Operating System HW 3

Ali Siddiqi 26902

Ans1) In xv6, `push\_off` and `pop\_off` manage interrupt states for nested locking, ensuring interrupts are only re-enabled when the last lock is released. While `intr\_off` and `intr\_on` directly control interrupt states, they don't handle nested critical sections. Using `push\_off` increments a counter each time a lock is acquired, and `pop\_off` decrements it upon release, re-enabling interrupts only when all locks are released. This prevents deadlocks by ensuring an interrupt handler doesn’t attempt to acquire a lock already held by the interrupted code. The strict order of `push\_off` before acquiring a lock and `pop\_off` after releasing it eliminates race conditions that could otherwise lead to deadlock.

Ans2) Locks and semaphores both synchronize concurrent access to resources, but they differ in flexibility and usage. Locks are binary, allowing only one thread at a time to access a critical section, ensuring mutual exclusion. They are typically used for simple resource protection and are susceptible to deadlocks if not carefully managed. Semaphores, on the other hand, use a counter to allow multiple threads up to a specified limit, making them ideal for controlling access to a limited pool of resources (e.g., database connections) or coordinating complex thread interactions, like in producer-consumer scenarios. Unlike locks, semaphores aren’t owned by a single thread and offer more flexibility for signaling and inter-thread communication.

Ans3)

Ans4)