COS 417 Precept

server

HTTP

- Protocol for communication between web servers (like wikipedia.org) and clients (your computer)
- URLs (Uniform Resource Locators) uniquely identify and locate resources on a server (like wikipedia.org/wiki/legume)
- *Static* content is just an unchanging file, while *dynamic* content can change based on provided data, such as arguments to the HTTP request
  - To append arguments to a URL, use the `?` character
    - e.g. wikipedia.org/wiki/legume?fake_arg=99

Requests and Responses

- Requests are sent by clients and contain:
  - `method`
    - Describes the way the client and server interacts
      - E.g. GET requests data from the server
  - `uri`
    - Where to find the file, e.g. `/wiki/legume`
  - `version`
    - HTTP version
- Responses are sent by servers and contain:
  - `status` & `message`
    - State of the request (e.g. 200 OK, 404 Not Found)
  - response headers & response body
    - Type of content, and actual content

server

- The provided code is a working *single-threaded* webserver
- Make it multithreaded, so the server can handle multiple requests at once
  - Only need to turn in `pserver.c`
  - **Highly recommend** to edit `pclient.c` for testing
- In short, we're solving the producer-consumer problem
  - Create a "pool" of worker threads that pick up connections from clients
- Also:
  - Add two arguments to server:
    - Number of worker threads
    - Buffer size
      - Holds incoming connections, placed into buffer by a "master" server thread and taken and handled by worker threads

Code Walkthrough & Running

- Take a look through these files:
  - `pclient.c`
  - `request.c`
    - `request_handle()`
  - `pserver.c`
- Run a server and client(s):
  - `./bin/pserver -d contents -p <port_num>`
  - `./bin/pclient localhost <port_num> "/spin.cgi?5"`
    - Spins for 5 seconds
      - Good for testing concurrency!

Hints & Thoughts

- Will need to use mutex locks and condition variables
  - What data needs to be locked?
  - How many of each is necessary?
- Scheduling algorithms:
  - FIFO
    - Circular buffer - how can we track where to take and remove items from the buffer?
  - Shortest File First (SFF)
    - How can we find the file sizes? `stat -> st_size`
    - Take a look at `recv()`, especially the flags argument

Testing

- The provided tests are mainly testing for **regression**
  - i.e. that your changes have not broken the basic server functionality
- We will run additional tests on your final submissions to thoroughly check for race conditions, functionality, etc.
- One strategy for finding race conditions is stress-testing your code
  - For example, you can edit the client code in `pclient.c` to spawn multiple clients/connections to the server
- **Make sure your locks are as fine-grained as possible**
  - Identify critical sections of code and **only lock those**

ThreadSanitizer

Add `--fsanitize=thread` to `CFLAGS` in your Makefile.

- Detects and prints out data race errors, similar to Helgrind
- *Doesn't seem to be installed on courselab :(. Go with Helgrind!*

https://clang.llvm.org/docs/ThreadSanitizer.html