# Asgn1: Design Document

Alexander Soe cruzid: asoe

CSE130, Fall 2019

## Goal

The goal of this assignment is to create an http server that can receive GET and PUT requests. If a client GET requests the server, the server should be able to return the data present in the file that the client requests if it exists. If a client PUT requests the server, the server should be able to create a file specified by the client and print the contents of the resource file from the client into the local file. On any other requests the server should respond with the appropriate status responses or error messages.

## Assumptions

I'm assuming that the client is going to be the person using curl or net cat while the actual server is going to be created by running ./httpserver. I'm going to need to find someway to parse the header and only the header to retrieve the content length (if it exists), the name of the resource file, and either the GET or PUT command. Additionally, I'm also going to need to find a way to check if a file exists or if I have permissions to certain files/if the file is a directory. Lastly I need to figure out a way to return response headers to each of the clients with the respective status codes and content lengths.

## Design

First we have to set up the server socket by setting the sock opt, binding the socket to either local host and port 8080, or whatever address and port the user gives us, listen for a client socket, accept said socket. Then in order to start sending and receiving data via GET and PUT commands, we have to find the first header and parse it for its content length if it has one and its command. We also need to parse for the filename to check if it is a legal filename of length 27 as well as check if the command is one that we can actually perform. Now based on its command and content length we can decide what to do. If the command is a GET we must first check if the file the client is requesting exists. Then we open the file descriptor and read all of its contents into a buffer in which we send it to the client's socket. If read doesn't return 0, then we keep filling the buffer and send it to them until there is no more data. We must also keep the connection open if it is a GET command. If the client issues a PUT command with no content length, then we must open a file on our side or overwrite a file on our side if the file already exists. Then we must open the client's socket descriptor and read the contents until we read EOF. If the PUT has a content length then we must do the same as the last PUT command except for that we read until the content length. Also we must leave the connection open in case there are more commands that need to be read.

## Pseudocode

```
String readHeader(int fd)
{
	while(read(fd) > 0)
	{
		if(strstr(“\r\n\r\n”) != nullptr)//check for end of header
		{
			break;
		}
	}
	if(n < 0)
	{
		close(fd);
		Return string
	}
	Return string
}

Int getContentLength(string header)
{
	strstr(“Content-Length:”) //find content length keyword
	if(Content-length is there)
	{
		sscanf(“Content-Length: %d”)
		If(contentLength > 0)
		{
			Return length;
		}
	}
	Else
	{
		Return error
	}
	Return -1;
}
```

```
Bool isCorrectFileName(string filename)
{
	if (filename is greater than or less than 27)
	{
		Return false;
	}
	if(characters are the right ascii values)
	{
		Return false;
	}
	Return true;
}

String getFileName(string header)
{
	Vector tokenVector;
	Stringstream tokStream;
	String token;
	while(getline(tokstream))
	{
		tokenVector.push_back(token);
	}
	String filename = tokenVector[1];
	Return filename;
}

Void handlPut(filename, contentLength, client_fd)
{
	if(access(file access))
	{
		if (access(write access))
		{
			Write 403 response
		}
	}
	if(fileExists and there is contentlength)
	{
		Write 200 response
	}
	if(!fileExits and contentLength)
	{
		Write 201 response
	}
```

```
        if(contLength != -1)
        {
                create(file);
                while(contLength > 0)
                {
                        read(client);
                        write(filecontents to newfile);
                }
        }
        close(newfd);
        Else
        {
                read(client);
                write(till eof)
        }
}

Void handleGet(filename, client_fd)
{
        if (access(file exists))
        {
                Write 404;
        }
        if (access(read ok))
        {
                Write 403;
        }
        write(client_fd, header length);
        while(contentlength != 0)
        {
                Read(fileContents);
                write(client_fd)
        }
}

handle()
{
        while(there is a header)
        {
                //Call get
                //Call put
                //Check for errors
        }
}
```

```
Int main()
{
	//Do all the socket setup stuff
	//Call all the file
	while(1)
	{
		handle(client_fd)
	}
}
```