Lab 5

Deeksha Juneja

2/24/2016

Introduction

In this lab, we have to learn about what happens when we want to share variables between two threads. We look at problems that sharing variables causes and how we can solve it. We specifically at mutex_lock, conditional variables and semaphores. We also work on a print server.

Part 2

The thread counter:

Q1. What is the expected output?

2000000

Q2. What is the calculated output?

1029152

Q3. What caused the discrepancy between the expected and calculated values?

The discrepancy is due to the shared variable and it wasn't locked by one thread at a time. Both threads were trying to write to it. Hence, we need to use mutex to avoid it.

Part 2.1

Q4. Did this fix the issue with the original code?

Yes, we got 2000000 as the answer by putting mutex lock.

Part 3.1

Q5. What is the minimum number of conditions needed for the example to work as intended?

Two conditional variables

Q6. What would those conditions be, and which thread (producer or consumer) should wait on the condition?

The consumer should not try to consume when producer is writing to the queue or when the queue is empty. Also the producer should not write to the queue when the queue is full or the consumer is reading from the queue.

The output of cond_example.c

```
deeksha@co2046-03 lab-05]$ gcc cond_example.c -pthread -o cond_example
deeksha@co2046-03 lab-05]$ ./cond_example
ever going to give you up
ever going to let you down
ever going to run around and desert you
deeksha@co2046-03 lab-05]$
```

Part 3.2

Run the program, and note the order in which the buffer is read/written to. Run the program multiple times, and again not in the order in which the buffer is read/written to. Do they look different? Why do you think this is the case?

I think this happens because the producer and consumer are not working at the same pace. What that means is that, if a producer produces twice, then the consumer might only consume once before the producer produces again. Or, the cosumer may consume twice before the producer produces again. Hence, it is not syncronised. If their actions were syncronised, then this wouldn't happen.

Part 4

There were 3 warning tags. The first one was in the consumer thread, the second one was in the producer thread and the third one was about joining all the threads. The first thing I did was the joining part. I did so because, I already knew how to do that part because of lab 4. I do it in a nested for loop for each printer group and each printer in the group.

Next, I worked on my consumer and producer threads. In the printer_thread I am doing a wait jobs, and then I am putting a lock in. Inside the lock I am just retreiveing the first job. Then outside the lock I am sending it to the printer and this can be done easily outside the lock because each printer can print independent of each other. Due to the way I am retreiving jobs from the head, I do not require prev. Hence, I have commented it out.

In my producer, I am doing something similar to the consumer except I am adding jobs to the queue and I am following it with sem_post.

We have to also make a logfile. For this I made a separate function to open the file. I have added a case in parse_command_line for "l" which would write to the log_file. For system time, I am using time_t. I am doing this in both producer and consumer thread to record the time.

I was getting some problems with my exit flag. Hence, I added an empty job that would go into the consumer thread and then if was file name for that was null, I would simply not do the print and exit. Moreover, to further handle it, I changed the condition of the while loop to exit_flag!=1 instead of having it 1.

Doing the exit part was really hard because I didn't know what the problem actually was. My code was not showing any error and it still wasn't working. My TA Kris really helped me a lot with this lab. I also didn't know how the time stamping works, so I had to spend sometime going over that as well. It also took me a long time to understand how the whole code worked together because there were so many struct in there.

Conclusion

This lab was pretty hard for me because there were a lot of things involved. Moreover, it took me a while to properly understand producer and consumers. But it was also a fun expereince. In the end I was successfully able to make output.txt file and change .ps file to .pdf file.