長庚大學109學年度第一學期作業系統期末測驗（滿分109）

系級:　　　　　　姓名:　　　　　　學號:

1. (8%) Please (a) define “Race Condition” and (b) provide an example for Race Condition. You can use the case, counter ++ and counter -- are in two different processes, as the example. (Hint: the assembly code of counter ++ could be: r_1 = counter; $r_1 = r_1 + 1$; counter = r_1;)

   Answer:

   (a) (4%) A situation where the outcome of the execution depends on the particular order of process scheduling.

   (b) (4%)

- One counter++ and one counter--

```
r1 = counter        r2 = counter
r1 = r1 + 1         r2 = r2 - 1
counter = r1        counter = r2
```

- Initially, let counter = 5
  1. P: r1 = counter
  2. P: r1 = r1 + 1
  3. C: r2 = counter
  4. C: r2 = r2 – 1　$\Rightarrow$ A Race Condition!
  5. P: counter = r1
  6. C: counter = r2 = 4
- The result can be 4, 5 or 6

2. (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 manage 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 :                      Process Pj :
do {                              do {
   while (turn != i) ;               while (turn != j) ;
   critical section                  critical section
   turn=j;                           turn=i;

   remainder section                 remainder section
} while (1);                      } while (1);
```

Algorithm 2

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

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

3. (8%) There are three processes:

P_1: a * b → a

P_2: a + d → d

P_3: e + d → d

P_1 should run before P_2 does. The access to valuable "d" must be protected in a critical session. The order of P_2 and P_3 is arbitrary. We have two semaphores S_1 and S_2, and they are initialized as S_1=0 and S_2=1. Now, the code of P_1 is provided as follows:

```
a =  a * b;
signal(S1);
```

Please provide the code of P_2 and P_3.

Answer:

P_2:

```
wait(S1);
wait(S2);
d = a + d;
signal(S2);
```

P_3:

```
wait(S2)
d = e + d;
signal(S2);
```

4. (8%) Please explain the problem of the following application.

```
Code:
void transaction(Account from, Account to, double amount)
{
   mutex lock1, lock2;
   lock1 = get lock(from);
   lock2 = get lock(to);
   acquire(lock1);
      acquire(lock2);
         withdraw(from, amount);
         deposit(to, amount);
      release(lock2);
   release(lock1);
}

