Design Document for Assignment 2

William Kudsk

CRUZID: wkudsk

CSE 130, Fall 2019

1. Goal:

The goal of this assignment is to build on our http server so that it uses multiple threads and logs the information

2. Assumptions:

That the client will send me an end of file message once I am done receiving data for a PUT if no content length is specified.

The server is not expected to log a PUT request that does not contain a content length

That there is no / at the beginning of the resource name

3. Design:

First I need to fix the mistakes I made on the last assignment. Then I need to create a group of worker threads that will pull client requests from the master thread. The master thread will loop infinitely adding sockets to a queue, which will be picked up from the worker threads. Worker threads are supposed to sleep until the master thread signals them to work, which is only done once it has put a socket into queue.

Worker threads will share the queue of clients, as well as the log file. To deal with this there is a lock on the offset of the log file, a lock on the pushing and pulling of the queue, and a condition variable to keep the threads idle while there is nothing to do.

For logging, we want all the threads to have access to the log file at the same time, so we use an offset to decide where each thread gets to write to the log file. The offset if calculated by a thread first, then it “reserves” a spot in the file by increasing the global offset by the amount of space it needs. This will make it

so other threads will choose the next spot in the file to log their requests. Worker threads will log error codes, GET requests, and PUT requests along with their data in hexadecimal.

4. Pseudocode:

```
handleRequest(socket) {
        if(badRequest) {
                sendRightError();
                if(log) {
                        lock(logOffset);
                        getOffset();
                        increaseOffset();
                        unlock(logOffset);
                        pwrite(loginput offset);


                }
        }
        else {
                do request…..;
        }
}
handleClient(clients) {
        if(clients.size() <= 0) wait();
        lock(client);
        socket = clients.pull();
        unlock(client);
        handleRequest(socket);

        if(log) {
                lock(logOffset);
                getOffset();
                increaseOffset();
                unlock(logOffset);
```

```
                pwrite(loginput, offset);
        }

}

createsocket()
bindsocket()
listensocket()
Let workerQueue -> size of numberOfThreads
for(worker : workerQueue) {
        createThread(worker, handClient, &clients)
}
while(1)
{
        accept()
        clients.push(new socket)
        signal();
}
```