Design Document

Rishab Jain
CruzID: rjain9

CSE:130, Fall 2019

1 Goal

The goal of this program is to create a multi-threaded server that can handle PUT and GET commands until the server is closed by the user. It also creates a log of all the requests made.

2 Assumptions

For the assignment I am assuming that the filename that is passed in the header from the client doesn't contain the '/' character. Also assuming that the response header are sent to the client and that no messages are printed to the server. Also if a file exists but it isn't a correct 27 character ascii name that the program will respond with a forbidden error. Assume for PUT case where there is no content length that the client must close the connection and the server will purposely keep reading from client until the connection is closed.

3 Design

The general design of my program is to set up the initial client server socket connection. Then I create threads depending on the command line argument and pass my dispatcher function. Initialize a semaphore for idle threads. Have an infinite while loop that contains the accept and pushes the value from the accept into my queue. Make sure to use a mutex to lock and unlock around the queue functionality because it is a global variable. My dispatcher function has all the functionality from assignment 1 and has mutexes for writing to the same file.

3 Multi-Threading

My system is thread safe because it only creates a limited number of threads given from the command line. It also makes sure the queue can't get bigger than the number of threads. If a queue is full then it will wait for a thread to be free and then proceed. This is achieved through the use of a semaphore. Also I use mutex for read/write operations that would be writing to a file. I also know that my interleaving is working correctly because I used a mutex whenever one thread is writing to the log file so another thread can't write to the log as well.

4 Pseudocode

Global Variables:

- logFileBuffer
- logFileExists(bool)
- logFd (int)
- Mutex for logFile
- Semaphore for idle threads
- Mutex for Queue
- con_t() queue conditional
- Vector for sockets (Queue)

ParseCommandLine

- Use getOpt() to check for N and l flags
- Set var for n,l,hostname, and port

Parser(buffer,vector)

- strtok() on header
- While loop while strtok() element is not NULL
    - Push back each element into vector

asciCheck(string)

- If length of string is not 27 return false
- For loop for each character in the string
    - Check if letter,number, “-”, or “_” and if it isn’t return false

logHttpOpen()

- Opens log file

logHttpClose()

- Closes log file

logHttpReq()

- Writes request header to log file

logHttpBuffer()

- While (counter < buffersize)
    - Write buffer count (00000000)

        - for(i<write Count)
            - Write hex values to log
- Write ==== endl

log FailRequest()

- Sprintf Fail request file and response
- Lock mutex
- Write to logFile
- Unlock mutex

getFuctionality(vector, socket number)

- Open file with name taken from vector
- If file throws error then
    - Print 404 not found header
    - logFailRequest
- Else if aschiCheck() returns false then
    - Print forbidden error
    - logFailRequest
- Else if fstat() return error then
    - Print 400 bad request
    - logFailRequest
- Else
    - Mutex lock
    - logHttpReq()
    - Print ok header message
    - While read from file is > 0
        - Send buffer to client
        - logHttpReq()
- Close the file
- Unlock mutex()

putFunctionality(vector, socket number)

- Go through vector and look for content length
    - If found set boolean to true and grab store content length number in variable
- Open file with create flags filename from vector
- If file throws an error then
    - Send 500 internal error
    - logFailRequest
- Else if asciCheck returns false then
    - Send forbidden header
    - logFailRequest
- If size of content length is 0 then
    - Send created message and send file that's empty
    - logFailRequest
- If content length is false then
    - Send created header
    - While recv > 0
        - Write to file
- Else
    - Send created header
    - mute
    - While recv > 0
        - logHttpBuffer()
        - Write to file
        - Keep incrementing characters read
        - Break when content length is the same as total bytes read
    - Unlock mutex
- Close file

worker()

- while(1)
    - Lock mutex()

    - If queue empty stay in cond_wait()
    - Pop first element from queue
    - read(buffer)
    - parser(buffer)
    - For vector elements
        - Compare first element to either GET or PUT
        - Call appropriate function
        - If neither get or put
            - Send 400 bad request
    - Closer socket
    - Sem post

main()

- parseCommandLine()
- socket()
- setsocket()
- gethostbyname()
- bind()
- listen()
- sem_init()
- logHttpOpen
- Create pthreads
- while(1)
    - accept()
    - sem_wait()
    - Mutex lock(queue)
    - Queue add accept() value
    - cond_signal(Queue)
    - mutex() unlock
- logHttpClose()

- Close() server

-