

UNIVERSITÉ DE ROUEN



FACULTÉ DES SCIENCES TECHNOLOGIES
MENTION INFORMATIQUE

Intitulé du Projet :

Projet Du Réseau



Présenté par :

- * **Fofana Abdoulaye.**
- * **Khimeche Sakina.**

Année Académique 2021-2022

TABLE DES MATIÈRES

1	Présentation Du Projet :	4
1.1	Description D'Un Serveur HTTP 1.0 :	4
1.2	Les Fonctionnalités :	5
2	Réponses aux Questions :	6
2.1	Etape 1 :	6
2.2	Etape 2 :	8
2.3	Etape 3 :	9
2.4	Etape 4 :	9
2.5	Etape 5 :	12

TABLE DES FIGURES

1.1	Schéma de l'architecture clientS/Serveur HTTP	4
2.1	Serveur HTTP -Bonjour-	6
2.2	Déscription de la reponse de serveur	7
2.3	Type contenu Format (text/plain)	7
2.4	Type contenu Format (text/html)	8
2.5	Affichage Page HTML	8
2.6	Gonction Pour ouvrir l'index.html	9
2.7	Appel Index.html	9
2.8	Serveur HTTP	10
2.9	Les Cas D'Erreurs	10
2.10	Cas D'Erreur 404	11
2.11	Cas D'Erreur 400	11
2.12	Cas D'Erreur 501	12
2.13	Fonction Mime	12

Introduction Générale

Dans ce projet, notre objectif est de réaliser un serveur HTTP minimal capable de répondre à une requête provenant d'un navigateur web.

le serveur HTTP sera multithreads qui répondra au navigateur client par une page html contenant des messages ou des codes d'état avec une description .

Pour la réalisation finale de ce projet on avait affaire a des questions, chaque question nous aidera à passer a une autre étape pour la réussite du projet.

CHAPITRE 1

PRÉSENTATION DU PROJET :

1.1 Description D'Un Serveur HTTP 1.0 :

Dans ce chapitre nous allons presenter notre projet et ses fonctionnalités :
Voici le schema associé au projet :

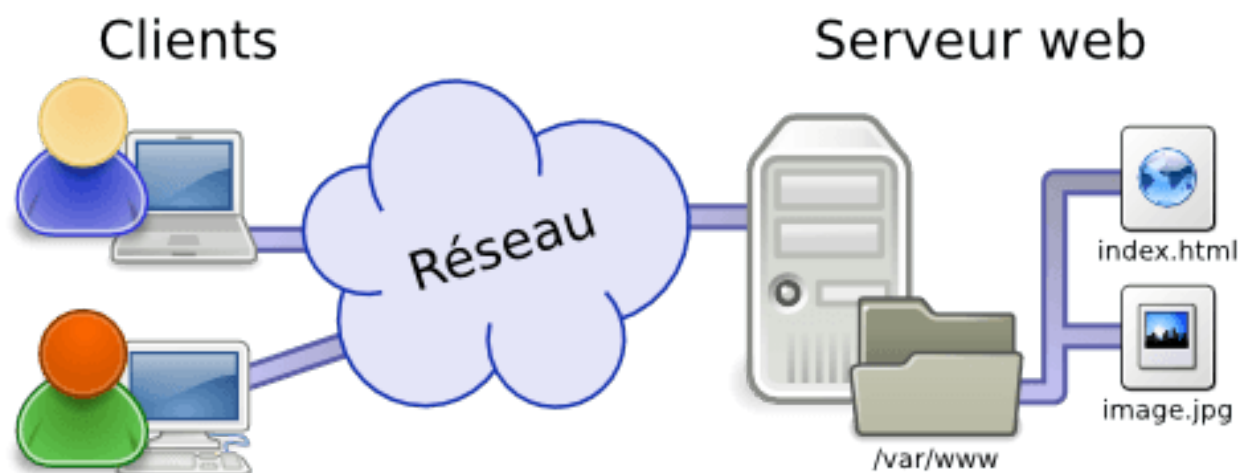


FIGURE 1.1 – Schéma de l'architecture clientS/Serveur HTTP

La version 1.0 du protocole, spécifiée dans la RFC 1945 , permet aux client et serveur d'envoyer plus d'informations entre eux, ainsi que de gérer le cache et les erreurs .[extrait du polycopie de l'enoncé du projet]

1.2 Les Fonctionnalités :

Le but de ce projet est de créer un serveur HTTP minimal capable de répondre à une requête provenant d'un navigateur web.

