Lab 5 Lab Report: CprE 308

Matt McKillip

1. Introduction

This lab is introducing the concept of sharing variables between threads. In our previous lab, the program had no need to share variables, so no locking needed to be done. In this lab we will be doing examples where we will need to lock the critical section using mutex. These examples will apply what we have learned in lecture about mutexes. Another way to use shared variables is to create condition variables, which will help in producer and consumer programs.

2. Questions

2.1 std_dev_calc.c

2.1.1What is one expected output?

sum = 6846.00

avg = 6.846

sum = 18903.314

std = 4.348

2.1.2 What is the calculated output?

sum = 6846.00

avg = 6.846

sum = 15799.27

std = 3.9748

```
e_sum: 6846.000000      sum: 6846.000000
e_avg: 6.846000         avg: 6.846000
e_sum: 18906.314453     sum: 15799.273438
e_std: 4.348139 std: 3.974830
[mamckill@co2048-12 Lab5]$
```

2.1.3 What caused the discrepancy between the expected and calculated values?

The values were different due to the shared variables not locking. Since no mutex is being used.

2.2 std_dev_calc.c with mutex added

2.2.1 Did this fix the issue with the original code?

Yes, now each time the program runs, the expected and calculated are within .01 for sum and .00001 for the std.

```
e_sum: 6846.000000      sum: 6846.000000
e_avg: 6.846000         avg: 6.846000
e_sum: 18906.314453     sum: 18906.306641
e_std: 4.348139 std: 4.348138
[mamckill@co2048-12 Lab5]$
```

2.2.2 Does it match with the expected output of the program?

Yes, that is the order that I expected.

2.3 Condition Variables

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

2 condition variables

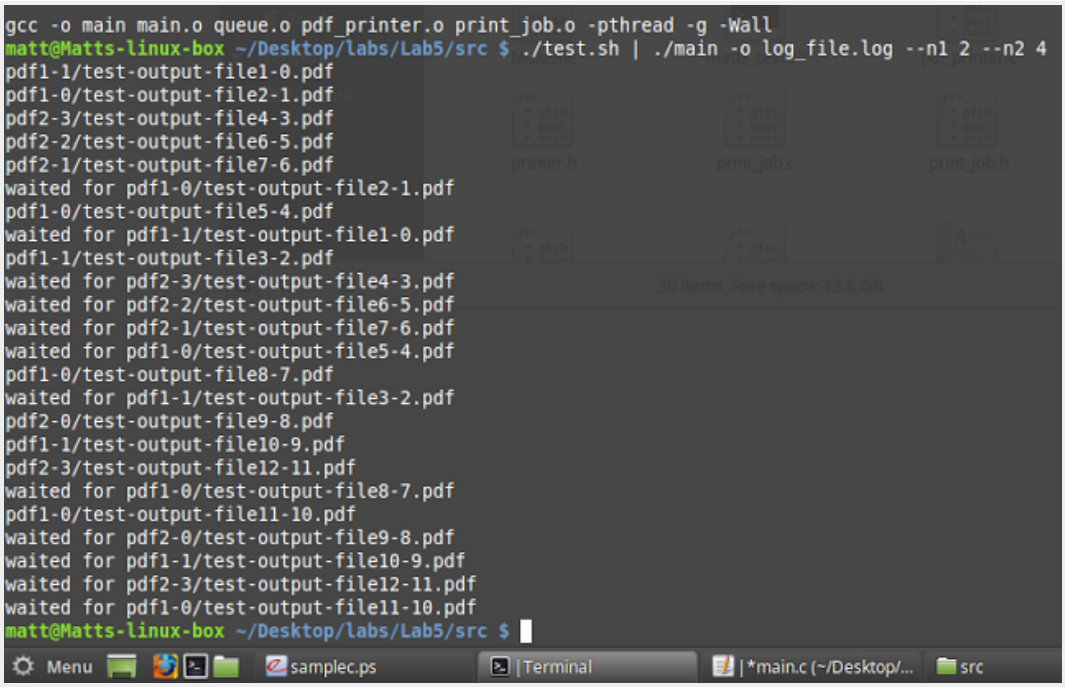
2.3.2 What would those conditions be, and which thread (producer or consumer) should wait on that condition?

The std and avg threads would be waiting on the condition variables set by the producer thread.

