Yoni Stern

Assignment 1 writeup

Design:

main: First I put main’s arguments into global variables. Next I created n threads and passed each thread its thread number (had to put the numbers on the heap because it's possible the int the pointer is pointing to would be out of scope by the time it is used).

Worker Thread: Each thread starts with some startup code that gets the thread number using the pointer that was passed in. Afterwards I put a barrier to make sure all the threads are caught up before they start the real work. After this the threads enter an infinite loop. The loop starts by having the thread wait for its turn to run. I keep track of the thread which is currently sending out a request using the global variable curr. A thread sends a request to the server once the thread number == curr. Because we can’t use busy waiting I used cond_wait if the thread is not ready to run yet. This puts the thread to sleep until it receives a signal which notifies it that a thread finished sending its request. When it receives a signal it checks to see if it can send its own request. After a request is sent that thread will send a broadcast (wake up every thread that is waiting for a signal (notifies waiting threads that a new thread can send a request)) (request is not sent if it is the last thread). After this the thread that sent the signal receives the information from the server (concurrency). At the end of each loop I added a barrier to ensure that every thread finished their work before starting again with thread 0;

Get File: Use file 1 and 2 as global variables and use another global variable (int file) to keep track of which file to use. Every time we use a file I do file= !file to get the other file.

Complete Specifications:

When the assignment doc said to send a signal to the next thread I had to come up with a way to send the signal and make sure the correct thread runs(talked about this in design). Another ambiguous specification was where the assignment doc stated that we must switch between the 2 files. I also explained how I solved this in design

Bugs / Problems:

I don't think I have any bugs, if I knew of any I would have fixed them.
Might count as a bug, I'm guessing if you created a million threads bad things would happen.

Testing:

I tested my code using print statements. Everytime a thread sent a request I would print out its number and that it was sending a request. This helped me make sure that first thread 0 ran then thread 1 up to thread n and then restarted from 0. Every time I used a file I would print out the name of the file to make sure I was switching between the 2 files. I also printed out when I sent a signal to make sure that was working correctly.