Project 1 Writeup

Team 10
Yuan-Pu Hsu 862057597
TianXiang Sun 862051319

1. First, the process in userspace request the system call. (If there's a corresponding system call wrap in C library, the library would create another task to perform the system call.) It will also trigger a interrupt or trap, OS switches the context and change the mode from user mode to kernel mode. After entering the kernel mode, it executes the system call. After the system call finished, the OS store the result of system call into the register for that userspace process, and switches back to user mode and load the context, then beginning to execute process in userspace again.

2. When there is a race condition, for instance, multiple processes are trying to access certain variable (or register or memory address), and change the value inside, the data then need to be synchronized. Or the result of the data would be unpredictable after the execution.

3. Synchronization mechanisms
    a. Atomic operation: Ensure that an operation is atomic at the hardware level.
    b. spinlock: locking mechanism for multiprocessor systems.
    c. Mutexes: Kernel's main locking primitive, task accessing a lock suspends when the lock is already held.
    d. Semaphores: providing mutual exclusion for multiple resource.
    e. RCU (Read-Copy Update) : Useful for read-mostly data structures

4. No, spinlock is not appropriate for single processor. For spinlock, it is a lock which causes a thread trying to acquire it to simply wait in loop while repeatedly checking if the lock is available. And for single processor, just one process can be executed during a period of time. If one thread is executing and finding that it cannot acquire the lock. It can just wait the lock to be released. However, the thread is not suspended. Thread which get the lock cannot be executed. It might be so difficult to schedule.Therefore, there would be a hugh cost for using spinlock in single processor system.