長庚大學104學年度第一學期作業系統期末測驗（滿分100）

系級: 姓名: 學號:

1. (16%) Assume that you are an OS genius and helping Peterson to revise his algorithms for protecting the critical sections of processes P_i and P_j. (1) Please illustrate the problems of Algorithm 1 and Algorithm 2. (2) Please provide Algorithm 3 which can properly mange the critical sections of P_i and P_j, and make sure that Algorithm 3 can meet the requirements of Mutual Exclusion, Progress, and Bounded Waiting.

Algorithm 1

```
Process Pi :
 do {
   while (turn != i) ;
    critical section
   turn=j;
    remainder section
 } while (1);
```

```
Process Pj :
 do {
   while (turn != j) ;
    critical section
   turn=i;
    remainder section
 } while (1);
```

Algorithm 2

Initially, flag[i]=flag[j]=false

```
Process Pi :
 do {
   flag[i]=true;
   while (flag[j]) ;
    critical section
   flag[i]=false;
    remainder section
 } while (1);
```

```
Process Pj :
 do {
   flag[j]=true;
   while (flag[i]) ;
    critical section
   flag[j]=false;
    remainder section
 } while (1);
```

Answer:

(1) (8%) Algorithm 1: When one process leaves, the other one will wait infinitely. Algorithm 2: When the two processes set flag[i] and flag[j] as true, no one can break the while loop of the enter section.

(2) (8%)

```
Process Pi :
 do {
   flag[i]=true;
   turn=j;
   while (flag[j] && turn==j) ;
    critical section
   flag[i]=false;
    remainder section
 } while (1);
```

2. (12%) There are three processes:
   - P_1: a * b → c
   - P_2: c + d → c
   - P_3: c - e → c

- P_1 has to run before P_2 and P_3 do
- P_2 can run before P_3, and P_3 also can run before P_2
- The access to valuable "c" must be protected
- The initial states are: S_1=0; S_2=0; S_3=1;
- The code of P_1 is: c = a * b; signal(S_1); signal(S_2);

Please provide P_2 and P_3 by using wait() and signal() and meet the above requirements.

Answer:

- (6%) P_2: wait(S_1); wati(S_3) ; c = c + d ; signal(S_3) ;
- (6%) P_3: wait(S_2); wati(S_3) ; c = c - e ; signal(S_3) ;

3. (14%) Banker's Algorithm is a deadlock avoidance algorithm. Assume there are 5 processes {P_0, P_1, P_2, P_3, P_4} and three types of shared resources {A, B, C} in the system, and the details are in the following table. (1) By Banker's Algorithm, is the system in a safe state? If your answer is yes, please provide a safe sequence. If your answer is no, please provide the reason. (2) Now, P_0 further has a request (2, 1 , 0) to use 2 more instances of A and 1 more instances of B. Should the request be granted? Again, provide the reason to support your answer.

	Allocation			Max			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	7	4	3	3	3	2
P1	2	0	0	3	2	3	1	2	3			
P2	3	0	2	9	0	2	6	0	0			
P3	2	1	1	2	2	2	0	1	1			
P4	0	0	2	4	3	3	4	3	1			

Answer:

(1) (7%) Yes. Available(3, 3, 2) → P3 Need(0, 1, 1) → Available(5, 4, 3) → P1 Need(1, 2, 3) → Available(7, 4, 3) → P0 Need(7, 4, 3) → Available(7, 5, 3) → P2 Need(6, 0, 0) → Available(10, 5, 5) → P4 Need(4, 3, 1)

(2) (7%) Yes

(I) (2, 1, 0) <= P_0 Need (7, 4, 3)

(II) (2, 1, 0) <= Available (3, 3, 2)

(III) Available(1, 2, 2) → P3 Need(0, 1, 1) → Available(3, 3, 3) → P1 Need(1, 2, 3) → Available(5, 3, 3) → P0 Need(5, 3, 3) → Available(7, 5, 3) → P2 Need(6, 0, 0) → Available(10, 5, 5) → P4 Need(4, 3, 1)

4. (10%) Now, we consider the deadlock detection for a system with the following resource allocation and waiting state. Is the system in a deadlock state? You need to provide the reason for your answer.

