**Second readers-writers problem Wiki**

// Mohamad Salaam

// COMP3500-002

// Second Reader-Writer problem using pthreads

int readcount, writecount; *//(initial value = 0)*

semaphore rmutex, wmutex, readTry, resource; *//(initial value = 1)*

*//READER*

reader() {

<ENTRY Section>

readTry.P(); *//Indicate a reader is trying to enter*

rmutex.P(); *//lock entry section to avoid race condition with other readers*

readcount++; *//report yourself as a reader*

**if** (readcount == 1) *//checks if you are first reader*

resource.P(); *//if you are first reader, lock the resource*

rmutex.V(); *//release entry section for other readers*

readTry.V(); *//indicate you are done trying to access the resource*

<CRITICAL Section>

*//reading is performed*

<EXIT Section>

rmutex.P(); *//reserve exit section - avoids race condition with readers*

readcount--; *//indicate you're leaving*

**if** (readcount == 0) *//checks if you are last reader leaving*

resource.V(); *//if last, you must release the locked resource*

rmutex.V(); *//release exit section for other readers*

}

*//WRITER*

writer() {

<ENTRY Section>

wmutex.P(); *//reserve entry section for writers - avoids race conditions*

writecount++; *//report yourself as a writer entering*

**if** (writecount == 1) *//checks if you're first writer*

readTry.P(); *//if you're first, then you must lock the readers out. Prevent them from trying to enter CS*

wmutex.V(); *//release entry section*

resource.P(); *//reserve the resource for yourself - prevents other writers from simultaneously editing the shared resource*

<CRITICAL Section>

*//writing is performed*

resource.V(); *//release file*

<EXIT Section>

wmutex.P(); *//reserve exit section*

writecount--; *//indicate you're leaving*

**if** (writecount == 0) *//checks if you're the last writer*

readTry.V(); *//if you're last writer, you must unlock the readers. Allows them to try enter CS for reading*

wmutex.V(); *//release exit section*

}