Operating System Assignment discussion

Rares Popa - 19159700

**Pthreads**

The first part of this assignment was to implement the readers – writers problem using pthreads, mutual exclusion was achieved using 4 functions. Pthread\_mutex\_lock(), Pthread\_mutex\_unlock(), Pthread\_cond\_wait() and Pthread\_cond\_signal().

All the readers and writers accesses the shared resources

The way which I used these 4 functions was firstly I called the Pthread\_mutex\_lock() and passed the mutex through, what this did is that it makes sure no other readers/writers can enter the critical section while another reader is there, then I could increment the reader\_count. Later after the critical section is done I called Pthread\_mutex\_unlock() and passed the mutex again, unlocking the critical section for another reader to enter. Doing this prevents race conditions also (When 2 values try access the same shared memory location). I did all this again when decrementing the reader\_count. I did the same thing in the writer function.

I also used Pthread\_cond\_wait() and passed the mutex and a condition if there was more than one reader or more than 0 writers already reading/writing. This puts the reader to sleep until later where we call Pthread\_cond\_broadcast() and pass the mutex which wakes up all the readers/writers.

I decided with my pthread program to only allow one reader read the data\_buffer at a time, the assignment specifications stated that “… multiple readers can read the buffer at the same time” no specifying that it is a requirement (a must). I decided to do this because it makes my code much easier to decode, read and understand.

I have found no problems within my pthread program, it is doing all the required tasks and giving the correct output, I have linked an example of the output in the pthread\_eg.txt file, I tested this via the shared\_data.txt file, kept running the program to stress test it and it worked fine every test.

**Processes**

The second part of this assignment was to implement readers – writers problem using processes, mutual exclusion was achieved using 2 functions, wait() and signal() and 3 mutexes, mutex, rf\_mutex and w\_mutex.

All the readers and writers accesses the shared resources

Firstly I needed to initialize the mutexes using the sem\_init() function. Then in the reader function I called wait() and passed the mutex through, this made sure no other reader process can access their critical section until this process is done. In the critical section I would increment the reader\_count, once done I would then call signal() and passed the mutex, which released the mutex allowing another reader to get it and go into their critical section.

After the reader read\_file() function was called I then called the wait() function, passing the mutex and then I decremented the reader count, afterwards I called the function signal\_next() which signalled the next waiting reader/writer then I called signal() again passing the mutex which released the mutex. I did the same thing in the writer function.

My process program does not run well, it has multiple bugs which I cannot fix. Sometimes the process will break and then just loop forever until it is terminated. At other times it will work fine but give the incorrect readerPieces data. I have tried many different aspects on fixing this problem but I think it runs much deeper. I have included an example of the outputs in many files, process\_eg\_loop.txt, process\_eg\_pieces.txt which presents the several types of errors. I believe it might be a problem with the semaphores and the allRead value. I tested this program the same way and the pthread program via the shared\_data.txt file, kept running the program to stress test it. Mutual exclusion does work.