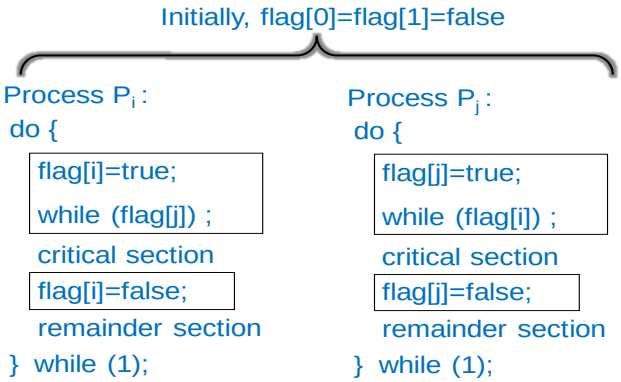
長庚大學112學年度第一學期作業系統期末測驗（滿分108）

系級: 姓名: 學號:

1. (10%) For the **second 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:



Answer: When the two processes set flag[i] and flag[j] as true, no one can break the while loop of the entry section.

2. (10%) Let's consider the **Bounded-Buffer Problem**. The pseudo code of **Consumer** is provided as follows. Please provide the pseudo code of **Producer**. After the Producer produces an item in a valuable nextp at the beginning of the loop, you have to note the position for adding the item into the buffer.

```
Consumer:
  do {
        wait(full);  /* control buffer availability */
        wait(mutex);  /* mutual exclusion */
        …….
        remove an item from buffer to nextp;
        ……
        signal(mutex);
        signal(empty);  /* increase item counts */
        consume nextp;
  } while (1);
```

Answer:

```
Producer:
  do {
        produce an item in nextp;
        …….
        wait(empty);
        wait(mutex);
        ……
        add nextp to buffer;
        signal(mutex);
        signal(full);
  } while (1);
```

3. (10%) For the Dining-Philosophers problem with the following figure, there could be a deadlock with the situation that each philosopher has picked up a chopstick on the right hand and just waits for the other at left. Please provide one remedy to the deadlock problem.

Answer:

Allow at most four philosophers.

or

Allow a philosopher to pick up chopsticks only if both are available.

or

Let some philosophers pick up their right chopsticks first and the other philosophers pick up their left chopsticks first. (asymmetric solution)

4. (10%) 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. Is the system in a deadlock state? If no, please provide a sequence to complete all processes. If yes, please still try to provide a sequence to complete as many processes as possible, and then indicate the processes which are in the deadlock.

	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	3	5	0	1			
P2	3	0	1	0	3	0			
P3	1	1	1	3	3	1			
P4	4	0	1	0	0	3			

Answer:

Yes, it is in a deadlock.

Available (0, 2, 0) → P_0 Request (0, 0, 0) → Available (0, 3, 0) → P_2 Request (0, 3, 0) → Available (3, 3, 1) → P_3 Request (3, 3, 1) → Available (4, 4, 2).

P_1 and P_4 can not be completed and are in the deadlock.

5. (12%) Please define (a) external fragmentation and (b) internal fragmentation of memory management. If we use "Paging" to manage memory, (c) is it possible to have external fragmentation? (d) is it possible to have internal fragmentation? You have to provide reasons for the answers of sub-questions c and d.

Answer:

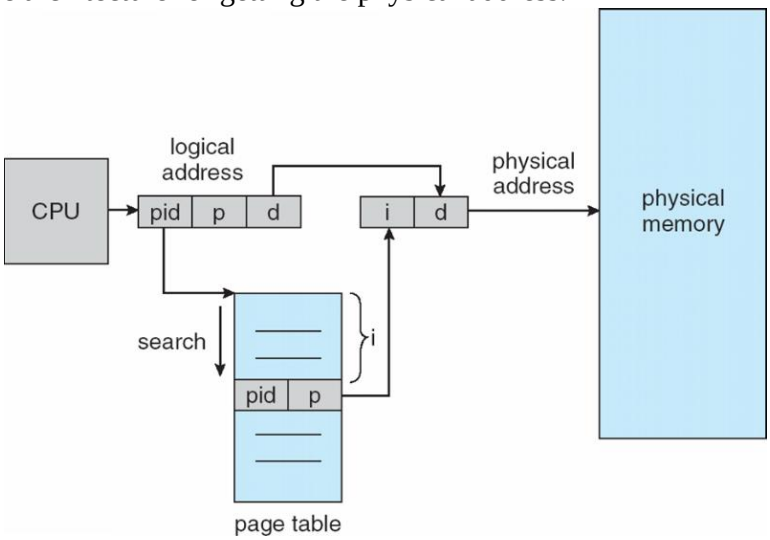
(a) External Fragmentation – total memory space exists to satisfy a request, but it is not contiguous