1. On utilisera les bibliothèques AdresseInternet et SocketTCP.
2. On utilisera comme un langage de programmation le langage C.
3. Un fichier HTTP
4. Les appels systemes "cas d'erreur" 400, 404 et 501

CHAPITRE 2

RÉPONES AUX QUESTIONS :

Dans ce chapitre nous allons répondre aux questions donnaient au sein du projet comme étapes :

2.1 Etape 1 :

on a debutait par crée un serveur HTTP 1.0 qui répond au navigateur client par le message Bonjour :

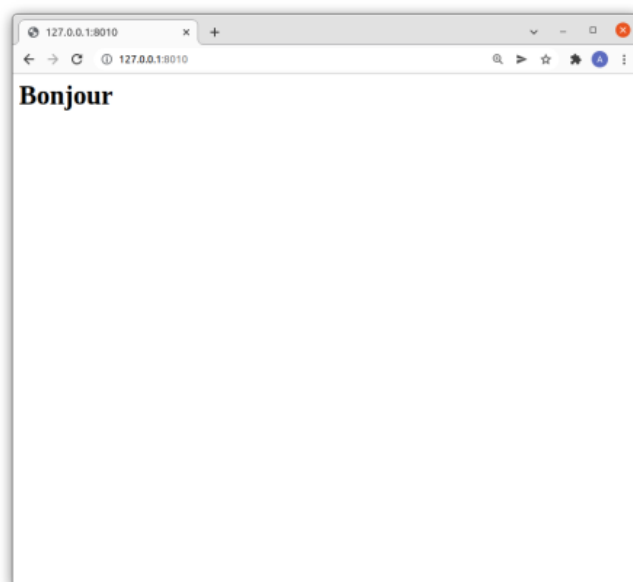
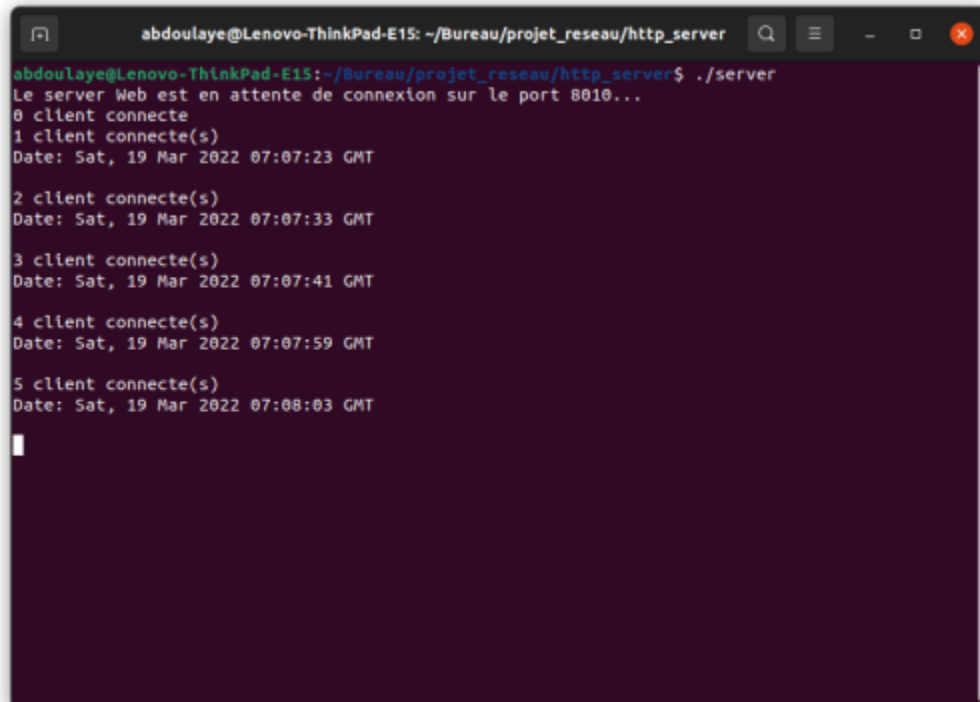


