National University of Singapore
CS2106 – Introduction to Operating Systems

(Semester 2: AY2018/2019)
Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. **Please write your Student Number below**. Do not write your name.
2. This assessment paper contains **Four** questions and comprises **Twelve** pages, including the cover page. **BOTH** sides of the page are used.
3. Answer **ALL** questions within the space provided in this booklet. You may use a pencil as long as your writing is LEGIBLE and CLEAR.
4. This is an **CLOSED BOOK** assessment. You are allowed a single A4-sheet of notes.
5. The **TOTAL** marks is 100.
6. Calculators may be used.

STUDENT NO: _____

Examiner's Use Only - do not write in this table

Question	Total	Marks
Q1	48	
Q2	18	
Q3	18	
Q4	16	
Total	100	

QUESTION 1 (48 marks)

Answer all sub-questions. Each of the 8 sub-questions is worth 6 marks and are independent of each other. Answers can be brief but should explain what the sub-question asks.

(a) In Unix (& Linux), the OS provides certain support to C function calls in processes. (i) Explain **two** ways where the OS has an effect on function calls. (ii) Explain briefly how the kernel supports (i).

(b) You are designing the software for a new Lunar lander. Space certified hardware has limited features. The software should be as correct as possible. Code execution must meet real-time requirements, e.g. sensor data is to be collected and processed regularly. You can choose between two architectures: (i) no operating system with custom software including assembly code; (ii) building on top of Linux (the processor has Linux support). Pick a choice and briefly give **3 reasons** supporting that choice given the above requirements.

QUESTION 1 (continued)

(c) In Unix, the `kill()` system call is to "kill" a running process. A process is *completely gone* when no more information can be found on the process. Two cases can happen after the `kill()` system call is used to terminate process p: (i) process p is completely gone; (ii) there remains some information on p. Explain why each case can happen, giving a specific example to show how it happens.

(d) Suppose `usleep(`n`)` is implemented as a system call to suspend the process for at least n microseconds. Alternatively a user-mode delay loop, `delay(`d`)`, can be written which executes for n microseconds. A process needs to perform a certain operation by calling a function `f()` and each call to `f()` must be separated by at least time T. Give two scenarios: (i) `usleep()` is preferred over `delay()`; and (ii) `delay()` is preferred over `usleep()`. Explain your reasoning in both scenarios.

(e) Suppose you want to implement the `malloc` library. Describe an implementation strategy for `malloc` (the code is not needed) and how it affects: (i) speed of `malloc()` operation; (ii) amount of internal fragmentation; and (iii) amount of external fragmentation.

(f) The following are file APIs: `write()`, `printf()`. Which is a Unix system call? Give a reason for using `write()` over `printf()` and another reason for using `printf()` over `write()`.

(g) Suppose the current working directory for process p in Unix is "`/tmp/u`" and we want to access file "`/tmp/u/x`". Give 6 pathnames for the same file in process p where the pathname length is is up to 6 characters long.

(h) Let S be a binary semaphore. Using only semaphore S, give pseudo-code for implementing a critical section. In addition, give modified code for the critical section code which always deadlocks.

QUESTION 2 (18 marks)

In this question, parts (a) and (b) are related.

(a) [9 marks]
Suppose Machine A has the following two machine instructions: (i) `SWITCH_KERNEL` which puts the processor in kernel mode; (ii) `USER_MODE` which puts the processor in user mode. In user mode, privileged instructions including those used to deal with devices are not allowed, and OS kernel memory cannot be accessed. In kernel mode, there are no restrictions.

Suppose the goal is to implement Unix and provide protections between processes and Unix file permissions. Explain whether or not Machine A can be used for this purpose. Your explanation must give a pseudo-code sketch (including the instructions) on how system calls would be made. You must also explain how the pseudo-code meets or does not meet the goals. (Hint: assume the program is malicious, i.e. malware, how well does OS on Machine A work?).

(b) [9 marks]
Propose instructions for Machine B which can improve over Machine A for the given goal of providing Unix protections and permissions. Describe what the new instructions in Machine B do. Give pseudo-code using the new instructions for system calls. Explain why the code in (b) improves over (a).

QUESTION 3 (18 marks)

In this question, parts (a) and (b) are related.

(a) [9 marks]
The `void runproc(char *cmdline, char *altcmd)` function creates a new process in Unix where `cmdline` gives the *command line* of the process to be executed. If the `cmdline` process cannot be created or the `cmdline` executable cannot be executed, then it attempts to create an alternate process to execute the command line given in `altcmd`. `runproc()` will wait for the process created (if any). Give a C-like pseudo-code implementation for `runproc()` for Unix (or Linux). A command line consists of words (sequence of non-space characters) separated by a space (you can assume it is exactly one space).

QUESTION 3 (continued)

(b) [9 marks]
In principle, it could be possible to turn `runproc()` into a system call. Give three reasons why it is not sufficient to replace the "process-relevant" normal Unix system calls by a single `runproc()` system call.

QUESTION 4 (16 marks)

In this question, parts (a) and (b) are related.

The OS uses paged virtual memory with 32-bit virtual addresses. A C program uses an array `X` of 32-bit integers and is compiled into executable E. A process p running E has so far only accessed the following array elements of `X`: `X[0]`, `X[100]`, `X[2000]`, `X[16384]` ($2^{14} = 16384$).

(a) [10 marks]
The OS runs on a machine using 2-level paging with 4K pages. The page table entries (PTEs) are 32 bits and a page table stores 1024 PTEs. Assume that no pages are evicted for `X`. Draw the current 2-level page table for process p showing **only** the entries for `X`. Any non-zero entries should be labelled with their corresponding index in the page table and page directory. The diagram should ignore portions of the page directory and page tables which are not relevant to `X`. A diagram without explanatory text is sufficient as long as it clearly explains the virtual addressing for the accesses to `X`.

QUESTION 4 (continued)

(b) [6 marks]
Suppose the size of `X` exceeds 512MB (2^{29}) of memory. The machine only has 1GB (2^{30}) of memory. Briefly explain how paged virtual memory allows 4 processes to run E concurrently and why `X` can have the same virtual address in all processes.

End of Questions