Operating Systems Homework 4

Chapter 7:

7.4 Describe how deadlock is possible with the dining-philosophers problem.

-> 철학자는 한 번에 2개의 젓가락을 들어야 식사를 할 수 있다. 하지만 각 철학자가 자신의 왼쪽(혹은 오른쪽)에 놓인 젓가락을 집었을 때, 이미 다른 철학자가 든 젓가락을 집을 수 없으므로, 각자 1개의 젓가락을 든 채로 deadlock 상태에 빠지게 된다.

Chapter 8:

8.3 Consider the following snapshot of a system:

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
T_0	0 0 1 2	0 0 1 2	1 5 2 0
T_1	1 0 0 0	1 7 5 0	
T_2	1 3 5 4	2 3 5 6	
T_3	0 6 3 2	0 6 5 2	
T_4	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?

-> 행렬 Need는 Max – Allocation 이므로

T0 (0, 0, 0, 0)

T1 (0, 7, 5, 0)

T2 (1, 0, 0, 2)

T3 (0, 0, 2, 0)

T4 (0, 6, 4, 2) 이다.

b. Is the system in a safe state?

-> Safe state이다.

Available의 상태가
T0 -> 1532 -> T2 -> 2886 -> 이 이후로는 모든 쓰레드가 complete 가능하다.

c. If a request from thread T1 arrives for (0, 4, 2, 0), can the request be granted immediately?

-> 1520 <= 0420 이기 때문에 Request1 <= Available을 만족한다.
T1의 alloc이 1420이 되고, available은 1520 – 0420 = 1100이 된다.
T0 -> 1112 -> T2 -> 2466 -> T3 -> 이 이후로는 모든 쓰레드가 complete 가능하다.

8.7 Can a system detect that some of its threads are starving? If you answer "yes," explain how it can. If you answer "no," explain how the system can deal with the starvation problem.

-> 한 쓰레드가 너무 오래 자원을 할당받지 못할 경우, 기아상태라고 판단할 수 있다. 이럴 경우에는 시간이 지남에 따라 쓰레드에 우선순위를 가중하여 공정성을 제공하면 된다.

8.11 Is it possible to have a deadlock involving only one single-threaded process? Explain your answer

-> 다중 쓰레드 process에서도 4가지 조건을 만족한다면 데드락이 발생할 수 있다.

Chapter 9:

9.4 Consider a logical address space of 64 pages of 1,024 words each, mapped

onto a physical memory of 32 frames.

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

-> 64 = 2 ^ 6, 1024 = 2 ^ 10, 32 = 2 ^ 5

64 * 1024 = 2 ^ 16, 답은 16 bits

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

-> 4 * 32 = 2 ^ 15, 답은 15 bits

9.8 The BTV operating system has a 21-bit virtual address, yet on certain embedded devices, it has only a 16-bit physical address. It also has a 2-KB page size. How many entries are there in each of the following?

a. A conventional, single-level page table

-> 2KB = 2 ^ 11

2 ^ 21 / 2 ^ 11 = 2 ^ 10 항목.

b. An inverted page table

-> 2 ^ 16 / 2 ^ 11 = 2 ^ 5 항목.

9.10 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

4 KB = 2 ^ 12

512 MB = (2 ^ 9) * (2 ^ 10)

2 ^ 32 / 2 ^ 12 = 2 ^ 20 항목.

b. An inverted page table

2 ^ 9 * 2 ^ 10 / 2 ^ 12 = 2 ^ 7 항목.

Chapter 10:

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

-> 프로그램이 자신의 주소 공간에는 존재하지만 RAM에는 현재 없는 데이터나 코드에 접근을 시도하였을 경우 page fault가 발생한다. 페이지 폴트가 발생하면 trap을 발생시키고, page fault handler가 작동된다. 메모리에 가용 공간이 있으면 페이지를 가져와서 넣고, 없으면 victim을 정해 페이지를 교체한다.

10.6 Discuss the hardware functions required to support demand paging.

-> 페이지가 메모리에 상주하지 않을 때 현재 메모리에 없다라는 것을 표현해 주는 플래그가 있어야 한다. 또, page table에 virtual address의 주소값을 찾는 방법이 포함되어 있어야 한다.

10.7 Consider the two-dimensional array A:

int A[][] = new int[100][100];

where A[0][0] is at location 200 in a paged memory system with pages of size 200. A small process that manipulates the matrix resides in page 0 (locations 0 to 199). Thus, every instruction fetch will be from page 0. For three page frames, how many page faults are generated by the following array-initialization loops? Use LRU replacement, and assume that page frame 1 contains the process and the other two are initially empty.

A

-> a는 j를 고정하여 주소를 찾기 때문에, 2 x 100을 한 번 페이지 폴트 하였을 때, 2번의

접근밖에 하지 못하고 다음 페이지 폴트를 진행한다. 따라서

(page fault) x (2) = 100 x 100

총 페이지 폴트는 5000번 발생한다.

B

-> b는 i를 기준으로 주소를 찾기 때문에, 2 x 100을 한 번 페이지 폴트 하였을 때, 2 x 100 = 200의 페이지 크기를 모두 사용하고 난 후에 다음 페이지 폴트가 발생한다. 따라서

(page fault) x (2 x 100) = 100 x 100

총 페이지 폴트는 50번 발생한다.