CSE 4300 Programming Assignment 2
Assigned: Feb 4, 2025
Due: Feb 12, 2025, 11:59 pm (Midnight)

The purpose of this assignment is to introduce you to the following concepts:

1. Writing Multithreaded programs and Use of synchronization primitives, which you will do on Linux.
2. Modifying and compiling OS161 Kernel, which you will require you to modify OS161 kernel code.

Suggested Reading for this assignment (Please note that you can find many other great tutorials by simply searching on Google!).

https://hpc-tutorials.llnl.gov/posix/

(Please read this! You can also find sample code here!)

You may find the followings useful for part of this project –

- pthread_mutex_t *mutuex;
- pthread_cond_t *condition;
- pthread_create ();

Please note that, to compile multithreaded program, you need to specify "-lpthread" as follows

gcc -o output.txt program.c -lpthread

Part 1: (50 points)

In this problem, you are going to write a multithreaded program using pthread.

You are going to create a number of threads and label some of them as "User level thread" and some as "Kernel level thread".

- **Note that this is just a simulation and all threads are created the same way.**

Let us assume that there are **N** user level threads that need to send information over the network. Each thread is labeled from 1 to N, and they must send data in that order. For example, Thread 3 cannot send data until Thread 1 is done sending data.

- **Note that you do not have to send data over the network. You are simulating a producer-consumer scenario in this assignment.**

To send data, a thread first needs to copy the data to a designated 1-item buffer (i.e., packet), which is then read by a kernel level thread. Let us assume that there are **X** number of packets available, and **M** number of kernel level threads available that are dedicated to read these packets and send the data over the network.

Once a user level thread copies data to a packet (i.e., copy in the shared buffer), the number of available packets decreases by one. If there are no empty packet, the user thread must go to sleep. Also, a thread has to go to sleep if it is not its turn to send data yet, even if there are empty packets. Once the data is written to the packet, the user thread terminates.

Once the kernel thread read data from the packet, the number of available packets is increased by one, and kernel thread will wake up user threads who might be sleeping as there were no packets available. A kernel thread will go to sleep if all packets are empty and there is no data to send. A kernel level thread terminates once N user level threads are served.

Think carefully how you can terminate the program.

Each time a user level thread needs to sleep due to unavailable buffer or not being its turn, it needs to print
printf("No packet available or not my turn to produce, user level thread %d going to sleep\n", my_id);

Each time a user thread gets to put data in a buffer, it should print -
printf("User level thread %d is going to put data in a packet\n", my_id);

Each time a kernel level thread goes to sleep, it needs to print:
printf("No data available, Going to sleep kernel thread %d\n", my_id);

Each time kernel thread serves a user thread, it should print -
printf("user thread %d getting served\n", servedId);

Note

1. Note that you are simulating user level threads (you can think of them as producer threads) and kernel level threads (you can think of them as consumer threads).
2. You should not use busy waiting, and should use condition variable.
3. To add randomness within each thread, please use something similar to this to add random delay: usleep(10000 + 10000*(my_id*1000 % 5));
4. **Your program should take X, N, M as input from the command line, in this order.**

Sample Output:

>./data_to_kernel 2 6 2

User level thread 1 is going to put data in a packet
User level thread 2 is going to put data in a packet
No packet available or not my turn to produce, user level thread 3 going to sleep
No packet available or not my turn to produce, user level thread 4 going to sleep
No packet available or not my turn to produce, user level thread 6 going to sleep
No packet available or not my turn to produce, user level thread 5 going to sleep
user thread 1 getting served
user thread 2 getting served
User level thread 3 is going to put data in a packet
No packet available or not my turn to produce, user level thread 5 going to sleep
No packet available or not my turn to produce, user level thread 6 going to sleep
User level thread 4 is going to put data in a packet
user thread 3 getting served
User level thread 5 is going to put data in a packet
No packet available or not my turn to produce, user level thread 6 going to sleep
user thread 4 getting served
User level thread 6 is going to put data in a packet
user thread 5 getting served
user thread 6 getting served

What to submit

Please submit a well-documented listing of your source code and output through HuslyCT. You can copy paste your code in a word file if you want.

Part 2. (50 points)

In this part, you are going to implement the producer-consumer problem using pthreads.
You are going to create 3 producer threads and 2 consumer threads in the program.
After implementing, let the program run until at least 1000 items are produced by the 3 producers combined, and at least 1000 items are consumed by the 2 consumers combined.

Repeat the experiment for the following settings:

1. Queue size 10
2. Queue size 20
3. Queue size 100

For each setting, count the number of times each producer and consumer goes to sleep because of empty queue or full queue. **Print the total number of times each goes to sleep at the end of the experiment. Include this in your report.**

Note that the threads should go to sleep instead of doing busy waiting for the appropriate conditions.

What to submit

Please submit a well-documented listing of your source code and output through HuslyCT. You can copy paste your code in word file if you want.

Part 3: 25 points

This part is to get yourself familiarized with the OS/161 source code. Your job is to identify and revise the proper module(s) to customize your OS/161 greeting message that is displayed on the menu screen.
Print your name during the boot process.

Hints:
1. Look inside os161-1.11/kern/main directory
2. You will need to compile kernel and execute ASST0. You should already know how to compile kernel (assuming you have already installed OS/161).

What to submit:
- Please include a screenshot of the boot screen with your message and include in the report.
- Please specify the name of the file that you changed.