NATIONAL UNIVERSITY OF SINGAPORE

CS2106 – INTRODUCTION TO OPERATING SYSTEM

(Semester 1: AY2015/16)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper consists of **SIXTEEN (16)** questions and comprises **TEN (10)** printed pages.
2. This is a **CLOSED BOOK** assessment. Two handwritten A4 reference sheets are allowed. Calculators are not allowed.
3. Answer all questions and write your answers in the **ANSWER BOOKLET** provided.
4. Fill in your Student Number with a **pen, clearly on odd-numbered pages** of your ANSWER BOOKLET.
5. You may use pencil to write your answers.
6. Marks allocated to each question are indicated. Total marks for the paper is **100.**
7. You are to submit only the **ANSWER BOOKLET** and no other document.

Questions 1 - 10: Each question has only one correct answer. Write your answers in the boxes provided in the Answer Booklet. Two marks are awarded for a correct answer and no penalty for wrong answer.

1. Suppose a compiled executable is moved from a system with 4KB page size to a new compatible system but with 8KB page size. There is no other difference in the execution environment other than the page size. Which of the following statement(s) is/are TRUE?

    i. The executable need to be recompiled due to the page size difference.
    ii. The size of page table is now ½ of the original page table size.
    iii. The size of page table entry is now ½ of the original page table entry size.

    a. (i) only.
    b. (ii) only.
    c. (ii) and (iii) only.
    d. (i) and (iii) only.
    e. (i), (ii) and (iii).

2. Under the buddy system, which of the following statement regarding a block B with starting address **40** is/are TRUE?

    i. The buddy block of B at size 4 has starting address 44.
    ii. The buddy block of B at size 8 has starting address 32.
    iii. There is no buddy block of B at size 16 because B is not valid at that size.

    a. (i) only.
    b. (ii) only.
    c. (ii) and (iii) only.
    d. (i) and (iii) only.
    e. (i), (ii) and (iii).

3. Suppose the content of a file F is spread across N disk sectors on a hard disk. Each track on the hard disk contains S sectors in total. Which of the following gives the most accurate approximation of the minimum and maximum number of **disk seeks** required to access the **entire** file F?

    a. Minimum = 1 seek, Maximum = S seeks.
    b. Minimum = S seeks, Maximum = N seeks.
    c. Minimum = N / S seeks, Maximum = S seeks.
    d. Minimum = N / S seeks, Maximum = N seeks.
    e. None of the above.

4. On a system with 1024 physical memory frames and a 32-bit virtual memory space, which of the following regarding the structure of page table is/are TRUE?

   i. Even when the physical frames are not fully occupied, the **inverted page table** contains 1024 entries.
   ii. If **direct paging** is used, the total number of page table entries for a process depends on the amount of virtual memory space used.
   iii. If **2-level paging** is used, the total number of page directory entries for a process depends on the amount of virtual memory space used.

   a. (i) only.
   b. (ii) only.
   c. (ii) and (iii) only.
   d. (i) and (iii) only.
   e. (i), (ii) and (iii).

5. OS can use the system wide "open file table" to:

   i. Allow two or more processes to share a file.
   ii. Ensure each processes have their own set of file descriptors.
   iii. Provide protection of file between processes.

   a. (i) only.
   b. (ii) only.
   c. (ii) and (iii) only.
   d. (i) and (iii) only.
   e. (i), (ii) and (iii).

6. Under the FAT file system, to locate a file with filepath "`/d1/d2/d3/f`", how many disk accesses are required? Note the following assumptions:

   - The first "`/`" represents the root directory.
   - All directory entries (including the root directory) are in disk (i.e. not cached) at the beginning. Directory entries for all directories fit in one disk block.
   - The file path is valid.

   a. 1
   b. 4
   c. 5
   d. 6
   e. None of the above

7. Which of the following statement regarding execution mode change (user mode ←→ kernel mode) is TRUE?

    a. If pure user thread is used, thread switching does not involve an execution mode change.
    b. If pure kernel thread is used, thread switching does not involve an execution mode change.
    c. Process termination does not involve an execution mode change.
    d. Closing a file does not involve an execution mode change.
    e. None of the above.

8. Which of the following statement(s) is/are TRUE regarding the **interval between time interrupt (ITI)** and **time quantum (TQ)**?

    i. Shorter ITI (i.e. more frequent timer interrupt) but with the same TQ can increase the responsiveness of processes.
    ii. Shorter ITI reduces the overall CPU time spent on processes.
    iii. Shorter TQ reduces the overall CPU time spent on processes.

    a. (i) only.
    b. (ii) only.
    c. (ii) and (iii) only.
    d. (i) and (iii) only.
    e. (i), (ii) and (iii) only.

