California State University, Sacramento
CSC 139 Operating System Principles
Midterm Exam 1, Spring 2018

57

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: Danny Pham

Section: 4

Multiple Choice (3 points each)

1. An interrupt is
   A. a temporary suspension of the PC clock signal to allow device controllers to access memory.
   B. a suspension of code executing in a critical section.
   (C) a signal that causes the control unit to branch to a specific location.
   D. the exiting of a loop due to a break statement.

2. When a process is created using the classical `fork()` system call, which of the following is NOT inherited by the child process?
   A. Process address space
   B. Process ID
   C. User ID
   D. Open files
   E. Signal handlers
   (F) None of the above

30 X

3. A race condition is when
   A. multiple processes are all trying to finish first.
   B. a process runs too slow.
   (C) the correctness of the code depends upon the timing of the execution.
   D. a set of processes is waiting for an event that only another process in the set can cause.

4. The text segment of a process address space contains
   A. the statically allocated data associated with the process.
   B. the dynamically allocated data associated with the process.
   (C) the executable code associated with the process.
   D. the inter-process communication (IPC) messages for the process.
   E. all of the above

5. Which of the following would lead you to believe that a given system is an SMP-type system? (Symmetric)

   A. Each processor is assigned a specific task.

   B. There is a bossworker relationship between the processors.

   (C) Each processor performs all tasks within the operating system.

   D. None of the above

6. Embedded computers typically run on a _____ operating system.

   (A) real-time

   B. Windows XP

   C. network

   D. clustered

7. A message-passing model is

   (A) easier to implement than a shared memory model for inter-computer communication.

   B. faster than the shared memory model.

   C. a network protocol, and does not apply to operating systems.

   D. only useful for small simple operating systems.

8. The major difficulty in designing a layered operating system approach is

   A. making sure each layer is easily converted to modules.

   B. making sure that each layer hides certain data structures, hardware, and operations.

   C. debugging a particular layer.

   (D) appropriately defining the various layers.

9. Which is true about processes and threads?

   X (A) Threads in a process share the same stack.

   B. Threads in a process share the same file descriptors.

   C. Threads in a process share the same register values.

   D. Threads in a process share the same program counter.

10. Most often, application programs access system resources using _____.

    A. system calls

    B. kernel threads

    X (C) user threads

 Please go on to the next page...

D. application program interfaces

11. For a single-processor system _____.
    A. processes spend long times waiting to execute
    B. there will never be more than one running process
    C. input-output always causes CPU slowdown
    D. process scheduling is always optimal

12. A thread control block _____.
    A. is managed by the parent process
    B. contains the same information as the process control block
    C. has the identical structure as the process control block
    D. does not include information about the parent process resource allocation

13. The Producer-Consumer Problem is related to _____.
    A. the handling of process control blocks
    B. the scheduling of process states
    C. the allocation of resources to process states
    D. Both A and C are correct

14. When a process is accessing its heap space, it exists in the _____.
    A. Running state
    B. Waiting state
    C. Terminating state
    D. Ready state

15. Long-term scheduling is performed _____.
    A. typically on submitted jobs
    B. when processes must be moved from waiting to ready state
    C. on processes in the ready queue
    D. All of the above are correct

True or False (2 points each)

16. True/False T The two primary purposes of an operating system are to manage the resources of the computer and to provide a convenient interface to the hardware for programmers.

17. True/False F Each thread of a process has its own virtual address space.

18. True/False F A deadlock-free solution eliminates the possibility of starvation.

19. True/False T An interrupt vector contains the addresses of the handlers for the various interrupts.

20. True/False T The code that changes the system clock runs in user mode. F

21. True/False T A thread can be blocked on multiple condition variables simultaneously. F

22. True/False T It is possible to have concurrency without parallelism.

23. True/False F The main difference between the use of test-and-set and the use of semaphores is that semaphores require the OS to do the busy waiting rather than the user program.

24. True/False T Aging can alleviate the starvation problem of a low priority job.

25. True/False T In a monolithic kernel, most operating system components, e.g., memory management, inter-process communication, and basic synchronization modules, execute outside the kernel. F

Short Answer (5 points each)

26. What is the purpose of a process control block (PCB)? When is it updated and when is it read by the operating system?

The PCB describes the way a process is handled. It shows the steps in what a process takes place on the stack. It is updated when the counter is incremented, and read when put onto READY state.

2

27. Given two processes in the READY state, one that is CPU-bound and one that is I/O bound, which process should be given a higher priority for running next (all other things being equal)? Justify your answer.

CPU-bound - more time doing computations. This should be given a higher priority when in the READY state. It contains more cpu bursts than I/O

2

28. Describe the difference between the *wait* and *signal* operations of a semaphore and a condition variable (of a monitor).

Semaphore
- wait(), signal()

Monitor
- X.wait(), X.signal()
- only one process at a time

1

Long Questions

29. (10 points) Suppose two threads execute the following C code concurrently, accessing shared variable a, b, and c.

```
//Initialization
int a = 4;
int b = 0;
int c = 0;

//Thread 1
if (a < 0) {
    c = b - a;
} else {
    c = b + a;
}

//Thread 2
b = 10;
a = -3;
```

c = 10 - 3 = 7 c = 10 - (-3) = 13
c = 0 - (-3) = 3
c = 0 + 4 = 4
c = 10 + 4 = 14

What are the possible values for c after both threads complete? You can assume that reads and writes of the variables are atomic, and that the order of statements within each thread is preserved in the code generated by the C compiler.

atomic - uninterrupted process

Thread 1 - c = 4 ✓

Thread 2 - c = 7 ✓

-3, 13, 14 also possible

4

30. (10 points) The code below contains a proposed solution to the Producer-Consumer Problem. Assume that `full` is initialized to 0, and `empty` is initialized to the number of slots in the buffer.

```
//Producer Process
do {
    ...
    produce an item in nextp
    ...
    wait(empty); /*block if no empty slots*/
    wait(mutex);
    ...
    add nextp to buffer
    ...
    signal(mutex);
    signal(full); /*increment full slots count*/
} while (true);

//Consumer Process
do {
    wait(full); /*ensure there is something to consume*/
    wait(mutex);
    ...
    remove an item from buffer to nextc
    ...
    signal(mutex);
    signal(empty); /*increment empty slots count*/
    ...
    consume the item in nextc
    ...
} while (true);
```

(a) Why does this proposed solution use 2 semaphores, `empty` and `full`, rather than just one (call it `count`)?

There are two semaphores because we need to keep track of the amount of number of slots in the buffer.

5

(b) Is it a problem if we switch the order of the two lines `wait(empty)` and `wait(mutex)` in the producer process? Justify your answer.

No, switching the wait(empty) and wait(mutex) will not cause a problem. Only changing the order of the signal method will make the program to differ