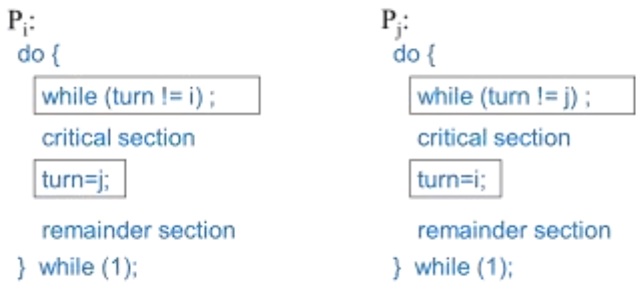
長庚大學106學年度第一學期作業系統期末測驗（滿分108）

系級:　　　　　　　姓名:　　　　　　　學號:

1. (10%) For the **first version** of **Peterson's Solution**, please explain the problem when we use the following algorithm for protecting the critical sections of P_i and P_j:



```
Pi:                                  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);
```

Answer:　When a task is finished, and the turn is on it, the other task has no chance to get the turn for its execution.

2. (10%) Let's consider the Readers and Writers Problem. We now have two mutex instances, mutex1 and mutex2. Please complete the code of writers and readers. (Hint: you should fill in all $\boxed{?}$ with mutex1 or mutex2 in the following sample code.)

Writer:

wait($\boxed{?}$);
… writing … ;
signal($\boxed{?}$);

Reader:

wait($\boxed{?}$);
readcount++;
if (readcount == 1) {wait($\boxed{?}$);}
signal($\boxed{?}$);
… reading … ;
wait($\boxed{?}$);
readcount--;
if (readcount== 0) {signal($\boxed{?}$);}
signal($\boxed{?}$);

Answer:

Writer:

```
wait(mutex1);
 … writing … ;
signal(mutex1);
```

Reader:

```
wait(mutex2);
readcount++;
if (readcount == 1) {wait(mutex1);}
signal(mutex2);
… reading … ;
wait(mutex2);
readcount--;
if (readcount== 0) {signal(mutex1);}
signal(mutex2);
```

如果所有mutex1都換成mutex2，且所有mutex2都換成mutex1，也正確。
不做部份給分，只有十分或零分。

3. (10%) For the Dining-Philosophers problem, assume that we have 5 philosophers, and we use the following code to manage the Dining-Philosophers problem. However, the following code might have a deadlock problem. Please explain the situation of have a deadlock with the following code.

We have 5 philosophers and semaphore chopstick[5].

For philosopher i (i = 0, 1, 2, 3, 4) we do:

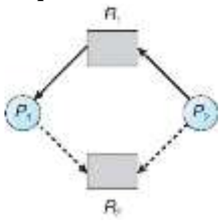
```
do {
        wait(chopstick[i]);
        wait(chopstick[(i + 1) % 5 ]);
        … eat …
        signal(chopstick[i]);
        signal(chopstick[(i+1) % 5]);
        …think …
} while (1);
```

Answer: When all tasks (philosophers) just get chopstick[i] and wait for chopstick[(i+1) % 5], there is a deadlock.

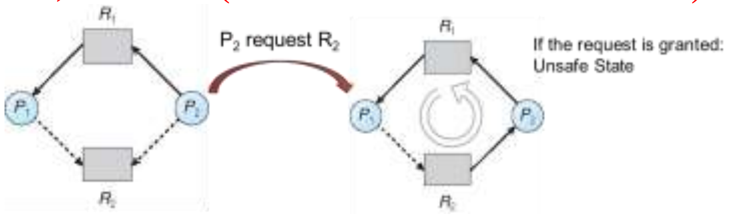
4. (10%) To manage deadlocks, OS can do “deadlock prevention” or “deadlock avoidance” to guarantee that there is no deadlock. If we allow deadlocks, OS can do “deadlock detection” and recover the system from deadlock states. Please define (a) deadlock prevention and (b) deadlock avoidance.
Answer:
(a) Prevent the necessary conditions (i.e., mutual exclusion, hold and wait, no preemption, and circular wait) of a deadlock
(b) Make sure that the system always stays at a “safe” state (by Banker’s Algorithm).

5. (8%) For the resource-allocation graph scheme, we have:
    - Claim edge $P_i \rightarrow R_j$ indicated that process P_i may request resource R_j; represented by a dashed line
    - Claim edge converts to request edge when a process requests a resource
    - Request edge converted to an assignment edge when the resource is allocated to the process
    - When a resource is released by a process, assignment edge reconverts to a claim edge

If we adopt deadlock avoidance to manage the deadlock problem, could we grant the request that P_2 request R_2 with the current situation in the following graph?



