Tecnológico de Costa Rica

31 de agosto de 2017

Escuela de Ingeniería en Computación
Principios de Sistemas Operativos

Samantha Arburola León
2013101697

Operating System Concepts

Chapter 5 - Practice Exercises

5.1 In Section 5.4, we mentioned that disabling interrupts frequently can affect the system's clock. Explain why this can occur and how such effects can be minimized

Disabling interrupts frequently: It is updated at every clock interrupt. If interrupts were disabled, it is possible the system clock could lose the correct time

It affects on programming schedules, so if clock interrupts were disabled, the scheduler could not accurately assign time quantums.

How such effects can be minimized? By disabling clock interrupts for only very short periods.

5.2 Explain why Windows, Linux, and Solaris implement multiple locking mechanisms. Describe the circumstances under which they use spinlocks, mutex locks, semaphores, adaptive mutex locks, and condition variables. In each case, explain why the mechanism is needed.

They provide locking mechanisms depending on the application developers' needs
Spinlocks: useful for multiprocessor systems because a thread can run in a busy-loop, rather than incurring overhead of being put in a sleep queue
Mutexes: useful for locking resources
Solaris: uses adaptive mutexes; the mutex is implemented with a spin lock on multiprocessor machines
Semaphores and condition variables: appropriate tools for synchronization when a resource must be held for a long period because spinning is inefficient for a long duration.

5.3 What is the meaning of the term busy waiting?
It means that a process is waiting for a condition to be satisfied in a tight loop without relinquishing the processor.

What other kinds of waiting are there in an operating system?
A process could wait by relinquishing the processor, and block on a condition and wait to be awakened at some appropriate time in the future

Can busy waiting be avoided altogether? Explain your answer
It can be avoided but incurs the overhead associated with putting a process to sleep and having to wake it up when the appropriate program state is reached.

5.4 Explain why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems.
They are not appropriate for single-processor systems because the condition that would break a process out of the spinlock can be obtained only by executing a different process and if the process is not relinquishing the processor, other processes do not get the opportunity to set the program condition required for the first process to make progress

5.5 Show that, if the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated
If 2 wait operations are executed on a semaphore when its value is 1, if the 2 operations are not performed atomically, it is possible both might proceed to decrement the semaphore value, thereby violating mutual exclusion.

5.6 Illustrate how a binary semaphore can be used to implement mutual exclusion among n processes
Mutex initialized to 1.
Each process Pi is organized as follows:

```
do {
   wait(mutex);
   /* critical section */
   signal(mutex);
   /* remainder section */
} while (true)
```