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	0	0	0	2	0
P1	2	0	0	4	0	2			
P2	3	0	3	0	3	0			
P3	2	1	1	1	0	0			
P4	0	0	2	0	0	4			

Answer:

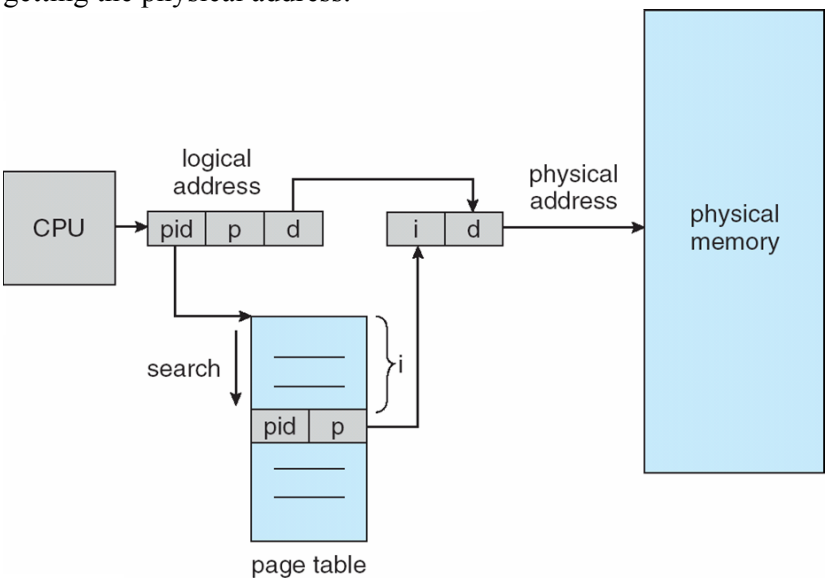
No. Available(0, 2, 0) → P0 Request(0, 0, 0) → Available(0, 3, 0) → P2 Request(0, 3, 0) → Available(3, 3, 3) → P3 Request(1, 0, 0) → Available(5, 4, 4) → P1 Request(4, 0, 2) → Available(7, 4, 4) → P4 Request(0, 0, 4)

5. (12%) Please define (1) logical address, (2) physical address, (3) static link, (4) dynamic link, (5) external fragmentation, and (6) internal fragmentation of memory and address management.

Answer:

(1) (2%) Logical address – generated by the CPU; also referred to as virtual address
(2) (2%) Physical address – address seen by the memory unit
(3) (2%) Static linking – system libraries and program code combined by the loader into the binary program image
(4) (2%) Dynamic linking – linking postponed until execution time
(5) (2%) External Fragmentation – total memory space exists to satisfy a request, but it is not contiguous
(6) (2%) Internal Fragmentation – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

6. (12%) For the inverted page table architecture, please define the (1) pid, (2) p, (3) d, and (4) i of the following figure. (5) Please briefly explain the mechanism of inverted page table architecture for getting the physical address.



Answer:

(1) (2%) Process ID to identify which process it is
(2) (2%) Page number in logical address
(3) (2%) The offset within the page
(4) (2%) The frame number in the physical memory
(5) (4%) It sequentially searches the pair (pid, p) in the page table. When the pair is found in the i-th entry of the page table, i is then used as the the frame number of the physical address, and d is the offset in the frame.

7. (10%) (1) What is reason of thrashing? Please explain it in detail. (2) How can we detect and prevent trashing?

Answer:

(1) (5%) When page fault rate increases, the CPU utilization might decrease. Since the CPU utilization decreases, the OS might launch more programs. Thus, the page fault rate further increases, and that is thrashing
(2) (5%) If the page fault rate is too high, the OS should not increase the total number of processes.

8. (14%) There is system with only 3 memory frames. Given a reference string of pages {7→0→0→2→0→3→0→4→1→3→7}. Please illustrate the page replacement of (1) the LRU algorithm and (2) the optimal algorithm. You should show the memory frames and the queue for the LRU algorithm. The explanation for each page replacement of the optimal algorithm should be provided.

Answer:

(1) (7%)

Memory Frame

7	7	7	7	7	3	3	3	1	1	1
	0	0	0	0	0	0	0	0	3	3
			2	2	2	2	4	4	4	7
LRU Queue										
7	0	0	2	0	3	0	4	1	3	7
	7	7	0	2	0	3	0	4	1	3
			7	7	2	2	3	0	4	1

(2) (7%)

Memory Frame

7	7	7	7	7	7	7	7	7	7	7
	0	0	0	0	0	0	4(#)	1(*)	1	1
			2	2	3(@)	3	3	3	3	3

@: 2 is no more used. The distance of the next using is infinite.

#: 0 is no more used. The distance of the next using is infinite.

*: 4 is no more used. The distance of the next using is infinite.