Ryan Yurkanin

Intro to Systems

Lab 3 Linux Documentation

HOW TO RUN

In order to run use this command to compile through GCC: gcc main.c –lpthread –o main

BRIEF OVERVIEW OF PROBLEMS

Problem: Create a buffer that is accessed by multiple consumer and producer threads. The producer threads will create unique content to put into the buffer and the consumers will take the unique content out. Only one should be accessing the critical point at a time.

FUNCTIONS USED

Sem_init – semaphore.h function that allows me to create a semaphore. I used this to create both full and empty and these two semaphores are used to prevent consumer threads from consuming from an empty buffer and producers to produce to a full buffer.

Sem_wait – semaphore.h function that I use to have the producer/consumer threads to wait for full/empty status of the buffer.

Pthread_join – pthread.h function that I use to make main wait for all of the other threads to finish before closing the main process. This way the program actually functions for more than a microsecond before closing. Also cleans up all the zombie threads.

Sem_post – semaphore.h function that I use to let all the other threads know that an object has either been taken or added to the buffer. I also use it to update mutual exclusion for every other thread..

Pthread_create – pthread.h function that I use to actually make all of the producers and consumers. I then tell them to execute either producer() or consumer().

usleep - function that I use to make the threads wait a bit before running after their creation.

Printf – function that I use to print lines to the screen. I used this a lot not just while testing, but during the actual running of the program.

Fprintf – function that I use to print lines to the log. That way you can actually take some time and observe what is happening during the course of the program.

Fopen – function that I use to open the log file that all of the fprintf's are being printed to.

close() – I use this function to close file streams that no longer need to be open for the current time.

Srand – function I use to set the seed for rand. I use a random variables address for this.

Rand – Function I use to make names for each thread and for the items put into the buffer. That way they can be distinguished from other threads.

TESTING LOG, BUG NOTES, AND PROOF OF SOLUTIONS

Step 1: In the beginning I just had one producer and one consumer. I had the producer make a single item and put it in the buffer and then the consumer would take it. This only occurred one time and I used print statements to track exactly when things were happening. This went smoothly and made a good base for the future.

Step 2: After that I expanded the buffer and allowed for the producer and the consumer to run for more than one time. The signaling was a little wonky but then I realized I had both of them looking at the wrong semaphores. Once I switched them around it began to work properly.

Step 3: Finally I added multiple producers and consumers and began to test from there. I ran into a strange bug where all of the items in the buffer would go through the same pattern of numbers for names per thread. I solved it by having srand generate a random seed in each thread instead of right in main. Then I also made it use the address of a random variable to seed instead of time(NULL)

Improvements: Going forward I wish I would of added a way to visually see what was in the buffer. That way it would be a lot easier then parsing the log to see what is actually going on. Also, the text isn't always clear when it comes to telling the user what step is actually happening.