever a new page is associated with that frame.
   iii. whenever one of the pages associated with that frame is no longer required.
   iv. find a frame with the smallest counter.

9.19 What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

Hint/Ans:

Underallocated memory(paged). Evaluate the level of CPU utilization vs. the level of multiprogramming. Reduce the level of multiprogramming.

9.21 Consider the parameter Δ used to define the working-set window in the working-set model. What is the effect of setting Δ to a small value on the page fault frequency and the number of active (non-suspended) processes currently executing in the system? What is the effect when Δ is set to a very high value?

Hint/Ans:

See Section 9.6.2 (p. 419)

CHAPTER 10

10.2 The open-file table is used to maintain information about files that are currently open. Should the operating system maintain a separate table for each user or just maintain one table that contains references to files that are being accessed by all users at the current time? If the same file is being accessed by two different programs or users, should there be separate entries in the open file table?

Hint/Ans:

Centralized control for modifications. Separate access and state maintenance for reading

10.4 Provide examples of applications that typically access files according to the following methods:

• Sequential • Random

Hint/Ans:

• multimedia players, wordprocessors. • databases, video and audio editors.

10.6 If the operating system were to know that a certain application is going to access the file data in a sequential manner, how could it exploit this information to improve performance?

Hint/Ans:

Prefetching

10.11 What are the implications of supporting UNIX consistency semantics for shared access for those files that are stored on remote file systems?

Hint/Ans:

UNIX consistency semantics requires updates to a file to be immediately available to other processes. Inefficiencies: all updates by a client have to be immediately reported to the file server, and updates have to be communicated by the file server to clients caching the data immediately.

CHAPTER 11

11.4 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.

b. Consider a file system similar to the one used by UNIX with indexed allocation. How many disk I/O operations might be required to read the contents of a small local file at /a/b/c? Assume that none of the disk blocks is currently being cached.

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

Hint/Ans:

a. garbage collection: mark & sweep

b. 4

c. use disk space.

11.5 Some file systems allow disk storage to be allocated at different levels of granularity. For instance, a file system could allocate 4 KB of disk space as a single 4-KB block or as eight 512-byte blocks. How could we take advantage of this flexibility to improve performance? What modifications would have to be made to the free-space management scheme in order to support this feature?

Hint/Ans:

Less internal fragmentation. If a file is 5 KB, then it could be allocated a 4 KB block and two contiguous 512-byte blocks. In addition to maintaining a bitmap of free blocks, one would also have to maintain extra state regarding which of the subblocks are currently being used inside a block. The allocator would then have to examine this extra state to allocate subblocks and coalesce the subblocks to obtain the larger block when all of the subblocks become free.

11.7 Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous,

linked, and indexed), answer these questions:

a. How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)
b. If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

Hint/Ans:

Z = starting file address (block number).

- **Contiguous**. logical address / 512 → *X*(quotient) and *Y* (remainder)
  a. *X* + *Z* → physical block number. *Y* → displacement b. 1
- **Linked**. logical address / 511 → *X*(quotient) and *Y* (remainder)
  a. Chase down the linked list (getting *X* + 1 blocks). *Y* + 1 → the displacement into the last physical block. b. 4 (start from logical block 0 because no backward links are allowed)
- **Indexed**. logical address / 512 → *X*(quotient) and *Y* (remainder)
  a. Get the index block into memory. Physical block address is contained in the index block at location *X*. b. 2

CHAPTER 12

12.2 Explain why SSDs often use a FCFS disk scheduling algorithm.

Hint/Ans:

No moving parts

12.3 Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is:

2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681

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 disk-scheduling algorithms?

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

Hint/Ans:

a. 13,011. b. 7586. c. 7492. d. 7424. e. 9917. f. 9137.

12.10 Compare the throughput achieved by a RAID Level 5 organization with that achieved by a RAID Level 1 organization for the following:

a. Read operations on single blocks
b. Read operations on multiple contiguous blocks

Hint/Ans:

a. RAID Level 5: the number of data disks plus a parity disk, RAID Level 1: 2 for two-disk mirroring
b. RAID Level 5 organization achieves greater bandwidth for accesses to multiple contiguous blocks since the adjacent blocks could be simultaneously accessed. Such bandwidth improvements are not possible in RAID Level 1.

12.15 Discuss the reasons why the operating system might require accurate information on how blocks are stored on a disk. How could the operating system improve file system performance with this knowledge?