Answer:
No, the request should be rejected. If it is granted, there is a circle in the resource-allocation graph, and thus, it is unsafe. (note: It is not a deadlock. Just unsafe.)

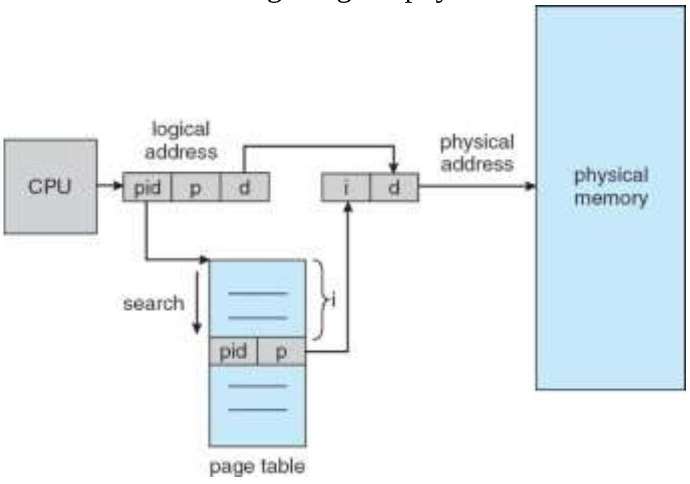


6. (12%) Please define (a) external fragmentation and (b) internal fragmentation of memory management. If we use “**Segmentation**” to manage memory, (c) is it possible to have external fragmentation? You have to provide reasons for the answer of sub-question c.

   Answer:

   (a) External Fragmentation – total memory space exists to satisfy a request, but it is not contiguous
   (b) Internal Fragmentation – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
   (c) Yes. The sizes of segmentations can be different. Thus, after a sequence of segmentation allocation and deallocation, there could be some fragments

7. (10%) For the inverted page table architecture, please briefly explain the mechanism of inverted page table architecture for getting the physical address.



   Answer: 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.

8. (10%) When we use “paging” to manage main memory, we need to use Translation Look-aside Buffers (TLBs). (a) Please explain the purpose of TLB. (b) What is the problem if a paging memory management system does not have TLB?

   Answer:

   (a) TLB keeps the (recently) used mapping results of page numbers (of virtual address) to frame numbers (of physical address).
   (b) If there is no TLB, whenever CPU wants to get one instruction or data unit from memory, it has to access the memory device twice; once for accessing the page table (in memory) and the other one for accessing the instruction or data unit in memory.

9. (12%) There is system with only 3 memory frames. Given a reference string of pages {5→2→0→1→3→1→4→5→1→4}, please illustrate the page replacement of (a) the Least-Recently-Used (LRU) algorithm and (b) the First-In-First-Out (FIFO) algorithm. You should show the contents of memory frames and the LRU and FIFO queues.

   Answer:

   (a)

Memory Frame

5	5	5	1	1	1	1	1	1	1
	2	2	2	3	3	3	5	5	5
		0	0	0	0	4	4	4	4

LRU Queue

5	2	0	1	3	1	4	5	1	4
	5	2	0	1	3	1	4	5	1
		5	2	0	0	3	1	4	5

(b)

Memory Frame

5	5	5	1	1	1	1	5	5	5
	2	2	2	3	3	3	3	1	1
		0	0	0	0	4	4	4	4

FIFO Queue

5	2	0	1	3	3	4	5	1	1
	5	2	0	1	1	3	4	5	5
		5	2	0	0	1	3	4	4

Queue的頭在最上面或是最下面都可以，只要內容正確即可，方向不拘。
Memory frames裡面的內容如果有不必要的移動會產生額外的page writes，會扣分。

10. (8%) Please define “Copy on Write” (COW).
Answer: Copy-on-Write (COW) allows both parent and child processes to initially share the same pages in memory. When either process modifies a shared page, then the page is copied.

11. (8%) For Asymmetric Encryption, to send some critical data (which have to be encrypted), we need to have a public key and a corresponding private key. The public key can be publically sent via Internet, and the private key should be protected and be private. Why can we publically announce the public key?
Answer: The public key is used to encrypt the plaintext (to get the ciphertext), but it can not be used to decrypt the ciphertext. To decrypt the ciphertext, we need to use the private key. Thus, it is safe to announce the public key.