9. Which of the following statement(s) regarding the **zombie process state in Unix systems** is/are TRUE ?

    i. All child process will transit to zombie process state upon normal termination.
    ii. A process will transit to zombie process state even if it was terminated due to exception (e.g. segmentation fault).
    iii. Zombie process state was created so that wait() or similar system call can be performed correctly.

    a. (i) only.
    b. (ii) only.
    c. (ii) and (iii) only.
    d. (i) and (iii) only.
    e. (i), (ii) and (iii).

10. Which of the following statement(s) regarding the "Limited Seat" solution for the dining philosopher problem (reproduced below for your reference) is/are TRUE?

```
seats = Semaphore( 4 );
fork = Sempahore(1)[5]; //array of 5 semaphores
                        //each with initial value = 1
void philosopher( int i ){
    while (TRUE){
        Think( );
        wait( seats );
        wait( fork[LEFT] );
        wait( fork[RIGHT] );
        Eat( );
        signal( fork[LEFT] );
        signal( fork[RIGHT] );
        signal( seats );
    }
}
```

i. A philosopher task can still stuck on the "wait(fork[LEFT])" under this solution.

ii. There could be up to 4 philosophers performing the "Eat( )" at the same time.

iii. A philosopher task can still stuck on the "wait(fork[RIGHT])" under this solution.

a. (i) only.
b. (ii) only.
c. (ii) and (iii) only.
d. (i) and (iii) only.
e. (i), (ii) and (iii) only.

11. **[10 marks]** Suppose we have two types of tasks on a system: **Coffee** and **Milk**. There are many tasks of each type running at any point in time. If we have a special "critical section" with the following requirement:

- Only **a pair of** 1 Coffee and 1 Milk task can enter the critical section at any time.
- Both Coffee and Milk must enter the critical section at the same time. (i.e. a single coffee task alone or a single milk task alone cannot be allowed to enter).

One possible solution is to use an additional task called the **Mixer** for monitoring the entry condition.

a. **[2 marks]** Give the initialization value for each of the semaphore. You are allow to use **at most 4 semaphores** to solve this problem.

b. **[8marks]** Give the implementation of the following 3 functions:

i. ***CoffeeEnterCS( ):*** Coffee tasks call this when reaching the critical section.

ii. ***MilkEnterCS( ):*** Milk tasks call this when reaching the critical section.

iii. ***MixerMonitor( ):*** Mixer task run this "forever" to provide correct access to the critical section.

Important notes

- All functions are just a series of **wait() / signal()** calls. **You are not allowed to use any other programming construct in the functions.**

- Your task is just to ensure the entry condition of the "special critical section" is met. There is no need to consider how/when the coffee/milk task exit the critical section.

12. [**16 marks**]

a. [**4 marks**] During the execution of a process, the OS may still take over the CPU sometimes. Give one example for each of the following scenarios.

   i. The running process explicitly invokes the OS to take over.

   ii. The OS takes over independent of the current running process.

b. [**6 marks**] Consider the following pure CPU tasks and arrival time:

Task	Arrival Time	Total CPU time needed
A	0	3
B	0	4
C	3	4
D	5	2

Give the CPU schedule under Round Robin Algorithm with a time quantum of 2 time unit. Just indicate the tasks in the order of their execution.

You can assume: task arrives in the ready queue before the scheduler is triggered for that time unit. (e.g. at time 0, scheduler will have task A and B in ready queue for consideration).

c. [**6 marks**] Given the following definition (all units are in ***seconds***):

Definitions	
Tcs:	Time to context switch between two processes.
ITI:	Interval between timer interrupt.
TQ:	Time quantum.
Tsch:	Time to perform scheduling bookkeeping, excluding the cost for actual context switch.

Express the overhead of scheduling per second of the **round robin scheduling algorithm**.

13. [**10 marks**]

a. [**6 marks**] Given 3 memory frames, consider the following memory reference strings:

A	1 2 3 4 4 3 2 5 6 2 3
B	6 3 4 2 3 4 2 1 5 3 4

Give the number of page faults for OPT, LRU and Second Chance page replacement algorithms.

b. [**4 marks**] Give 1 advantage and 1 disadvantage of the Second Chance algorithm as compared to LRU algorithm.

---

14. [**16 marks**] We can protect a memory page by adding permission bits to the page table entry (PTE). Suppose we add 3 bits: {**R**: Readable, **W**:Writable, **X**: Executable} to each PTE, answer the following briefly:

a. [**2 marks**] What type of process instruction requires the checking of the "W" bit?

b. [**2 marks**] If processor issues a "fetch instruction from page X", what are the permission bits to be checked?

