CS-350 - Fundamentals of Computing Systems
Homework Assignment #4 - BUILD

Due on October 10, 2024 — Late deadline: October 11, 2024 EoD at 11:59 pm

Prof. Renato Mancuso

Renato Mancuso

BUILD Problem 1

To improve throughput and utilize system resources more efficiently, you'll now modify your server to process requests using multiple worker threads. The new server will still allow to specify a maximum queue size to limit its maximum workload.

Output File: `server_multi.c`

Overview. Once again by building on top of what you have implemented so far, it is now time to add more workers to the system. The idea is quite simple: instead of launching 1 worker thread, launch N worker threads that can run the same logic and can fetch requests from a single, shared queue. This way, if a thread, say T0, is busy processing a request and another request is pending, then T1 can start working on it.

Design. The design is relatively simple. You will reuse what you learned about `pthreads` in `hw1`. This time, instead of creating only one worker thread, you will create `N` of them. All of the workers will execute the same `worker_main` function, but might be passed slightly different parameters. For instance, the first thread you create might be passed a thread ID of 0, the second one a thread ID of 1 and so on.

The important bit is to pass to all of them the same pointer to the same shared queue. The process that is the parent of all the worker threads will still be the only one responsible for performing `recv(...)` operation on the connection socket. It will also be responsible for enqueuing (or rejecting!) new incoming requests, just like in `hw3`.

Number of Threads. Just like in `hw3`, the number of worker threads to activate will be passed as a command line parameter to your code. The new parameter will be `-w <workers>` where `<workers>` is a positive integer greater than 1 representing the number of worker threads to spawn. Their ID will go from 0 to (`<workers>` - 1).

Overall your server will be launched from command line with the following parameters:

`./server -q <size> -w <workers> <port_number>`, where `<size>` is a positive integer representing the maximum number of requests that can be held in the queue; and `<port_number>`, just like before, is the port on which the server should bind its socket.

Desired Behavior. Apart from spawning worker threads, processing, and rejecting requests, three pieces of information will need to be produced in output by your server.

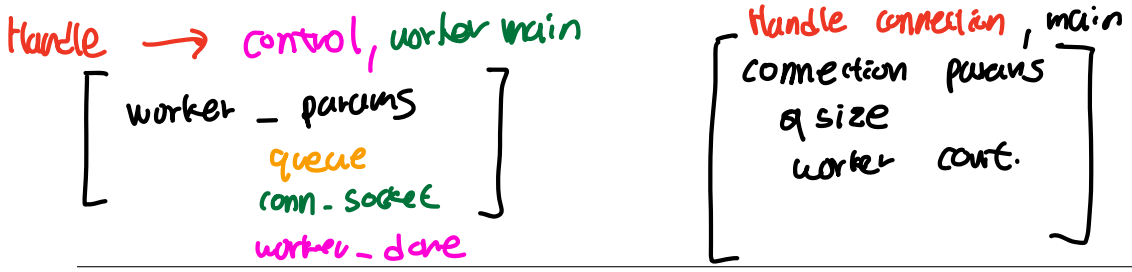
Two of these pieces of information, i.e., the queue status dumps and rejection notice are identical in format to Part 1. The only difference is that the queue dump status is printed when *any* of the worker threads completes processing of any of the requests.

Unlike Part 1 and the previous assignments, when a request successfully completes service, the thread ID of the worker thread that has completed the request will need to be added at the beginning of the line following the format below.

```
T<thread ID> R<request ID>:<sent time>,<req. length>,<receipt time>,<start time>,<completion time>
```

Here, `<thread ID>` is the ID of the worker thread that completed the given request. Once again, these IDs must go from 0 to `<workers> - 1`. If multiple worker threads are available to process a pending request, any one of them (but only at most one!) can begin processing the request.

Once again, queue status dumps and rejection notices should NOT be prefixed with a thread ID. Thus, their format is identical to what you had in `hw3`.



Submission Instructions: in order to submit the code produced as part of the solution for this homework assignment, please follow the instructions below.
You should submit your solution in the form of C source code. To submit your code, place all the `.c` and `.h` files inside a compressed folder named `hw4`.zip. Make sure they compile and run correctly according to the provided instructions. The first round of grading will be done by running your code. Use CodeBuddy to submit the entire `hw4`.zip archive at `https://cs-people.bu.edu/rmancuso/courses/cs350-fa24/codebuddy.php?hw=hw4`. You can submit your homework multiple times until the deadline. Only your most recently updated version will be graded. You will be given instructions on Piazza on how to interpret the feedback on the correctness of your code before the deadline.