Introduction to Operating Systems
Midterm Question Sheet
November 2019

You will get zero point from your answer lithe answer conies without a meaningful explanation.

1. [6 pts] Describe the life cycle of an I/O operation. Your answer should involve interrupt vectoring, and Direct-Memory Accessing. ¯11 7c)
2. [6 pts] Polling I/O and interrupt-driven I/0 are two schemes for input-output operations.
   a) In most cases, interrupt-driven i/Os use less CPU cycles than polling I/Os. Explain why.
   h) In what circumstance a device driver may choose to use polling I/Os instead of interrupt-drive I/0s?
3. [5 pts] List five different types of system call that a UNIX shell program may involve.
4. [6 pts] A microkernel-based operating system moves most of its functionalities to the user land while keeping a small, essential set of services in the kernel space. Compared to conventional monolithic operating systems, a microkernel-based one is more secure, robust, and configurable.
   a) However, a microkernel-based operating system imposes a higher time overhead on user processes. Explain why.
   b) Consider that the I/O path of user processes involves a user process, a file system server, and a disk server. How many context switches are necessary to perform a synchronous I/O operation?
5. [9 pts] Regarding processes in UNIX operating systems:
   a) What is an orphan process?
   b) What is a zombie process?
   c) Is a zombie process necessarily an orphan?
6. [8 pts] There are three process states: running, ready, and waiting. There are various queues in the operating systems, including the CPU ready queue, I/O waiting queues, and semaphore waiting queues. Consider a process that is currently in the running state. Show the *state transition* and *queue migration* of the process in the following scenarios:
   a) Performing a synchronous I/O operation (such as fread())
   b) Being blocked on a semaphore and later acquiring the semaphore.
7. [4 pts] Why context switch is implemented as an interrupt handler rather than a regular procedure call?
8. [6 pts] The benefit of multithreading is subject to various factors, including hardware and software structures. Give two examples for which multithreading barely improves the performance of a single-thread program.
9. [6 pts] Why it is necessary for threads to maintain their own stacks?
10. [6 pts] Does a low average waiting time always imply a low maximum waiting time? Justify your answer.
11. [8 pts] Consider a multilevel-feedback queue CPU scheduler based on the following parameters > There are three queues QO, Ql, and Q2. A new process enters Q2.

    The scheduling policy for QO is RR with a time slice 8ms, that for Q1 is RR with a time slice 16 ms, and that for Q2 is FCFS.

- Processes in QO are serviced first, while those in Q2 are serviced last.

}= A process is promoted to the next higher-level queue (e.g., Q2-3Q1) if it returns from sleep, and is demoted to the next lower-level queue if its time slice expires.

a) Suppose that the CPU burst lengths of interactive (i.e., 11/0-bound) processes are usually shorter than 5 ms. After a long-term use, in which queue will these processes appear?

b) In which queue will CPU-bound processes (whose CPU bursts are long) appear?

12. [4 pts] I/O wait is the proportion of time that a process spends on waiting for I/O completion. Multiple jobs can run on a single CPU with multitasking and finish faster than if they had run sequentially without multitasking. Suppose that two jobs, each needing 20 minutes of time to complete if running alone. Assume 50% I/O wait for each process. How long will the last one take to complete if the two jobs run sequentially without multitasking? How long if they start simultaneously and run with multitasking?

13. [8 pts] Regarding inultip“,...sso, CPU scheduling

a) Processor affinity and load balance are two crucial factors of this problem. What do they mean?

b) Continued from the prior question. Why these two factors usually conflict with each other?

14. 1-8 pts] Spinlocks and interrupt disabling are two basic solutions to the critical section problem. Now consider an embedded system operating system, in which applications and the kernel both runs in the privilege/kernel mode for efficiency. The system has only one CPU for cost reduction. Which one of the aforementioned solutions should the embedded operating system adopt?

15. r5 pts] Consider two processes, P1 and P2, that share a string buffer s[] of 100 characters. Let s[] has been ,:reated in a piece of shared memory between PI and P2. Consider that P1 can only produce characters of 'A' while P2 can only produce characters of 'W. Write a program with semaphores such that P1 and P2 produce a repetitive pattern "ABBA" in the string buffer. Notice that buffer overflow must be prevented.

16. 15 pts] Let there be N (N>10) threads. These threads-are based on the following code fragment

```
thread _entry(){
    …
    printf("ready");
    …
     do_something()
    …
    };
```

Use semaphores to synchronize these threads such that no thread calls do_something() until all threads have printed "ready" (you don't have to worry about I/O buffering). No more than 3 semaphores are allowed.

Hint: only 2 semaphores are required to solve this problem