(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) No. All memory is partitioned in a page size, and all requests are with a page unit.

(d) Yes. For example, if we need only 3 KB memory space, there will be a 1 KB internal fragmentation in an allocated 4-KB page.

6. (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 frame number of the physical address, and d is the offset in the frame.

7. (10%) When paging is used for the memory management, please explain the method for two processes to share some binary or data.

Answer: Having some entries in the two page tables of the two processes pointing to the same physical address which contains the to-be-shared binary or data.

8. (10%) There is a system with only 3 memory frames. Given a reference string of pages {7→0→1→2→0→3→0→4→2→4→1→2→4}, please illustrate the page replacement and the queue of (a) FIFO algorithm and (b) LRU algorithm.

Answer:

(a)

7	7	7	2	2	2	2	4	4	4	4	4	4
	0	0	0	0	3	3	3	2	2	2	2	2
		1	1	1	1	0	0	0	0	1	1	1
7	0	1	2	2	3	0	4	2	2	1	1	1
	7	0	1	1	2	3	0	4	4	2	2	2
		7	0	0	1	2	3	0	0	4	4	4

← The pages in the three frames

← The victim is here

(b)

7	7	7	2	2	2	2	4	4	4	4	4	4
	0	0	0	0	0	0	0	0	0	1	1	1
		1	1	1	3	3	3	2	2	2	2	2
7	0	1	2	0	3	0	4	2	4	1	2	4
	7	0	1	2	0	3	0	4	2	4	1	2
		7	0	1	2	2	3	0	0	2	4	1

← The pages in the three frames

← The victim is here

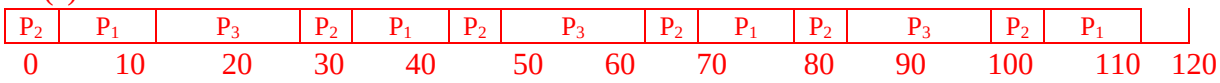
9. (8%) (a) Please explain the Copy-on-Write technique. (b) What is the benefit of using Copy-on-Write?
Answer:

(a) Copy-on-Write (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

(b) COW avoids coping all memory content and allows more efficient process creation as only modified pages are copied.

10. (10%) For three periodic tasks P_1, P_2 and P_3. P_1 has its period 30 and execution time 10. P_2 has its period 20 and execution time 5. P_3 has its period 40 and execution time 15. Please draw the scheduling results of (a) the Earliest Deadline First scheduling and (b) the Rate Monotonic Scheduling from time 0 to time 120. If there is any deadline missing, please point it out and stop the scheduling when it has the deadline missing.

Answer: (a)



(b)

Task P_3 misses its deadline at time 40.

11. (8%) For the following sample code of µC/OS-II, please explain the input parameters of function OSTaskCreate(). In other words, please explain the meaning of the parameters Task, (void*)&TaskData[i], &TaskStk[i][TASK_STK_SIZE - 1], and i + 1.

```
static void TaskStartCreateTasks (void)
{
    INT8U i;
    for (i = 0; i < N_TASKS; i++)
    {
        TaskData[i] = '0' + i;
        OSTaskCreate(
            Task,
            (void *)&TaskData[i],
            &TaskStk[i][TASK_STK_SIZE - 1],
            i + 1 );
    }
}
```

Answer:

First: An entry point of a function to be executed by the task

Second: The input data for the function

Third: A pointer to a stack to be used by the task

Fourth: The priority of the task.