Operating Systems

Homework Assignment #5

Due 15.1.2014, 23:59

Part 1

In this section you will write an HTTP server.

The HTTP server listens for connections, with worker threads constantly running in the background. Whenever a client connects, its socket is enqueued to a shared queue, to be handled by one of the worker threads.

Write a program `http_server.o`. It will receive three arguments: number of threads, max requests, and port number. Port number is an **optional** argument, use default HTTP port if not provided.

Implement a shared Queue data structure:

- Implemented by a linked-list (singly-linked) with a global lock, and head and tail pointers.
- The maximal size of the queue is determined by the $max\ requests$ argument. On an enqueue operation that exceeds this size, return an error code (-1) to the calling thread.
- You may assume $\max requests \geq 0$, where an argument of 0 is valid and can be used for testing.
- To enqueue or dequeue, grab the lock and perform the required operation.
  - Grab and release the lock in appropriate places! Do not hold the lock for performing operations where the lock isn't needed (allocating a queue item before enqueue, etc.).
- Use condition variables where appropriate.

In the main thread:

- Create the required number of worker threads.
- Create a socket and listen for incoming connections.
  - When a connection is made, enqueue the new socket into the shared queue.
  - If the queue is full, return *Service Unavailable* (HTTP 503) to the client and close the connection.

In each worker thread, repeatedly:

- Dequeue a socket item from the queue.
- Read the client's HTTP request and parse it.
  - Only GET/POST operations – return *Not Implemented* (HTTP 501) for any other operation.
  - Do not parse **any** info except the request line. Ignore request headers, cookies, etc.
- Return a minimal HTTP response to the client according to the request
  - Find the requested file on your machine (treat it as a path).
  - If it is not found, return *Not Found* (HTTP 404) response (including simple body).
  - If it is a directory, create a simple HTTP body listing the directory's contents.
  - If it is a file, return an HTTP response with the file's contents as the body.
- Close the socket.

Pressing CTRL+C (SIGINT) should exit the server **properly!**
i.e., use a signal handler to stop any running threads, close sockets, etc., before terminating.
You can & should wait for threads to finish handling any current clients before terminating (i.e., empty the queue).

Your browser should work with your server (navigate to URL $http://localhost:<port>/<file>$).