Parallel Computer Systems

Exercise 4.a Dean Shalev

209707470

1. Lock 1 – What works: Mutual exclusion (only one thread can be in the critical section at a time.
   What doesn't work: Deadlock – An infinite loop will occur if both threads set their flags simultaneously.
   Fairness – A thread can potentially delay another thread indefinitely.
   Lock 2 – What works: Mutual exclusion (only one thread can be in the critical section because the victim switches).
   What doesn't work: Deadlock freedom – no deadlock because one thread always gives way.
   Fairness: Unfair because the victim variable can alternate inefficiently, potentially starving one of the threads.
2. **Does the lock work in all cases?**
   Yes, Peterson's lock ensures mutual exclusion, no interleaving allows both threads to enter the critical section at the same time.
   Can both threads enter critical section at the same time?
   No they cannot
   Why can there be no deadlock?
   Deadlock is avoided because the while condition is dependent only on the other threads flag.
   Show and explain why the lock is fair
   The algorithm doesn't bias any particular thread and each thread get equal opportunity to enter the critical section.
3. **Mutual exclusion**
   The flag and turn variables ensure mutual exclusion because one thread will wait for the other to clear its flag.
   Deadlock
   Deadlock doesn't happen because each thread progresses based on turn
   Starvation
   No thread is delayed perpetually because of the turn variable
4. **Filter Algorithm**
   Requires extensive memory and computational overhead because each thread must monitor the others' levels. Complex to implement and debug for large number of threads.
   Bakery Algorithm
   Uses ticket-based ordering which can lead to high contention for shared memory. Suffers performance bottlenecks because of frequent memory access.