**Operating System Principle (DKT 221)**

**Tutorial 6 (Synchronization)**

1.Race conditions are possible in many computer systems.

Consider a banking system with the following two functions: deposit (amount) and withdraw (amount). These two functions are passed the amount that is to be deposited or withdrawn from a bank account. Assume a shared bank account exists between a husband and wife and concurrently the husband calls the withdraw () function and the wife calls deposit().

Describe how a race condition is possible and what might be done to prevent the race condition from occurring.

**Answer:** Assume the balance in the account is 250.00 and the husband calls withdraw (50) and the wife calls deposit (100). Obviously the correct value should be 300.00 Since these two transactions will be serialized, the local value of balance for the husband becomes 200.00, but before he can commit the transaction, the deposit (100) operation takes place and updates the shared value of balance to 300.00 We then switch back to the husband and the value of the shared balance is set to 200.00 - obviously an incorrect value.

2.a. Define the term *busy waiting.*

b. List other kinds of waiting in an operating system.

c. Can busy waiting be avoided, explain your answer.

**Answer:**

a. A process is waiting for a condition to be satisfied in a tight loop without relieving the processor.

b. A process could wait by relieving the processor, and block on a condition and wait to be awakened at some appropriate time in the future.

c. Busy waiting can be avoided but cause the overhead by putting a process to sleep and having to wake it up when the appropriate program state is reached.

3.Describe why disabling interrupts in synchronization is not appropriate in a single-processor system if the synchronization is used in user-level programs.

**Answer:**

If a user-level program is given the ability to disable interrupts, then it **can disable the timer interrupt** and **prevent context switching from taking place**, thereby **allowing it to use the processor without letting other processes execute**.

4.Describe why interrupts are not advisable to be use in synchronization in multiprocessor systems.

**Answer:**

Interrupts are not sufficient in multiprocessor systems since disabling interrupts only prevents other processes from executing on the processor in which interrupts were disabled;

there are no limitations on what processes could be executing on other processors and therefore the process disabling interrupts **cannot guarantee mutually exclusive access** to program state.

5.a. Discuss the role of fairness and throughput of operations in the readers-writers problem.

b. Propose a method for solving the readers writers problem without causing starvation.

**Answer:**

a. Throughput in the readers-writers problem is increased by favoring multiple readers as opposed to allowing a single writer to exclusively access the shared values;

favoring readers could result in starvation for writers.

b. The starvation in the readers writers problem could be avoided by keeping timestamps associated with waiting processes. When a writer is finished with its task, it would wake up the process that has been waiting for the longest duration.

When a reader arrives and notices that another reader is accessing the database, then it would enter the critical section only if there are no waiting writers. These restrictions would guarantee fairness.

6.Describe the reason why *spinlocks* is used as a synchronization mechanism mostly only on multiprocessor systems and not on single processor systems.

**Answer:** Solaris, Linux, and Windows 2000 use spinlocks as a synchronization mechanism only on multiprocessor systems. Spinlocks are not appropriate for single-processor systems because the condition that would break a process out of the spinlock could be obtained only by executing a different process. If the process is not relinquishing the processor, other processes do not get the opportunity to set the program

7. Explain the effect of disabling interrupt to the operating system.

The system clock is updated at every clock interrupt. If interrupts were disabled—particularly for a long period of time—it is possible the system clock could easily lose the correct time. The system clock is also used for scheduling purposes. For example, the time quantum for a process

is expressed as a number of clock ticks. At every clock interrupt, the scheduler determines if the time quantum for the currently running process has expired. If clock interrupts were disabled, the scheduler could not accurately assign time quantums. This effect can be minimized by disabling clock interrupts for only very short periods.

These operating systems provide different locking mechanisms depending on the application developers’ needs. Spinlocks are useful for multiprocessor systems where a thread can run in a busy-loop (for a short period of time) rather than incurring the overhead of being put in a sleep queue. Mutexes are useful for locking resources. Solaris 2 uses adaptive mutexes, meaning that the mutex is implemented with a spin lock on multiprocessor machines. Semaphores and condition variables are more appropriate tools for synchronization when a resource must be held for a long period of time, since spinning is inefficient for a long duration.

True / False Questions:

1. T / F – Distributed processing can be defined as the management of multiple processes executing on multiple, distributed computer systems.

ANS: T

1. T / F – Concurrency issues are a concern on multiprocessor systems, but do not impact uniprocessor systems.

ANS: F (impact both types of systems)

1. T / F – Starvation refers to the situation where competing processes are denied access to a resource due to scheduling problems.

ANS: T

1. T / F – The major difficulty with semaphores is that wait and signal operations may be scattered throughout a program and it is difficult to see the overall effect of these operations on the semaphores they affect.

ANS: T

1. T / F – Message passing provides both synchronization and communication, which are fundamental requirements for interacting processes.

ANS: T

1. T / F – In a message passing system, one queuing discipline alternative is to allow the receiver to inspect the message queue and select which message to receive next.

ANS: T

Multiple Choice Questions:

1. Concurrency plays a major part in which of the following specific contexts:
   1. Multiple applications
   2. Structured applications
   3. O/S structure
   4. All of the above

ANS: D

1. Examples of solutions to the concurrency problem that do not involve busy waiting are the following:
   1. Semaphores and monitors
   2. Message passing and caching
   3. Producers and consumers
   4. None of the above

ANS: D (all software solutions involve some form of busy waiting)

1. The following requirement must be met by any facility or capability that is to provide support for mutual exclusion:
   1. Only one process at a time can be allowed into a critical code section
   2. A process remains in its critical code section for a finite time only
   3. No assumptions can be made about relative process speeds
   4. All of the above

ANS: D

1. In a uniprocessor system, mutual exclusion can be guaranteed by:
   1. Overlapping processes
   2. Interleaving processes
   3. Disabling interrupts
   4. All of the above

ANS: C

1. A semaphore that does not specify the order in which processes are removed from the queue is called a:
   1. Weak semaphore
   2. Strong semaphore
   3. Binary semaphore
   4. None of the above

ANS: A