Use:
transaction(acc1, acc2, 1000);
transaction(acc2, acc1, 4000);
```

Answer:

When the first and second transactions successfully lock the "from" accounts (acc1 and acc2, respectively), they can not further lock the "to" accounts (acc2 and acca, respectively). Thus the Hold-and-Wait situation eventually cause the deadlock.

5. (12%) Banker's Algorithm is a deadlock avoidance algorithm. Assume that 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 (1, 1 , 0) to use 1 more instance of A and 1 more instance 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	1	0	1	2	4	3	1	4	2			
P2	3	0	2	9	0	2	6	0	0			
P3	0	1	1	0	2	2	0	1	1			
P4	2	1	1	6	4	2	4	3	1			

Answer:

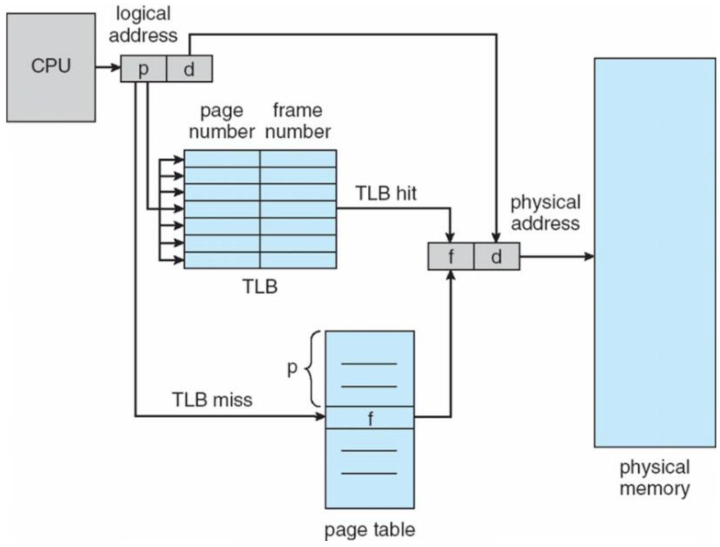
(1) Yes.

The safe sequence is: P_3 Need(0, 1, 1) $\leq$ Available(3, 3, 2) $\rightarrow$ P_1 Need(1, 4, 2) $\leq$ Available(3, 4, 3) $\rightarrow$ P_4 Need(4, 3, 1) $\leq$ Available(4, 4, 4) $\rightarrow$ P_2 Need(6, 0, 0) $\leq$ Available(6, 5, 5) $\rightarrow$ P_0 Need(7, 4, 3) $\leq$ Available(9, 5, 7)

(2) No.

1. P_0 Request(1, 1, 0) $\leq$ P_0 Need(7, 4, 3)
2. P_0 Request(1, 1, 0) $\leq$ Available(3, 3, 2)
3. After the system grants the request:
   Available(3, 3, 2) $\rightarrow$ Available(2, 2, 2).
   P0 has Need(6, 3, 3) and Allocation(1, 2, 0).
   We then run the Banker's Algorithm again: P_3 Need(0, 1, 1) $\leq$ Available(2, 2, 2) $\rightarrow$ Available is (2, 3, 3) now, But no one has Need no more than ($\leq$) (2, 3, 3). $\rightarrow$ fail to find a safe sequence.

6. (10%) For memory management with page tables, considering the following figure, please answer the questions: (a) What is p? (b) What is f? (c) What is d? (d) What is the function of TLB? (e) For two processes having two page tables respectively, please explain the mechanism of page sharing.



Answer:

(a) Page number

(b) Frame number

(c) Offset

(d) TLB is some on-chip SRAM which can keep the frame numbers of recently referred pages so as to solve the "two memory accesses" problem of using page table.

(e) To share a page in a physical memory frame, each of the page tables has an entry to map a page number to the same frame number so as to access the same physical memory frame.

7. (8%) Segmentation and paging are two different approaches for memory management. Please provide one advantage of paging when it is compared with segmentation.

Answer:

The paging approach partitions physical memory into frames with a fixed unit size and assign frames to processes. Thus, there is no external fragmentation for paging, but segmentation might have.

8. (16%) Please define (a) logical address, (b) physical address, (c) static link, (d) dynamic link, (e) external fragmentation, (f) internal fragmentation, (g) page fault, and (h) copy-on-write (COW) of memory and address management.
   (a) (2%) Logical address – generated by the CPU; also referred to as virtual address
   (b) (2%) Physical address – address seen by the memory unit
   (c) (2%) Static linking – system libraries and program code combined by the loader into the binary program image
   (d) (2%) Dynamic linking – linking postponed until execution time
   (e) (2%) External Fragmentation – total memory space exists to satisfy a request, but it is not contiguous
   (f) (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
   (g) (2%) A Page Fault occurs when a process references a non-memory-resident page
   (h) (2%) COW allows both parent and child processes to initially *share* the same pages in memory. If either process modifies a shared page, then the page is copied.

9. (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.

10. (9%) For the Second-Chance algorithm of page replacement, operating systems have to maintain a reference bit for each page. (a) When will a reference bit be changed from 1 (referred) to 0 (not referred)? (b) When will a reference bit be changed from 0 to 1? (c) How does the Second-Chance algorithm find out a victim page?

Answer:
   (a) The reference bit of a page is changed from 1 to 0 when the Second-Chance page replacement algorithm visit it.
   (b) The reference bit of a page is changed from 0 to 1 when any process or OS access the page.
   (c) The Second-Chance page replacement algorithm sequentially (round-robin) visits each reference bit until if find out a reference bit is 0. The page of the reference bit is then selected as the victim page.