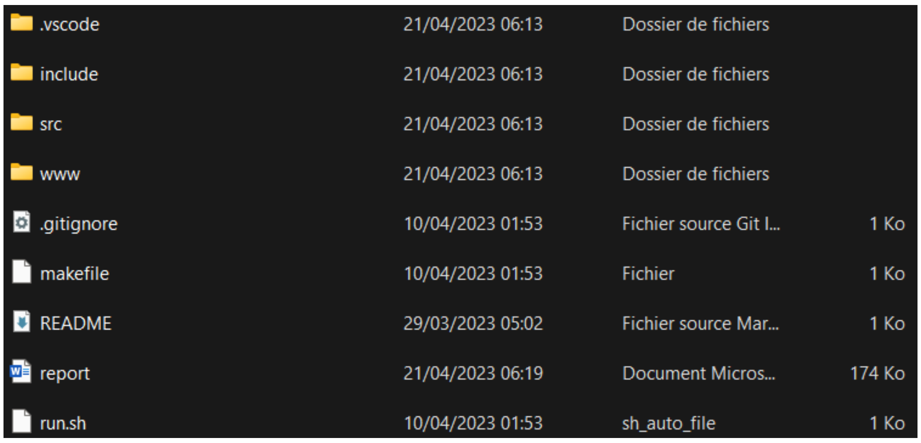
CITRINI Mathias

Project 1: Web server

I. Architecture of the project



.vscode	21/04/2023 06:13	Dossier de fichiers	
include	21/04/2023 06:13	Dossier de fichiers	
src	21/04/2023 06:13	Dossier de fichiers	
www	21/04/2023 06:13	Dossier de fichiers	
.gitignore	10/04/2023 01:53	Fichier source Git I...	1 Ko
makefile	10/04/2023 01:53	Fichier	1 Ko
README	29/03/2023 05:02	Fichier source Mar...	1 Ko
report	21/04/2023 06:19	Document Micros...	174 Ko
run.sh	10/04/2023 01:53	sh_auto_file	1 Ko

Makefile	makefile
README and .gitignore	Useful for GitHub dev
include/	Contains Header
src/	Contains source files (.c)
www/	Contains website and all the assets (css, images)

II. Commands (in root)

❖ To compile the project, or clean it:

Command: `make`
Command: `make clean`

❖ To start the server, use:

Command: `./server <port>`

Replace <port> with your own port (ex: 8077)

