Edward Basulto

5159172

Synchronized Threads:

Synchronizing threads on the C file allows for a more organized sequence in which the threads wait until it is their turn because of the lock and unlock by the mutex. Each thread affects the shared variable at its own turn. When compiling the synched threads will always return the same value depending on the amount of threads affecting the shared variable.

Unsynchronized Threads:

Unsynchronized threads on the C file affect the shared variable at their own time. It is random. If the threads try to affect the shared variable at the same time, the shared variable will only increment by 1. Unsynchronized threads will always return a random value. It is unknown the order in which the threads will affect the variable and if they will attempt to change the variable at the same time.

In a multi-threading environment, the risk of data being lost due to threads attempting to alter a variable concurrently. Synchronization of the threads can prevent this loss or alteration of data from happening. We use project one to visually see how unreliable threads can be if they are not synchronized causing loss of data and different results in each run. Synchronizing these threads provide the same result without fail when testing the code each time.