**OPERATING SYSTEMS LAB 03**

1. What is a critical section? What is the critical-section problem?
   1. Design an algorithm that can only have one process at a time within the critical selection with 3 requirements.
   2. Critical section is code in which share data is accessed
2. What three requirements must a solution to the critical-section problem satisfy? Explain the meaning of each requirement
   1. Mutual Exclusion: No processes can simultaneously be inside critical sections
   2. Progress: no process running outside its critical section may block other processes from entering their critical sections.
   3. Bounded Waiting: no process should have to wait forever to enter critical section
3. What is an *atomic* instruction? Show that if the *wait* operation is not executed atomically, then mutual exclusion may be violated
   1. An instruction that must be fully completed without interruption.
   2. If wait is not done atomically another wait process call can enter the critical section at the same time.
4. The first known correct software solution to the critical-section problem for two processes was developed by Dekker. The two processes, P0 and P1, share the following variables:   
   Prove that the algorithm satisfies all three requirements for the critical section problem.
   1. Mutual Exclusion: Checks if the other process wants to enter its critical section.
   2. Progress: Once finished changes so other process can enter its critical section
   3. Bounded Waiting: Because of the turn variable, next process will run once current one has finished.
5. What is spinlock? Discuss the advantage and disadvantage of using spinlock. Why do you think Solaris, Linux, and Windows 2000 use spinlocks as a synchronization mechanism only on multiprocessor systems and not on single- processor systems?
   1. Spinlock is an infinite block loop while waiting for the lock to be available.
   2. Advantage: Easy to implement, faster than context switch
   3. Disadvantage: Time consuming
   4. Only locks one processor of the system rather than the whole system if it only has one processor. Processes can both run at same time and if one finishes the other can quickly switch back and break out of while loop.
6. Suppose we want to execute statements S1, S2, and S3 in sequence, but that S2 has to be executed exclusively for one process at a time. Write the code needed using semaphores
   1. S=1 // remember to initialise semaphore  
      S1wait(s)  
      S2Signal(S)  
      S3
7. Explain when and how to use each of the following functions:
   1. Pthread\_Mutex\_Lock()
      1. Get mutex lock. Must initialise mutex variable.
   2. Pthread\_Mutex\_Unlock()
      1. Release mutex lock.
   3. Pthread\_Cond\_Wait()
      1. Used to wait for a conditional variable
   4. Pthread\_Cond\_Signal()
      1. Used to send signal to one thread waiting for a condition variable.
8. *Sleeping Barber* problem. Consider a barbershop with one barber, one barber chair, and *n* chairs for waiting customers, if any, to sit in. If there are no customers present, the barber sits down in the barber chair and falls asleep. When a customer arrives, he has to wake up the sleeping barber. If additional customers arrive while the barber is cutting a customer’s hair, they either sit down (if there are empty chairs) or leave the shop (if all chairs are full). The problem is to program the barber and the customers without getting into race conditions.