Hint/Ans:

While allocating blocks for a file, the operating system could allocate blocks that are geometrically close by on the disk if it had more information regarding the physical location of the blocks on the disk. In particular, it could allocate a block of data and then allocate the second block of data in the same cylinder but on a

different surface at a rotationally optimal place so that the access to the next block could be made with minimal cost.

CHAPTER 13

13.3 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.

Hint/Ans:

a. appropriate: buffering, interrupt-driven I/O. inappropriate: spooling and caching
b. appropriate: buffering, caching, spooling, interrupt-driven I/O
c. appropriate: buffering, caching, interrupt-driven I/O inappropriate: spooling
d. appropriate: buffering (double-buffering), polling and interrupts (both useful only for input and for I/O completion detection) inappropriate: caching, spooling

13.4 In most multiprogrammed systems, user programs access memory through virtual addresses, while the operating system uses raw physical addresses to access memory. What are the implications of this design on the initiation of I/O operations by the user program and their execution by the operating system?

Hint/Ans:

copy between the user buffer and the kernel buffer
translation of virtual address by OS
overhead incurred by above operations

13.5 What are the various kinds of performance overheads associated with servicing an interrupt?

Hint/Ans:

saving and restoring process state, and the cost of flushing the instruction pipeline and restoring the instructions into the pipeline when the process is restarted.

13.8 Some DMA controllers support direct virtual memory access, where the targets of I/O operations are specified as virtual addresses and a translation from virtual to physical address is performed during the DMA. How does this design complicate the design of the DMA controller? What are the advantages of providing such a functionality?

Hint/Ans:

increase in complexity: addition of an address-translation unit to the DMA controller, resulting in both hardware and software costs, and possibly in coherence problems between the data structures maintained by the CPU for address translation and corresponding structures used by the DMA controller.
advantages: no need for address translation initiation by CPU

CHAPTER 14

14.5 Discuss the strengths and weaknesses of implementing an access matrix using access lists that are associated with objects.
Hint/Ans:
Strength: the control that can be exerted by the object, that is, the object can revoke or expand the access privileges.
Weakness: the overhead of checking whether the requesting domain appears on the access list -- expensive

14.6 Discuss the strengths and weaknesses of implementing an access matrix using capabilities that are associated with domains.
Hint/Ans:
Strength: Provides substantial flexibility and faster access to objects. When a domain presents a capability, the system just needs to check the authenticity of the capability and that could be performed efficiently. Capabilities could also be passed around from one domain to another domain with great ease, allowing a system with a great amount of flexibility.
Weakness: the cost of a lack of control: revoking capabilities and restricting the flow of capabilities is a difficult task.

CHAPTER 15

15.1 Buffer-overflow attacks can be avoided by adopting a better programming methodology or by using special hardware support. Discuss these solutions.
Hint/Ans:
hardware support: prevent the execution of code located in the stack segment of a process's address space
better programming methodology: bounds-checking to guard against buffer overflows.

15.4 The list of all passwords is kept within the operating system. Thus, if a user manages to read this list, password protection is no longer provided. Suggest a scheme that will avoid this problem. (Hint: Use different internal and external representations.)
Hint/Ans:
Encrypt the passwords internally

15.11 What commonly used computer programs are prone to man-in-the-middle attacks? Discuss solutions for preventing this form of attack.
Hint/Ans:
Any protocol that requires a sender and a receiver to agree on a session key before they start communicating is prone to the man-in-the-middle attack. When the attacker receives the data from the client, it can decrypt the data, reencrypt it with the original key from the server, and transmit the encrypted data to the server without alerting either the client or the server about the attacker's presence. Such attacks could be avoided by using digital signatures to authenticate messages from the server; so, the attacker can't understand the message and the man-in-the-middle attack can be avoided.

15.15 Consider a system that generates 10 million audit records per day. Also assume that there are on average 10 attacks per day on this system and that each such attack is reflected in 20 records. If the intrusion-detection system has a true-alarm rate of 0.6 and a false-alarm rate of 0.0005, what percentage of alarms generated by the system correspond to real intrusions?
Hint/Ans:

The probability of occurrence of intrusive records is $10 * 20/10^7 = 0.00002$. Using Bayes' theorem, the probability that an alarm corresponds to a real intrusion is simply

$0.00002*0.6/(0.00002*0.6+0.99998*0.0005) = 0.0234$

$$P(W|L) = \frac{P(L|W)P(W)}{P(L)} = \frac{P(L|W)P(W)}{P(L|W)P(W) + P(L|M)P(M)}$$