2015 Operating System Midterm (CLD)

1. Please explain Deadlock and Starvation in operating system.
2. What is the meaning of the term busy waiting?
3. What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?
4. How does the signal() operation associated with monitors different from the corresponding operation defined for semaphores?
5. Consider the traffic deadlock depicted in Figure 1.

monitor

* 1. Show that the four necessary conditions for deadlock indeed hold in this example.
  2. State simple rule for avoiding deadlocks in this system.

1. Please name two differences between logical and physical memory addresses.
2. Explain the difference between internal and external fragmentation.
3. Peterson’s solution is a classic software-based solution to the critical-section problem. Please fill in the blanks in the Peterson’s solution.

| do {  \_\_\_\_\_\_\_\_\_\_;  \_\_\_\_\_\_\_\_\_\_;  while \_\_\_\_\_\_\_\_\_\_;  // critical section  \_\_\_\_\_\_\_\_\_\_;  // remainder section  } while (TRUE); |
| --- |

1. Consider a paging system with the page table stored in memory.
   1. If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?
   2. If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there.)
2. There are 4 page frames, we consider following reference string:

Use “FIFO”,” LRU” and “OPT” page replacement to replace pages. “Draw the table” to note the stages you did by using each replacement, and “count the number of page faults” for each replacement.

1. Consider the following snapshot of system:

| Process | Allocation | Max | Available |
| --- | --- | --- | --- |
|  | 0012 | 0012 | 1520 |
|  | 1000 | 1750 |  |
|  | 1354 | 2356 |
|  | 0632 | 0652 |
|  | 0014 | 0656 |

* 1. What is the content of the matrix need?
  2. Is the system in a safe state? If the answer is yes then please illustrate that the system is in a safe state by demonstrating an order in which the process may complete.
  3. If a request from process arrives for , can the request be granted immediately?

# Answers

* 1. Deadlocks  
     The waiting processes will never again change state, because the resources they have requested are held by other waiting processes.
  2. Starvation *(2017 Midterm)*  
     Process因為長期無法取得所需資源，導致無法完成工作、indefinite blocking。

1. 因為 semaphore 的 wait() 與 signal() 都是 atomic，所以進入 wait() 的 while loop 會耗用大量的 CPU 資源在等待進入 critical section。
2. *(2016 Final)*
   1. 因為 process 沒有分到足夠的 frame，因此會常常 page fault 而使得 CPU 使用率低，OS 試圖調高degree of multiprogramming 來提高CPU使用率，但反而因為 process 增加，每個 process 分到的frame 更少，CPU 使用率更低，惡性循環。

Total size of locality > total memory size.

* 1. OS 追蹤 CPU 使用率、page-fault rate，由 working-set model 或 page-fault frequency scheme 來決定是否已經產生 thrashing。
  2. 延遲部分 processes 的 frame allocation、降低部分 processes 的 frame 的數量或減少 degree of multiprogramming。



| Semaphore | Monitor |
| --- | --- |
| signal() in semaphore 必定會**改變一個值，讓要使用的 process 依據此值去判斷可否進入。** | signal() in monitor 用於**移走一個在所屬 waiting queue 中被 block 的 process**，若 waiting queue 為空，則沒有任何效果。 |

* 1. *(2016 Final)*
     1. **Mutual Exclusion:**  
        對 resource 而言，具有此性質的 resource，在任何時間點最多只允許一個 process 持有或使用，不可多個 processes 同時持有或使用。
     2. **Hold & Wait:**  
        Process 持有部分資源且又在等待其他 processes 所持有之資源。
     3. **No Preemption:**Process 不可任意剝奪其他 process 所持有之資源，必須等對方釋放資源後，才有機會取得資源。
     4. **Circuit Waiting:**  
        系統中存在一組 processes 形成循環等待之情況。
  2. There are many ways to avoid the deadlocks in this system. One way is break the second condition: a line of cars can't hold a cross and wait, it's that no car of a line can stay in the cross.

| Logical Address | Physical Address |
| --- | --- |
| * 由 CPU 產生。 * 也稱為 virtual address。 | * Memory Unit 所看到的實際位址。 |
| 在 execution time 時，logical and physical 位址會不同。 | |

1. *(2016 Final)*

| External Fragmentation | Internal Fragmentation |
| --- | --- |
| * 記憶體中的 available space (hole) 總和雖然大於 process request，但因這些空間不連續，無法配置，造成使用率降低。 * 由連續記憶體配置引起。 * 可以透過 paging 解決。 | * 系統配置給某個 process 的空間稍大於該 process 所需空間，造成浪費。 * 由 page size 太大引起。 * 可以透過 segmentation 解決。 |

1. *(2016 Final)*
   1. flag[i] = TRUE;
   2. turn = j;
   3. (flag[j] && turn = j);
   4. flag[i] = FALSE;
   5. nanoseconds.
   6. nanoseconds.  
      .
   7. FIFO *(2016 Final)*
   8. LRU *(2016 Final)*
   9. OPT → 8 次

| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 7 | 7 | 7 |  | 3 |  | 3 |  |  |  |  |  | 1 |  |  |  | 1 |  |  |
|  | 0 | 0 | 0 |  | 0 |  | 0 |  |  |  |  |  | 0 |  |  |  | 0 |  |  |
|  |  | 1 | 1 |  | 1 |  | 4 |  |  |  |  |  | 4 |  |  |  | 7 |  |  |
|  |  |  | 2 |  | 2 |  | 2 |  |  |  |  |  | 2 |  |  |  | 2 |  |  |

1. *(2016 Final)*

| Need | | | | |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 7 | 5 | 0 |
|  | 1 | 0 | 0 | 2 |
|  | 0 | 0 | 2 | 0 |
|  | 0 | 6 | 4 | 2 |

* 1. Yes. We can find a sequence is safe.
  2. Yes, there is a sequence which is safe.