**APLICAȚIE**

**MINI SERVER WEB**

****

Îndrumător:

Lt. Dan Avram

Realizat de :

Sd. Sg. Buzatu Constantin-Mihai

Sd. Sg. Banu Ion

Grupa C113A

**Crearea unui server web**

# Ce este un server web?

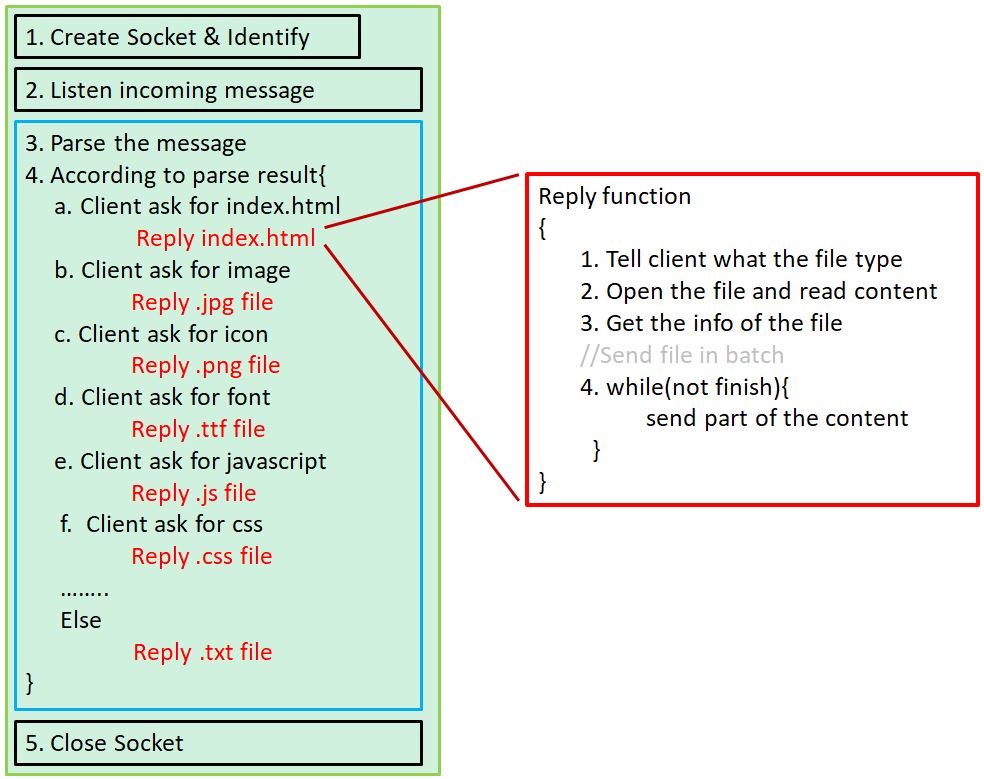
Este un program executabil, care primește cereri de la un client și generează un răspuns pentru acestea.

Cererea presupune dorința unui client de a accesa un anumit document stocat pe server, iar răspunsul constă în transmiterea conținutului documentului.

# Cerinte funcționale

* În crearea serverului vom folosi multithreading-ul. Acesta presupune folosirea mai multor fire de execuție în cadrul aceluiași process, cu scopul gestionării simultane a mai multor cereri de la clienți.

Construirea server-ului este evidențiată prin cerințele următoare:

* Crearea și configurarea socket-ului serverului:
  + Folosim funcția *socket* din librăria *sys/socket*.
  + Utilizăm *AF\_INET* pentru a specifica tipul de adresă cu care lucrăm (IPv4).
  + Setăm adresa ip de *localhost*.
  + Setăm numărul portului *8080*.
  + Asociem socket-ul cu adresa serverului folosind funcția *bind()*.
  + Folosim funcția *listen()* pentru a seta socket-ul să asculte conexiunile clienților.
* Așteptarea și acceptarea conexiunilor se vor face folosind funcția *accept()*.
* Crearea unui thread, pentru fiecare conexiune acceptată, se realizează folosind funcția *pthread\_create*. Fiecărui thread îi este asociat un socket, totodată și un client diferit.
* Implementarea funcției, care va gestiona cererile din cadrul fiecărui thread, presupune:
  + Serverul web trebuie să poată primi cereri HTTP de la client și să le gestioneaze corespunzător.
  + În funcția asociată fiecărui thread, citim mesajul primit de la client folosind funcții de citire de pe socket precum *read* sau *recv*.

Se va analiza mesajul HTTP pentru a extrage informații.Acestea sunt: metoda cererii (*GET*, *POST*), calea documentului, antetul și corpul cererii.

