**Operating Systems Principles**

**First Quiz**

**Wednesday, June 12th, 2024**

**Instructor: Ghassan Shobaki**

**Student Name: Jacob Francis Student Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Q1. In the questions below, circle the right answer. There is only one correct answer.** (36 points)

1. Which of the following is **not** true about context-switching?

a. Context switching is performed in privileged mode.

b. Before giving the CPU to the next process, the kernel must change the mode bit to user mode.

c. Context switching time depends on the memory speed.

d. Certain hardware features may enable the kernel to perform context switching faster.

e. The very first thing that the kernel does in context switching is invoking the scheduler.

f. Saving the state of the process that got interrupted must be done before updating the ready queue.

1. Which of the following hardware features is (are) required to enable the OS to disallow illegal operations?

a. A multi-core processor

b. A counter that sets up a timed interrupt

c. A large memory

d. Dual mode of operation

e. Exceptions

f. Both b and e are correct. g. Both c and d are correct. h. Both d and e are correct.

1. What is (are) the disadvantage(s) of using a shorter time quantum in a time-sharing OS?

a. With a shorter time quantum, some processes won’t get enough CPU time to complete their work.

b. With a shorter time quantum, more time will be spent in the kernel context switching.

c. With a shorter time quantum, the system will feel less responsive.

d. With a shorter time quantum, a CPU-bound process won’t work correctly.

e. Both a and b are correct. f. Both b and c are correct. g. Both b and d are correct.

1. When the kernel performs context switching from P1 to P2, in which order will it perform the following steps?
2. Saving P1 state, restoring P2 state, updating the ready queue, and finally invoking the scheduler.
3. Saving P1 state, invoking the scheduler, updating the ready queue, and finally restoring P2 state.
4. Updating the ready queue, invoking the scheduler, saving P1 state, and finally restoring P2 state.
5. Saving P1 state, updating the ready queue, invoking the scheduler, and finally restoring P2 state.
6. Restoring P2 state, saving P1state, updating the ready queue, and finally invoking the scheduler.
7. Different operating systems perform these steps in different orders.
8. These steps can be performed in any order.

1. How does the loadable module (Mod) structure compare with the monolithic (Mono) structure?
2. In Mono, if one service crashes, it will crash the whole kernel, but in Mod it won’t.

b. From a performance (speed) point of view, Mod is better than Mono.

c. From a software engineering point of view, Mod is better than Mono.

d. In Mod, there is a hierarchy of services, while in Mono there is no hierarchy.

e. Both a and c are correct. f. Both c and d are correct. g. a, c and d are correct.

1. Which OS structure(s) may load services at runtime into separate address spaces?

a. Microkernel b. Layered c. Loadable modules d. Monolithic

e. Both a and b are correct. f. Both c and d are correct. g. Both b and c are correct.

**Q2.** The table below shows three events that may cause a transition in a process’s state. For each event, the table shows the state transition that immediately follows the event and the queue in which the process gets placed after the transition. Fill in the missing entries in the table. If there are multiple possible causing events, only give one but describe it precisely (**don’t just say “interrupt”, because all events cause interrupts**). If there is no queue, write **none**. (30 points)

| Causing Event | Transition  From State To State | Queue after transition  (if none, enter **none**) |
| --- | --- | --- |
| Scheduler dispatch of a process | Ready Running | None |
| Process performs a divide by zero | Running Terminated | None |
| Process requests I/O | Running Waiting | Wait Queue |

**Q3.** An OS running on a **single CPU** has the following two processes:

* P1 consists of a 50-ms CPU burst followed by an I/O request followed by a 40-ms CPU burst
* P2 consists of a 20-ms CPU burst followed by an I/O request followed by a 30-ms CPU burst

Draw a complete timing diagram showing how a time-sharing operating system will handle these processes under the following assumptions:

* Each I/O request goes to a **different** device, P1’s I/O request takes 60 ms to process, while P2’s I/O request takes 120 ms to process.
* The kernel starts each I/O request right at the beginning of the kernel time.
* The I/O device will start processing a request as soon as the current request is done (immediately if it is not currently processing another request and the queue is empty).
* Whenever P1 and P2 are **both** ready, the scheduler will give the CPU to the process that has been sitting in the ready queue for a longer period of time, that is, **the ready queue is a First-In-First-Out (FIFO) queue**. When the kernel handles an I/O completion interrupt, it enters the interrupted process into the ready queue **before** entering the process whose I/O request has been completed. For the very first selection, assume that the scheduler will select **P1**, that is, **P1**will get the CPU at time 10.
* The time quantum is 30 ms
* The kernel time (including initiation of I/O requests, scheduling and context switching) is 10 ms.
* Ignore any delay or transition time that was ignored in class. (34 points)