Design document: HTTP server

Zhihang Zhou
ID: Zzhou37

Goals:

In this program we need to build a server can handle a single thread request, basically the server has to listen all the clients' request and respond one costumer each time. There are two requests the server need to handle PUT and GET. When client PUT the file into the server, first the server will reply OK to client, then the server will receive the data and overwrite the date into the local repo with the length client required. If there is not specific request of length, then keep receiving u data until client close the connection and acknowledge the client file been created. When client GET file, the server will send the file client request back to client and provide client the size of the file server send. If the file doesn't exist, the server will send to client 404 file not found. During this process, the server or client will not wait for each other(they are asynchronies).

If the requirement is bad then send BAD REQUEST, if the requirement is invalid then send REQUEST is invalid

In this assignment we can't use FILE* in standard library and HTTP in standard library neither. We can use standard network system call. We can use sprintf() and sscanf() to print.

Design:

Set up server:

Bind a socket with HTTP format and get ready to receive request:

- Turn a host name to network standard
- Create a socket
- Bind the socket with net word standard
- While true
  - Accept connection return socket descriptor
  - Handle the socket descriptor for client

Handle the socket descriptor for client:

1. Parse the header

Parse the header file to get operation and fileID which is the name of the file. If content length exists, catch the length as well

Note: Assume the file header < 4KB

1.1 get header
to get the header of the file:

take socket descriptor, and 4kb of buffer as input
read header into a 1kb buffer
if we read -1 byte
    send network error message
combine the 1kb of buffer with 27kb of buffer
repeat reading 1kb and combind until we reach the \r\n\r\n

1.2parse header:
Take header as input, pass out the operation and fileID. Return length
(if not valid, return -1; if no target, return -1)
Handing the invalid request and bad request at same time

read the first line of the header
get the operation, fileID
if fileID is not 27 char or have forbidden char in it
    send message request is not valid
if the operation is not GET or PUT
    send message request is not valid
if the fileID have "/" in it
    send message request is not valid

for each line in the header //ending in \r\n
    search for "content-length: "
    if content-length exist
        return the integer follow it

if nothing finds, return -1//means content

Handing the request:

PUT:

If client request to send n byte of date, the receive the data from client and write n byte of them into local repo. If not specific length of file request, read as much as it can until the client close the connection.

Implement: use read, write and open to read from client, write to server and open file

The header of PUT is like this:

```
PUT ABCDEFabcdef012345XYZxyz-mm HTTP/1.1
Content-Length: 460
```

Composed by the name of the command:
First line: PUT + the 27 characters HASCI file name and the destination.
Second line is optional: The length of the character, if the line doesn't exist then the server will read the following file until the client close the connection.

The following is the pseudo code for Put:

Taking the fileID, content-length as necessary input

Send message Creat to client

If length is -1
    then disable the length mode

if fileID exist create a new file and delate the old one, then open it

do
    read 28KB from client return read size
    if return size from read is -1
        send network error message
    length = length – read to see how much left that we need to read
    if length is less than 0 and length mode is on
        set read size as length + size we read
        set on off of the while as close

write the size of data of what we read to local repo
while the size we read is not zero repeat the process

send message OK
send content length: 0
empty line

return 0

GET:

The header is similar with the PUT, instead of PUT, the name is GET, which request the server send the file to client. When the file finished send costumer an acknowledgement OK(200)

Use the similar function in put
Note: the GET request doesn't have content length request.

Following is the sudo code for Get:
Take fileID and socket scripter as input

Open file with the fileID

If file doesn't exist
Send 404 page not found

Send message OK
send the sum size we write to client as content length

Do
Read 28KB from the file, get the read size
Keep tracking the amount of size we read
Do
write the same size of the data we read to client, get write Size
if we read 0 byte then jump off the loop
if write size is -1, send network error return -1
sum write = sum Write + write Size
read Size = read Size – write Size
while read Size is 0, which means we have already sent all the data we just read
while true then read more data

return 0