2018 S1 A2

QUESTION 1

1.1. Consider the following snapshot of a system:

	Allocation	**Max**	**Available**
	ABCD	ABCD	ABCD
P0	0012	0012	1520
P1	1000	1750	
P2	1354	2356	
P3	0632	0652	
P4	0014	0656	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix Need?

Need = Max – Allocation

	Need
	ABCD
P0	0000
P1	0750
P2	1002
P3	0020
P4	0642

b. Is the system in a safe state?

By using **Banker's Safety algorithm**

Step 1:

Work = (1, 5, 2, 0) : resource available

Finish = (f, f, f, f, f) : none of the process is complete.

Step 2:

P0 or P1 have Need less than Work and can be completed. We select P0.

Step 3:

Assume P0 is completed then

Work = (1,5,2,0) + (0,0,1,2) = (1,5,3,2)

Finish = (t, f, f, f, f)

Step 2:

P2 and P3 have Need less than Work and can be completed, we select P2.

Step 3:

Assume P2 is complete, then

Work = (1,5,3,2) + (1,3,5,4) = (2,8,8,6)

Finish = (t, f, t, f, f)

Step 2:

P1, P3, and P4 have Need less than Work and can be completed, we select P1.

Step 3:

Assume P1 is complete, then

Work = (2, 8, 8, 6) + (0, 7, 5, 0) = (2,15,13,6)

Finish = (t, t, t, f, f)

Step 2:

P3, and P4 have Need less than Work and can be completed, we select P3.

Step 3:

Assume P3 is complete, then

Work = (2,15,13,6) + (0,6,3,2) = (2, 21, 26, 8)

Finish = (t, t, t, t, f)

Step 2:

P4 is the only process left with Need less than Work and can be completed.

Step 3:

Assume P4 is complete, then

Work = (2, 21, 26, 8) + (0, 0, 1, 4) = (2, 21, 27, 12)

Finish = (t, t, t, t, t)

Step 4:

Finish = (t, t, t, t, t) therefore the system is in a safe state

and one safe sequence is **<P0, P2, P1, P3, P4>**

c. If a request from process **P1** arrives for **(0,4,2,0)**, can the request be granted immediately?

i) Comparing (0, 4, 2, 0) with the need of P1: (0,4,2,0) ≤ (0, 7, 5, 0).

ii) Comparing (0,4,2,0) with resources available: (0,4,2,0) ≤ (1, 5, 2, 0).

iii) Assume requested resource is allocated, updated the system and using Banker's Safety algorithm:

	Allocation	**Max**	**Need**	**Available**
	ABCD	ABCD	ABCD	ABCD
P0	0012	0012	0000	1100
P1	1420	1750	0330	
P2	1354	2356	1002	
P3	0632	0652	0020	
P4	0014	0656	0642	

Step 1:

Work = (1, 1, 0, 0) : resource available

Finish = (f, f, f, f, f) : none of the process is complete.

Step 2:

Only P0 has a need less than Work and can be completed. We select P0.

Step 3:

Assume P0 is completed then

Work = (1,1,0,0) + (0,0,1,2) = (1,1,1,2)

Finish = (t, f, f, f, f)

Step 2:

Only P2 has a need less than Work and can be completed, we select P2.

Step 3:

Assume P2 is complete, then

Work = (1,1,1,2) + (1,3,5,4) = (2,4,6,6)

Finish = (t, f, t, f, f)

Step 2:

P1 and P3, have Need less than Work and can be completed, we select P1.

Step 3:

Assume P1 is complete, then

Work = (2,4,6,6) + (0, 3, 3, 0) = (2,7,9,6)

Finish = (t, t, t, f, f)

Step 2:

P3 and P4 have Need less than Work and can be completed, we select P3.

Step 3:

Assume P3 is complete, then

Work = (2,7,9,6) + (0,6,3,2) = (2, 13, 12, 8)

Finish = (t, t, t, t, f)

Step 2:

P4 is the only process left with Need less than Work and can be completed.

Step 3:

Assume P4 is complete, then

Work = = (2, 13, 12, 8) + (0, 0, 1, 4) = (2, 13, 13, 12)

Finish = (t, t, t, t, t)

Step 4:

Finish = (t, t, t, t, t) therefore the system is in a safe state

and one safe sequence is **<P0, P2, P1, P3, P4>**

1.2. (7.10) **Is it possible to have a deadlock involving only one single-threaded process? Explain your answer.**

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

QUESTION 2

2.1. (8.15) **Explain why mobile operating systems such as iOS and Android do not support swapping.**

Answer: There are three reasons: First is that these mobile devices typically use flash memory with limited capacity and swapping is avoided because of this space constraint. Second, flash memory can support a limited number of write operations before it becomes less reliable. Lastly, there is typically poor throughput between main memory and flash memory.

