1 Goal

The goal of this program is to create a web server that implements caching.

2 Assumptions

None

3 Design

The general approach I'm taking is in the main function I will parse the command line arguments. Using the code from the socket programming geeks for geeks page, I was able to get a server set up and ready for one request. I then made it loop on the accept command so the server will process multiple requests back to back. In this loop the second thing I do is receive and parse the initial header message sent from the client. I parse the type of request and filename, and content length. I then run my isValidName() function that checks for the length of the file name and if it contains the valid ascii characters. If it is not a valid file name, the server sends the client a 400 error and closes the socket and continues in the loop. If it is a valid file, the server will call the putReq() or getReq() respectively if its a put or get request.
The putReq() function will create the file with the parsed name from the initial header message, and then will receive the file content in the socket and write to the file on the server. Put request will always write to index 0 of the cache(assuming, if the filename is in the cache, I will move it to the beginning). In this move operation, if a file at the end of the cache gets popped off, it will be written to disk so the client can still request this file. Once complete, the server will send an okay response back to the client and then close the file and socket. If a log file is passed into the function, the contents of the file will be logged while processing request.
The getReq() function will open a file on the server first. (It will first check the cache to see if the file is in the cache, if it is, the server will serve this request, if not, it will check to see if the file is saved on disk). If this fails, it will send 403 error if the file is locked/doesn't have permissions or the server will send a 404 error if the file is not found. It will then close the socket and return. If the files okay, the server send an okay message and the content length, after curl receives that, the server will read from the file and send the contents of the file out on the socket. It then closes the file and closes the socket. If a log file is passed into the function, the contents of the file will be logged while processing request.

If it’s neither a put or get request, the server will send the client a 500 error response and close the socket.

4 Pseudocode

Python like indentation for pseudocode.

```
Struct cacheArr{
char* content=””;
char* fileName=””;
};
Struct cacheArr cacheArray[4]

Main:

Parse arguments, set up flags for caching and logging.

[Socket code from geeks from geeks section here :D --
https://www.geeksforgeeks.org/socket-programming-cc/]

while(1)://inf while loop
    Newsockfd = Accept();
    Read from socket into buffer // this is the header message
    Parse Header using strtok and sscanf -- grab type of request, filename and content
length. Req holds request type, filename holds the name of file.
    Bool validFile = isValidName(filename)
    If (validFile == 0):
        Send 400 bad request error
        Close socket
        Continue
    if(req == Put & validFile!=0):
        //putReq()
        helpPut()
    If else(req == Get & validFile!= 0):
        getReq()
    Else:
        Send 500 Internal Server Error
End While
End Main
```

```
getReq(int newsockfd, char * fileName):
    Fd = Open(fileName)
    If (fd < 0):
        if(ERRNO == EACCES):
            Send 403 error.
            Close socket
            Return;
        else:{}
    Send 404 error
    Close socket
    Return
    End Initial IF
    Fs = getFileSize(fd)
    curPos = 0;
    While (CurPos < FS):
        read(fd, buf,..)
        send(newsockfd,buf,..)
    End While
    close(Fd)
    Send Okay and Content Length
    close(newsockfd)

End getReq()




putReq(int newsockfd, char * fileName, int len. Int logfd...):
    Process "header" log
    Bool createdFile = false
    If open(fileName) < 0:
        createdFile = true
    Fd = open(filename,....)
    curPosFile = 0
    contentLen = len
    Char buf[11]
    while(curPosFile < contentLen):
        recv(newsockfd,buf...)
        writeCache(buf);        // new
        :Log contents in hex:   // new
        write(fd,buff..)
        If recv or write fail -> close file and socket and return;
        Process “tail” log
        If (CreatedFile):
                dprintf(newsockfd,”HTTP/1.1  201 Created”)
        Else:
                dprintf(newsockfd,”HTTP/1.1 200 OK”)

        close(fd)
        close(newsockfd)
End putReq()



helpPut():
        if(in cache):
                Sweep swaping left at index.
                Delete and new char --- (or zero it idk whatever ezer)
                put(--into cache 0)
        if(NOT in cache):
                Sweep swapping left.
                put(--into cache 0)


helpGet():
        if(in cache)
                getCache();
        Else
                get():



getCache(int newsockfd, char * fileName):
        cacheArray[0].content
        Fs = getFileSize(fd)
        curPos = 0;
        Send header;
        While (CurPos < FS):
                memcpy(buf,cacheArray[0].content,length of content)
                send(newsockfd,buf,..)
        End While
        close(Fd)
        Send Okay and Content Length
        close(newsockfd)
```

```
End getReq()
```