When a processor instruction violates the access permission of a page, OS will be invoked to handle the problem. We can utilize this behaviour to implement the **copy-on-write** mechanism (memory pages are shared between parent and child until written).

Suppose process P has only 3 valid page table entries:

Page No	**Frame No**	**R**	**W**	**X**
0	7	1	0	1
1	2	1	1	0
2	5	1	0	0
3 …. N-1	---	---	---	---

c. [**4 marks**] Give the page table entries for the **child process** after P executed a **fork()** system call. If you need to use any new frame numbers, use them in this order {6, 0, 3, 4 1}.

Using the above scenario, explain the following clearly:

d. [**2 marks**] How does the OS know copy-on-write is needed?

e. [**6 marks**] What are the steps required to handle copy-on-write? Indicate any additional information that OS need to maintain. Show the affected PTE(s) for the child process afterwards.

15. **[12 marks]**

a. **[4 marks]** Give the disk I/O schedule for the following I/O requests using **Shortest Seek First (SSF)** algorithm:

Time	0	1	1	1	2	2	3	4
Request	19	2	6	9	11	17	4	5

For simplicity, we assume that all disk I/O requests can be completed in one time unit. An I/O request will be considered by the algorithm as soon as it arrives.

b. **[4 marks]** Rearrange the above requests (both timing and order) such that it gives the **same schedule** under the C-SCAN algorithm. We assume the seek direction goes from track 1 to track 20. The requests should be issued as early as possible.

c. **[4 marks]** Similarly, rearrange the above requests such that it gives the same schedule under the SCAN algorithm. The seek direction goes from track 1 to track 20, then track 20 back to track 1. The requests should be issued as early as possible.

---

16. **[16 marks]** Use the partial ext2 file system snapshot on the last page to answer the following questions. Note that only the crucial information are shown. **The "/" directory has the I-node number 8.**

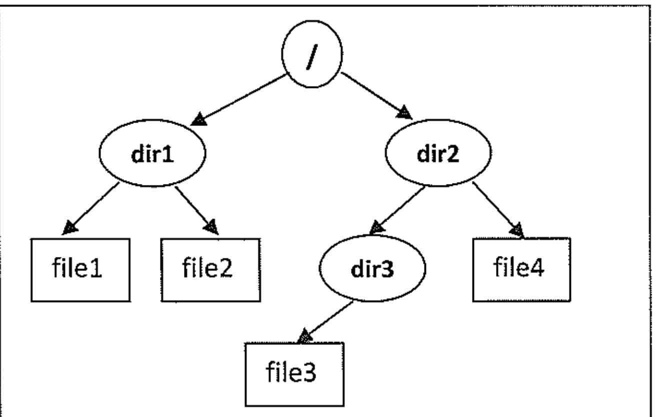
a. **[4 marks]** Give the sequence of **I-Node** accessed to **"list the content"** of directory **"/Zeta"**.

b. **[4 marks]** Give the sequence of **data blocks** which stores the content for the file **"/Zeta/Alpha"**.

c. **[8 marks]** Follow either of the following examples to draw / describe the entire directory structure **starting from the "/" directory.** Note that not all information given in the snapshot is linked to "/" directory.

d.

```
/
    dir1/
        file1
        file2
    dir2/
        dir3/
              file3
        file4
```



I-Node Table

1	9 7 ……
2	5 10 ……
3	12 ……
4	1 2 ……
5	3 ……
6	4 ……
7	13 14 ……
8	6 ……
9	… … … …

Disk Blocks

1	5 | F | 5 | Gamma 2 | F | 4 | Beta ……
2	4 | F | 5 | Alpha 3 | D | 5 | Gamma ……
3	3 | D | 5 | Gamma 1 | F | 5 | Alpha ……
4	2 | F | 6 | Lambda 4 | F | 5 | Delta ……
5	7 | F | 4 | Zeta 3 | F | 5 | Gamma ……
6	5 | D | 4 | Zeta 6 | D | 4 | Beta ……
7	2 | F | 5 | Sigma 7 | F | 4 | Beta ……

8	1 | F | 5 | Sigma 6 | D | 5 | Gamma 4 | D | 5 | Lambda ……
9	5 | F | 5 | Gamma 2 | F | 4 | Beta ……
10	2 | F | 5 | Sigma ……
11	3 | D | 5 | Gamma 1 | F | 5 | Alpha ……
12	2 | F | 5 | Sigma ……
13	4 | F | 5 | Alpha ……
14	5 | D | 4 | Zeta 6 | D | 4 | Beta ……
….	… … … … …

~~~~ END OF PAPER ~~~~