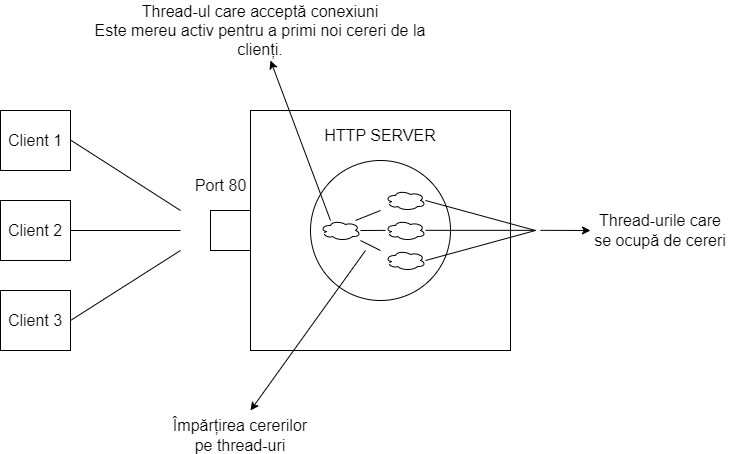
* Crearea răspunsului se realizează prin următorii pași:
  + Generăm linia de status care include: versiunea HTTP, codul de stare și un text ce reprezintă statusul comenzii.
  + Vom crea antetele ce conțin informațiile: Content Type, Content Length.
  + Corpul răspunsului este format din conținutul documentului. Acesta poate fi reprezentat de cod HTML, text, poze.
  + Vom calcula dimensiunea corpului pentru setarea antetului Content-Length.
  + Generarea răspunsului propriu-zis reprezintă combinarea liniei de status, antetelor si a corpului, care vor fi trimise ulterior către client prin socket-ul conexiunii utilizând funcțiile *write* sau *send*.
  + La final se va închide socket-ul clientului, se va elibera elibera memoria folosită și se va închide firul de execuție, urmând să se deschidă altul, în cazul apariției unei noi cereri.

Folosirea multiprocessing-ului:

* + Se va folosii aceași abordare ca la multithreading, de la crearea socket-ului până la acceptarea unei noi conexiuni.
  + În loc de thread-uri se vor crea procese copil, folosind funcția *fork()*.
  + Dacă ne aflăm în procesul copil (pid = 0): Se va inchide socket-ul de ascultare, nefiind nevoie de el în procesul copil. Gestionăm cererea clientului și apoi vom închide procesul copil.
  + Trecem la procesul părinte (pid > 0): După ce procesul copil a fost creat cu succes, părintele se întoarce la așteptarea de noi conexiuni. Închidem socket-ul clientului, deoarece nu avem nevoie de el și pentru că doar copilul se ocupă de acest socket. Procesul părinte va aștepta ca oricare dintre procesele copil să se termine. Folosim **waitpid(-1, NULL, WNOHANG)** cu scopul de a evita crearea de ”zombie processes” (procese care au terminat executarea, dar încă apar in tabelul de procese), parintele continuând să ruleze și să accepte noi conexiuni.
* Implementarea unei modalități de a folosi scripturi prin intermediul CGI (Common Gateway Interface), care se realizează astfel:
  + Identificăm cererile de tip CGI. Aceasta se poate face pe baza extensiei fișierului (\*.cgi), locație URL-ului.
  + Extragem informațiile din cerere, precum parametrii GET și POST.
  + Creăm un nou proces responsabil de executarea script-ului.
  + Manipularea redirecționării de intrare și ieșire se va realiza prin *stdin* care va furniza date către script și *stdout* care va captura rezultatele.
  + Rulăm scriptul redirecționând intrarea standard *stdin* ce furnizează date necesare script-ului, apoi vom captura ieșirea standard *stdout* pentru a obține rezultatele
  + Trimiterea rezultatelor către client se va realiza când script-ul CGI a terminat de rulat pe socket-ul asociat conexiunii.

# URL - structură

# Workflow



# Cererea

Există diferite tipuri de request-uri, dar cel mai comun este GET. Forma unui astfel de request este: GET document\_path HTTP/version.

Header-urile: Acestea conțin informații suplimentare despre request. Spre exemplu:

* Host specifică adresa serverului.
* User-Agent indică browserul și sistemul de operare utilizat.
* Accept specifică tipurile de conținut acceptate de client
* Accept-Language, Accept-Encoding specifică preferințele de limbă și de compresie a conținutului.

Exemplu:

**GET /pagina.html HTTP/1.1**

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:70.0) Gecko/20100101 Firefox/70.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

# Răspunsul

Când serverul primește o cerere HTTP, acesta localizează documentul cerut și îl trimite. Forma unui astfel de răspuns este: HTTP/version code text

Version = versiunea de HTTP folosită

Code = un număr format din 3 cifre, un status code

Text = un mesaj care descrie dacă totul este OK

După prima linie urmează antetul, care conține informații despre document, precum length-ul sau tipul, urmat de un spațiu și de conținutul documentului.

Exemplu:

HTTP/1.0 200 OK

Server: Netscape-Communications/1.1

Date: Tuesday, 25-Nov-97 01:22:04 GMT

Last-modified: Thursday, 20-Nov-97 10:44:53 GMT

Content-length: 6372

Content-type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">

<HTML>

...conținutul documentului...

# CGI

Este o modalitate pentru server de a interacționa cu alte programe externe.

Se poate folosi prin crearea unui director CGI pe server și setarea acestui director ca fiind unul cu scripturi, care se vor executa când se va cere. Acestea reprezinta simple programe executabile, scrise în orice limbaj de programare.

Înainte ca programul să pornească, serverul setează un număr de variabile care conțin informații primite prin cerere.

Răspunsul se va face prin standard output.

# Bibliografie

https://forhjy.medium.com/42-webserv-cgi-programming-66d63c3b22db

https://www.linkedin.com/pulse/damn-threads-web-server-too-nikhil-srivastava/

https://github.com/Dungyichao/http\_server

https://www.garshol.priv.no/download/text/http-tut.html