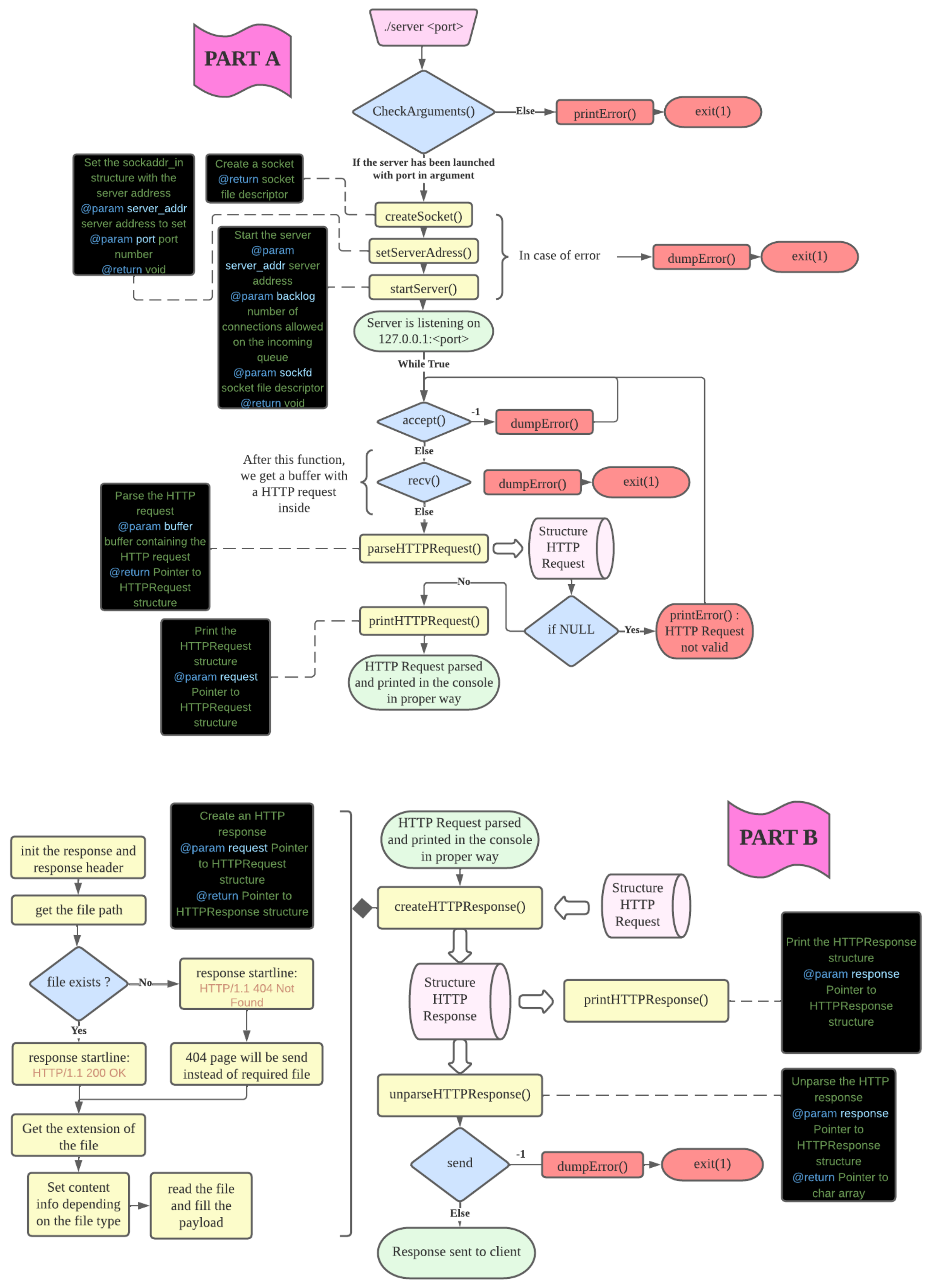
III. Webserver hub

I created a dummy website to test every requirement easily. This website is a basic one and use only one js script and one css file. It can display images, has video integration…

To access this website, please go to: <ip address>:<port>/index.html or <ip address>:<port>/

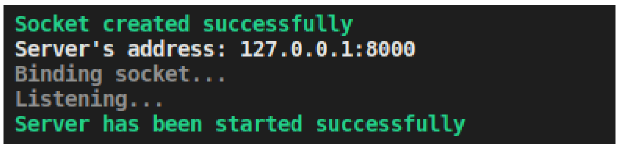
Use Google Chrome or Mozilla to have better performance.

IV. How the server works?

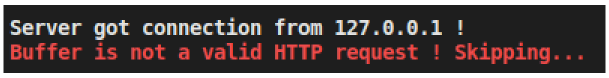

PART A
./server <port>
CheckArguments()
Else
printError()
exit(1)
If the server has been launched with port in argument
Set the sockaddr_in structure with the server address @param server_addr server address to set @param port port number @return void
Create a socket @return socket file descriptor
Start the server @param server_addr server address @param backlog number of connections allowed on the incoming queue @param sockfd socket file descriptor @return void
createSocket()
setServerAdress()
startServer()
In case of error
dumpError()
exit(1)
Server is listening on 127.0.0.1:<port>
While True
accept()
-1
dumpError()
Else
After this function, we get a buffer with a HTTP request inside
recv()
dumpError()
exit(1)
Else
Parse the HTTP request @param buffer buffer containing the HTTP request @return Pointer to HTTPRequest structure
parseHTTPRequest()
Structure HTTP Request
No
if NULL
Yes
printError() : HTTP Request not valid
Print the HTTPRequest structure @param request Pointer to HTTPRequest structure
printHTTPRequest()
HTTP Request parsed and printed in the console in proper way
PART B
init the response and response header
get the file path
file exists ?
No
response startline: HTTP/1.1 404 Not Found
Yes
response startline: HTTP/1.1 200 OK
404 page will be send instead of required file
Get the extension of the file
Set content info depending on the file type
read the file and fill the payload
Create an HTTP response @param request Pointer to HTTPRequest structure @return Pointer to HTTPResponse structure
HTTP Request parsed and printed in the console in proper way
createHTTPResponse()
Structure HTTP Request
Structure HTTP Response
printHTTPResponse()
Print the HTTPResponse structure @param response Pointer to HTTPResponse structure
unparseHTTPResponse()
Unparse the HTTP response @param response Pointer to HTTPResponse structure @return Pointer to char array
send
-1
dumpError()
exit(1)
Else
Response sent to client

