Lab 5: Web Server - Thread Pool

Name:_____

Due: Tuesday Apr 19 2pm

Gather the first two initials to get the in-class 1%. Four initials in total.

Tracing the Creation of Threads

To see the creation of pthreads in a process, we can use the `strace` tool to track the system calls used by the process. Try running `strace ls` to see the system calls involved with running this `ls` command.

The underlying Linux system call `clone` is used for creating both processes and threads. If we run our server with the following command, we can see the creation of a thread on the fly when there are client connections:

```
strace ./webserver 2>&1 | grep clone
```

where `2>&1` tells the shell to redirect standard error to standard output as `strace` writes to standard error and the "filter" `grep` reads from standard input. Given our lab 4 implementation, the server will create one new thread per connection.

(On Mac OS, `dtrace` is an equivalent but nowadays the default security feature prevents the use of `dtrace`.)

TODO: On a Linux system (sloop/lab machine, etc.), show the trace of the thread creations on the server side by sending requests from a client.

Instructor/SI initial _____

Parsing Options

Read the *Command-line Parameters* section from the webserver README. The `webserver.c` source file uses `getopt` to parse the command-line options.

TODO: Add the support for the `-t`, `-b` and `-s` options. Echo out all the option values.

Instructor/SI initial _____

Concurrent Data Structures

For our familiar data structures, such as lists and queues, their previous implementations may suffer from data races or race conditions when multiple threads attempt to use them in parallel. Chapter 29, *Lock-based Concurrent Data Structures*, shows how to use locks to make such data structures *thread safe*.

Read the two pages about *Concurrent Queues* (or the entire chapter if you prefer). Start in a new folder, copy down the implementation. Write a main program that starts multiple threads which will enqueue or dequeue values to such a queue.

TODO: Write a C program that accesses the concurrent queue in parallel with multiple threads.

`Instructor/SI initial _____`

A Fixed-size Thread Pool

When there are hundreds of connections, the overhead of thread creation and destruction are not trivial. More over, there's a limit on how many threads can run at the same time, depending on how many logical cores the CPU has. You can use `get_nprocs()` to see the actual number. If you start 10 threads at the same time, only a couple of them will be actually running and others are queued up anyway.

Read the *Part 1:Multi-threaded* section from the webserver README. In this lab, 1) we will not consider the scheduling policy, 2) we will use that concurrent queue as the "buffer".

Good news! Here I give you my thread pool implementation:

https://gitlab.engr.ship.edu/chuo/cmpe320-lab5-thread-pool-start

I didn't have an explicit "master thread". The "master thread" is just the main thread. So you will do something in `wserver.c` as the "master thread". The `worker` thread now runs in an infinite loop instead. If the buffer (queue) is empty, a worker will be waiting until the "master thread" signals on the condition variable. Once the buffer is filled with a socket connection `fd`, the worker will retrieve and process it.

TODO: Put the thread pool, the concurrent queue, and the webserver together. The modifications you need are a few lines in `webserver.c` and the Makefile (maybe a new header file too). Show a working server with thread-pool by tracing the calls to `clone`: only a few (e.g. 3, whatever the `-t` option says) threads are created at the start of the server; when there are new connections, no new threads are created.

`Instructor/SI initial _____`

1% Bonus: Respect the Buffer Size Limit

TODO: Make proper modifications (maybe in either `thread_pool.c` or `webserver.c`) so that when the buffer (queue) is "full" according to the `-b` option, the main thread will wait for the consumption of the connections before placing new sockets in the buffer. Design an experiment to show the pool works properly in terms of queue size limit. Put the project on our department gitlab and share it with me `chuo`.

`Instructor/SI initial _____`