2.2. (8.19) Program binaries in many systems are typically structured as follows. Code is stored starting with a small, fixed virtual address, such as 0. The code segment is followed by the data segment that is used for storing the program variables. When the program starts executing, the stack is allocated at the other end of the virtual address space and is allowed to grow toward lower virtual addresses. **What is the significance of this structure for the following schemes?**

a. Contiguous memory allocation

b. Pure segmentation

c. Pure paging

Answer:

1) **Contiguous-memory allocation** requires the operating system to allocate the entire extent of the virtual address space to the program when it starts executing. This could be much larger than the actual memory requirements of the process.

2) **Pure segmentation** gives the operating system flexibility to assign a small extent to each segment at program startup time and extend the segment if required.

3) **Pure paging** does not require the operating system to allocate the maximum extent of the virtual address space to a process at startup time, but it still requires the operating system to allocate a large page table spanning all of the program's virtual address space. When a program needs to extend the stack or the heap, it needs to allocate a new page but the corresponding page table entry is preallocated.

QUESTION 3

3.1. (9.8) 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

3.2 (9.19) 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:

$$0.2\,\mu\,sec\ = (1-P)\times 0.1\mu\,sec\ +\ (0.3P)\times 8\ \text{millisec}\ +\ (0.7P)\times 20\ \text{millisec}$$

$$0.1\ = -0.1P +\ 2400\,P +\ 14000\,P$$

$$0.1\ \simeq 16{,}400P$$

$$P \simeq 0.000006$$

QUESTION 4

4.1 (10.1) **Is disk scheduling, other than FCFS scheduling, useful in a single-user environment? Explain your answer.**

Answer:

In a single-user environment, the I/O queue usually is empty. Requests generally arrive from a single process for one block or for a sequence of consecutive blocks. In these cases, FCFS is an economical method of disk scheduling. But LOOK is nearly as easy to program and will give much better performance when multiple processes are performing concurrent I/O, such as when a Web browser retrieves data in the background while the operating system is paging and another application is active in the foreground.

4.2 (10.2) Explain why SSTF scheduling tends to favor middle cylinders over the innermost and outermost cylinders.

Answer:

The center of the disk is the location having the smallest average distance to all other tracks. Thus the disk head tends to move away from the edges of the disk. Here is another way to think of it. The current location of the head divides the cylinders into two groups. If the head is not in the center of the disk and a new request arrives, the new request is more likely to be in the group that includes the center of the disk; thus, the head is more likely to move in that direction.

QUESTION 5

5.1 (11.9) **Consider a file system where a file can be deleted and its disk space while links to that file still exist. What problems may occur if a new file is created in the same storage area or with the same absolute path name? How can these problems be avoided?**

Answer:

Let F1 be the old file and F2 be the new file. A user wishing to access F1 through an existing link will actually access F2. Note that the access protection for file F1 is used rather than the one associated with F2.

This problem can be avoided by insuring that all links to a deleted file are deleted also. This can be accomplished in several ways:

a. maintain a list of all links to a file, removing each of them when the file is deleted

b. retain the links, removing them when an attempt is made to access a deleted file

c. maintain a file reference list (or counter), deleting the file only after all links or references to that file have been deleted

5.2 (11.11) **What are the advantages and disadvantages of a system providing mandatory locks instead of providing advisory locks whose usage is left to the users' discretion?**

Answer:

In many cases, separate programs might be willing to tolerate concurrent access to a file without requiring the need to obtain locks and thereby guaranteeing mutual exclusion to the files. Mutual exclusion could be guaranteed by other program structures such as memory locks or other forms of synchronization. In such situations, the mandatory locks would limit the flexibility in how files could be accessed and might also increase the overheads associated with accessing files.

QUESTION 6

6.1 (14.12) The access-control matrix could be used to determine whether a process can switch from, say, domain A to domain B and enjoy the access privileges of domain B. Is this approach equivalent to including the access privileges of domain B in those of domain A?

Answer:

Yes, this approach is equivalent to including the access privileges of domain B in those of domain A as long as the switch privileges associated with domain B are also copied over to domain A.

6.2. (14.12) Consider a computer system in which computer games can be played by students only between 10P. M. and 6A.M., by faculty members between 5P. M. and 8A.M., and by the computer center staff at all times. Suggest a scheme for implementing this policy efficiently.

Answer:

Set up a dynamic protection structure that changes the set of resources available with respect to the time allotted to the three categories of users. As time changes, so does the domain of users eligible to play the computer games. When the time comes that a user's eligibility is over, a revocation process must occur. Revocation could be immediate, selective (since the computer staff may access it at any hour), total, and temporary (since rights to access will be given back later in the day).