V. Server's outputs analysis

❖ Server status messages



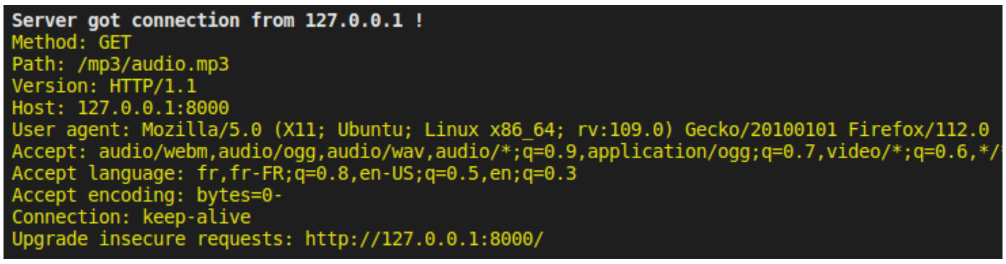
```
Socket created successfully
Server's address: 127.0.0.1:8000
Binding socket...
Listening...
Server has been started successfully
```



```
Server got connection from 127.0.0.1 !
Buffer is not a valid HTTP request ! Skipping...
```

The server can print status messages in the console. Green messages are confirmations of successful operations, red messages are errors and white and grey messages are server status and information.

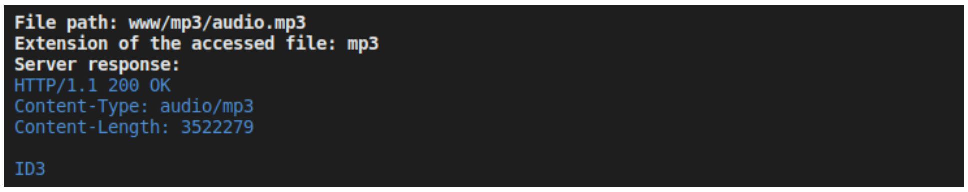
❖ Request received.



```
Server got connection from 127.0.0.1 !
Method: GET
Path: /mp3/audio.mp3
Version: HTTP/1.1
Host: 127.0.0.1:8000
User agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/112.0
Accept: audio/webm,audio/ogg,audio/wav,audio/*;q=0.9,application/ogg;q=0.7,video/*;q=0.6,*/
Accept language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept encoding: bytes=0-
Connection: keep-alive
Upgrade insecure requests: http://127.0.0.1:8000/
```

In this situation, we can see that the server received an HTTP request from a client (IP:127.0.0.1). The client is running the web browser Mozilla Firefox on Ubuntu and is requesting an mp3 file (audio.mp3). The path of the file is /mp3/audio.mp3. HTTP requests are printed in orange/yellow in the console.

❖ Respond sent.



```
File path: www/mp3/audio.mp3
Extension of the accessed file: mp3
Server response:
HTTP/1.1 200 OK
Content-Type: audio/mp3
Content-Length: 3522279

ID3
```

In this situation the server is responding to the previous request received (see above). The server gets the "real" file path form the path sent by client (all the website files are in www folder). The file exists so status code of the message is 200. Then, the server set the content-type to audio/mp3 and fill the payload with the file content. Moreover, Content-length will contain the size of the requested file.
HTTP response are printed in blue in the console.

VI. Difficulties

The first difficulty was when I had to parse the HTTP request. Sometimes, I had core dumped so I had to choose the best size for each field and optimize the memory allocation. So, I used malloc almost every time.

The second difficulty was to understand how to send binary files (Images, Audio …). In fact, I took some time to understand how binary files work and how to read it because I did not use to. I learned the best way was to use void* instead of char*, and so, memcpy instead of strcat. After long hours of pain, I finally managed to get the proper size of the file, read it with fread and copy the binary data in a string to send it to the client.

To summarize, memory problem and binary data were my biggest difficulties in this project.