3. Results/Output

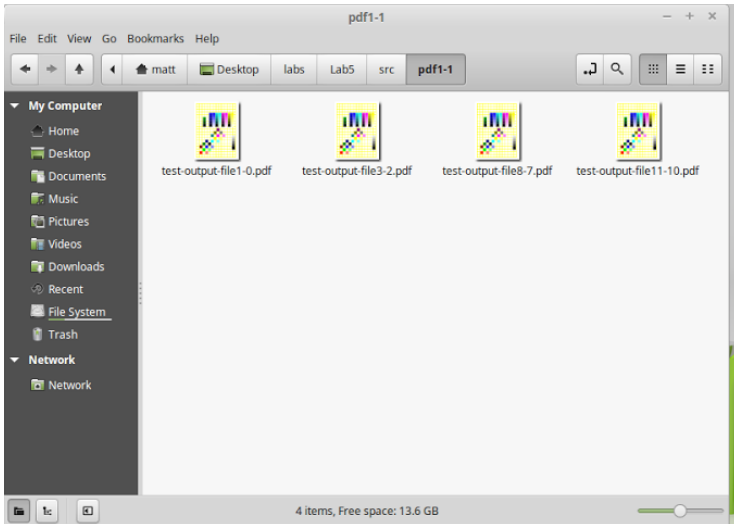
I have had a fair amount of difficulties during this lab from unknown errors to time constraints. I was able to get the pdf printer to output a pdf in the correct folders.

To test I used the provided test file - test.sh



The output to the command line shows the threads being sent to the printer, then completed.

The folders look like the following



Which includes the correctly named pdfs.

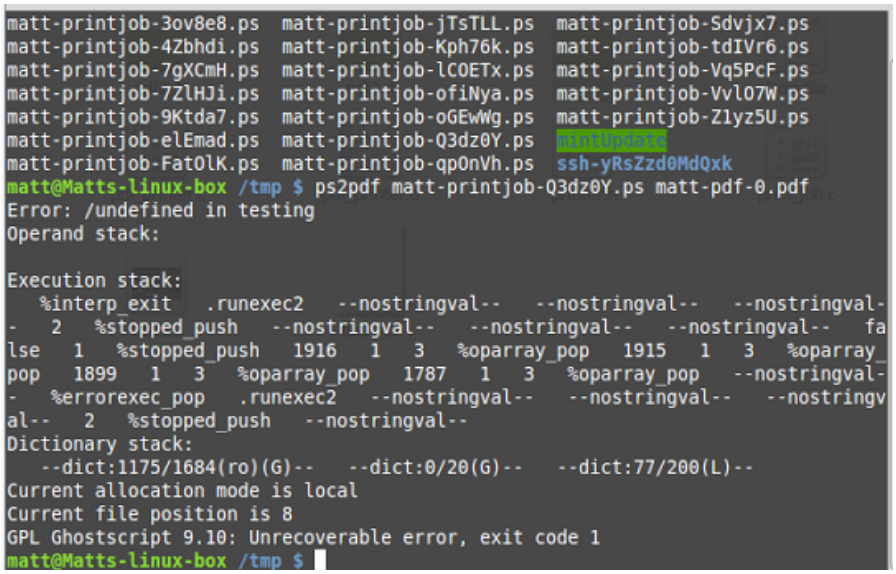
4. Design Decision

I decided not to change much of the code outside of the sections marked TODO. I decided this was the best approach since it was easiest and least error prone for integrating during lab 5 and future labs.

5. Issues

I had issues with segment faults early on in the lab when reading stdin. I ended up solving this with including more defensive code. Another issue I had was malloc and pointer problems. This is due to my inexperience with c programming.

My most problematic issue was a weird error when using ps2pdf from pdf_printer.c and from the command line. I cannot figure out if this is an error from my code or my linux box system. It seemed to happen every once in a while. So I am not positive how I can work around it.



6. Conclusion

I thought this lab was difficult and time consuming. I was unable to put in time to this lab due to midterms so I had trouble fixing all of the small bugs and errors from my code, but I was able to get this lab working correctly eventually.

7. Suggestions

I enjoyed how the lab tried to show us the topics in a real application, but I would have liked to have done this lab at a different time. During midterms it was tough to put in all of the required work to finish the lab completely.