Operating Systems

Homework 4

Question 1

Busy waiting is a technique in which a process is waiting for a condition to be met, such as the availability of a lock or an I/O device, by consistently checking over and over again. This keeps the CPU busy all the time that is is checking the condition.

Other alternatives to busy waiting include:

- Blocking - The process tells the operating system that it will be waiting for something, so the operating system suspends the process for a while until whatever the process was waiting for is available. Unlike busy waiting, blocking requires assistance from the operating system.
- Polling - The process periodically checks if the resource is available, and while it isn't, it will do other things so that it does not waste CPU time.

Busy waiting can be avoided, but it is sometimes preferable to use busy waiting for short times, since it is somewhat expensive to put a process to sleep and then resume it.

Question 2

In single processor systems, spinlocks are inappropriate because when a process acquires a lock, then the scheduler performs a context switch, all the other processes which are waiting for the lock will waste their quantum waiting for the lock.

However, in a multiprocessor system, spinlocks are more appropriate since when a process acquires a lock, and there are other processes running on the other processors, then while the other processes are spinning the process with the lock can continue working to release the lock, and then the other processes can continue. On such a system, it can avoid context switches which are expensive.

Question 3

If the wait and signal operations of a semaphore are not performed atomically, then if for example one process begins to wait, and the value in the semaphore is positive, then the first process can read the value into a register and subtract one, but before it manages to write the new value back in, another process can interrupt the first one. If the second process then tried to wait on the same semaphore, then it would read the old value, subtract one and write it back to memory. Then when the first process is resumed, it would write back the old value minus one, which is the same value the second process wrote. In such a case, the semaphore has allowed two people to take a resource, but only removed one from the list of available resources.

Question 4

1. An example of a real life deadlock (true story) is when my roommate wanted to become an Israeli citizen, and in order to complete the process, the government required that he present an Israeli bank account. But when he went to the bank to create an account, they required that he be an israeli citizen.
2. Another example is when two cars are heading towards each other on a narrow single lane road.

3. Another example of when a deadlock can occur, is if someone wants to buy a house, so he needs to borrow money from the bank, but the bank will only lend money to property owners.

Question 5

It is not possible for a deadlock to occur if there is a single process running, because a deadlock can only occur when the process is waiting for a resource that another process is holding, and that process can only release the resource once the first process releases its resources. So by definition, at least two processes are necessary for a deadlock to occur.

Question 6

A binary semaphore can be initialised to 1, symbolising that permission to run is available. When a process calls the wait function, it changes the value to 0, symbolising that permission to run is not available. Then when signal is called, it changes it back to 1, allowing other processes to enter a critical section.