Question Sheet for Introduction to Operating Systems@CS_NYCU, Fall 2021

**Notice: Provide sufficiently clear explanation in your answers to get full credits

1. [C1+13, 4pts] What is a trap? Enumerate two different reasons for triggering a trap.
2. [C1+13, 6pts] A latency exists between the firing of a hardware interrupt signal and the execution of the first line of code for the interrupt service routine (ISR). List at least three factors that contributes to the latency.
3. [C2, 8pts] Regarding the separation of user mode from kernel mode, answer the following:
   a) Why the dual-mode execution is a common feature in commodity desktop operating systems, such as Windows and Linux?
   b) The separation is rarely seen in deeply embedded operating systems. Why?
4. [C2, 8pts] Regarding the kernel structure, answer the following
   a) Microkernels are considered more advanced compared with monolithic kernels. Give two reasons for that.
   b) However, monolithic kernels are more common in desktop computer operating systems. Give a reason for it.
5. [C3, 6pts] For each of the following process state transition, give a possible reason that triggers each of the following transitions:
   a) Running to ready
   b) Ready to running
   c) Waiting to running
6. [C3, 10pts] Describe the procedure of interrupt-driven context switch (e.g., time slice expiration). Your answer must at least involve the following items: timer interrupt, interrupt vector table, interrupt service routine, process stack, and program counter.
7. [C4, 4pts] When programming with a thread library, it is important to know the threading model of the library, i.e., many-to-one, one-to-one, etc. This information is usually available in the programmer's manual of the library. Now, consider that you have no access to the manual. How do you determine that the threading model of the library is many-to-one or one-to-one? Please elaborate.
8. [C4, 6pts] Which one(s) of the following can be accessed by all threads of a multi-threaded process? Why?
   a) A global variable
   b) A piece of malloc()'ed memory
   c) A local variable in a function
9. [C5, 4pts] Prove that Shortest-Job-First (SJF) is optimal in terms of the average waiting time for a set of processes whose arrival times are all at time 0.
10. [C5, 6pts] Consider a NUMA system in which each CPU is attached to a piece of private memory. Private memory access is much faster than remote memory access, and it is possible

to move data between CPU private memories but the movement requires a high time overhead. Now consider global scheduling and partitioned scheduling as the CPU scheduling algorithm. which one better fits this system? Why?

11. [C5, 6pts] In the multilevel feedback queue algorithm, an interactive process will be escalated to a high priority with a small quantum. Explain why the algorithm is designed this way.
12. [C5, 6pts] In the Linux CFS algorithm, the virtual runtime of a CPU-bound process increases faster than other processes. Explain why.
13. [C6, 6pts] As a solution to the critical section problem, the test-and-set instruction operates on a target memory and the operation must satisfy two properties: 1) the instruction must be atomic and 2) the instruction has exclusive access to the target memory. Explain why the two conditions must hold.
14. [C6, 6pts] There are two typical hardware-based solutions to the critical section problem: 1) interrupt disabling and 2) atomic test-and-set instruction. Now consider a system that has only one CPU and there is no separation of the user mode from the kernel mode. Which one of the two solutions better fits this system? Why?
15. [C6, 8pts] As per the government regulation, in a childcare center, there must be at one adult per three children. Write pseudo code for adults and children to enforce this rule in a critical section. Hint: A solution uses two semaphores.
16. [P1, 2pts] Explain how you handle zombie processes in the first programming assignment and why it works.
17. [P2, 2pts] Why before accessing a piece of shared memory, a process must calls shmat() once on the memory?
18. [P3, 2pts] In programming assignments 2 and 3, when the degree of thread/process parallelism is not high, adding more threads/processes significantly improves the program running time. However, this improvement becomes small when the degree of thread/process parallelism is already high. Explain why.