CIS 415 Operating Systems

Assignment 3 Report Collection

Submitted to:

Prof. Allen Malony

Author:

Ryan Gurnick

Report

Introduction

This project is a publisher and subscriber system that works on various threads. With this project, there are multiple parts, the first of which is the basis of structures for queuing up topic entries into many buffers. Then we build on support for thread pools for subscribers and publishers. Additionally, in this project, we add in support for brokers that parse through the text files and populate information into the system via the main.txt file and command files for each publisher/subscriber. When we have that information we then go and attempt to add content to the subscriber HTML file.

Background

So I attempted to follow the quite convoluted instructions. That was the first step, then after going through the provided material, mainly the pub-sub given code. And modeled the rest of the program around that. There were some changes to the way that I did mutex locks where there are two, a primary and secondary mutex that help prevent deadlocking. The only portion of this project that has some issues is the HTML portion and there is a possible single memory leak that I could not narrow down.

Implementation

My implementation is nothing super amazing. I was provided the structs from you all, then developed a buffer as follows that it can all be stored in. I added two mutex's to it for better handling, however this made unlocking and locking far more complex to orchestrate.

```
typedef struct
{
    int entryNum;                           // entry number
    struct timeval timeStamp;               // created time
    pthread_t pubID;                        // publisher id
    char photoURL[URLSIZE];                 // url of topic
    char photoCaption[CAPSIZE];             // photo caption
} topicEntry;

typedef struct
{
    int head, tail, length, counter, topicID; // head, tail, length, counter &
    // topic for various purposes
    topicEntry *buffer;                      // buffer for storing topicEntry
    char name[MAXNAME];                     // name of the queue
    pthread_mutex_t primaryMutex;            // the main mutex
    pthread_mutex_t secondaryMutex;          // secondary mutex
} Buffer;

typedef struct
{
    pthread_t ID;                            // the thread id
    int flag;                                // flag to change state
    char location[MAXNAME];                  // name of thread
} threadArgs;
```

Overall these structs really dictate the way that things are handled, and there is a variety of parsing and sscanf *used to read files into the main(), subscriber(), and publisher() functions.*

Performance Results and Discussion

After running Valgrind this project has a single memory leak and some memory errors. According to the instruction, there are some bugs and possibly unchecked edge cases however the basics are there. This project does not run as well as I want it too however it is a huge progression from where I started two weeks ago with 10000000+ errors due to infinitely running threads. I have redone this project two full times and this submission is the cumulation of the best parts of those.

Conclusion

To conclude, this project really did not help me learn much about threads so much as the painful process of formatting HTML using languages they are not designed to be written in like for instance C. Please never make another sole write printf statements for HTML as its some of the worst quality code I have ever seen. There is little about this project that seems to be reinforcing anything about operating systems like Jared and I discussed during the first week. He was worried that it would not have relevance to the class and I totally agree, this project seems to be the farthest thing from operating systems while still using pthreads. UPDATE THIS PROJECT.