FIGURE 2.1 – Serveur HTTP -Bonjour-

Ici le serveur dans sa réponse, il spécifier la date et le type du contenu envoyé (text/plain). Le serveur devra gérer plusieurs clients à la fois en lançant pour chaque client un thread qui se charge de lui répondre comme c'est apparu dans la figure suivante on voit très bien



```


abdoulaye@Lenovo-ThinkPad-E15: ~/Bureau/projet_reseau/http_server
abdoulaye@Lenovo-ThinkPad-E15:~/Bureau/projet_reseau/http_server$ ./server
Le server Web est en attente de connexion sur le port 8010...
0 client connecte
1 client connecte(s)
Date: Sat, 19 Mar 2022 07:07:23 GMT
2 client connecte(s)
Date: Sat, 19 Mar 2022 07:07:33 GMT
3 client connecte(s)
Date: Sat, 19 Mar 2022 07:07:41 GMT
4 client connecte(s)
Date: Sat, 19 Mar 2022 07:07:59 GMT
5 client connecte(s)
Date: Sat, 19 Mar 2022 07:08:03 GMT

```

FIGURE 2.2 – Description de la reponse de serveur

que plusieurs clients peuvent se connecter à notre serveur, renvoyer la date .. la fonction qui permet de spécifié le type de contenu envoyé sous forme (text/plain) est :

```

/**
 * Server TTP réponse
 *
 * En-têtes: "HTTP/1.1 404 NOT FOUND" or "HTTP/1.1 200 OK", etc.
 * content_type: "text/plain", .
 *
 * Return the value from the send() function.
 */

```

FIGURE 2.3 – Type contenu Format (text/plain)

Ici Le serveur ignorera la requête du client et ses en-têtes

2.2 Etape 2 :

Ici le serveur devra spécifier le type de contenu envoyé sous forme (text/html) et répondra au client avec une page HTML.

```
<!DOCTYPE html>
<html>
<head>
  <title>Serveur HTTP</title>
  <style>
    body{
      }
    @media screen{
      text-align: center;
      color: red;
      font-size: 30px;
    }
    @media print{
      text-transform: capitalize;
      color: black;
      font-weight: bold;
      text-align: center;
      font-size: 15px;
    }
  </style>
</head>
<body>
<p><span>1234567890</span> <div> Page not Found</div>
</body>
</html>
```

FIGURE 2.4 – Type contenu Format (text/html)

Le serveur ignorera aussi l'en-tête, l'affichage sera ainsi :

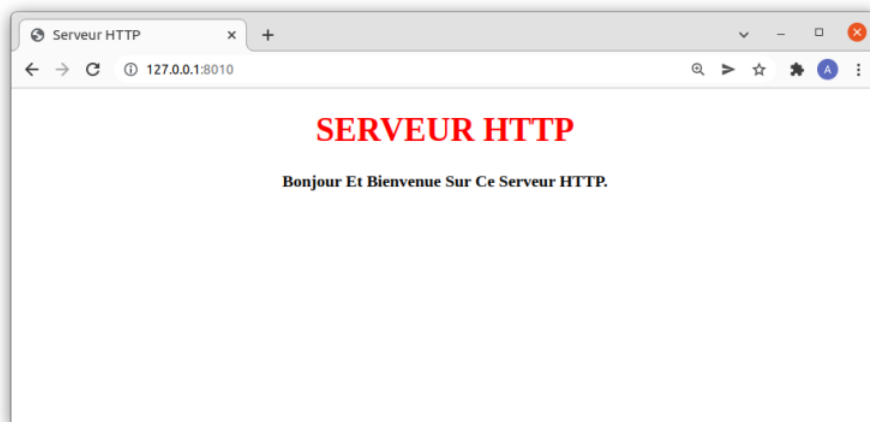


FIGURE 2.5 – Affichage Page HTML

2.3 Etape 3 :

Le serveur envoie au navigateur client un fichier index.html :

```
FILE * index = fopen("../data/index.html", "r");

if(index == NULL)
{
    printf("Erreur lors de l'ouverture du fichier ...");

    return NULL;
}
```

FIGURE 2.6 – Fonction Pour ouvrir l'index.html

ce fichier se trouve dans le répertoire d'exécution du serveur. Ce serveur ignorera aussi la requête du client et ses en-têtes.

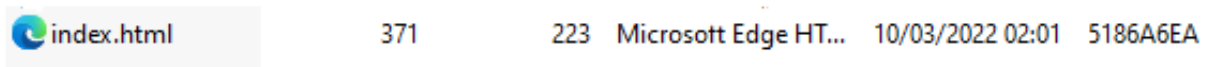


FIGURE 2.7 – Appel Index.html

2.4 Etape 4 :

Le serveur qui envoie le fichier HTML demandé par le client lorsque celui-ci fait une requête GET.

Ce fichier doit se trouver dans le répertoire d'exécution du serveur.

Le serveur devra lire la ligne de commande, la découper, extraire le chemin du fichier à partir de l'URL.

Comme ci-dessous :

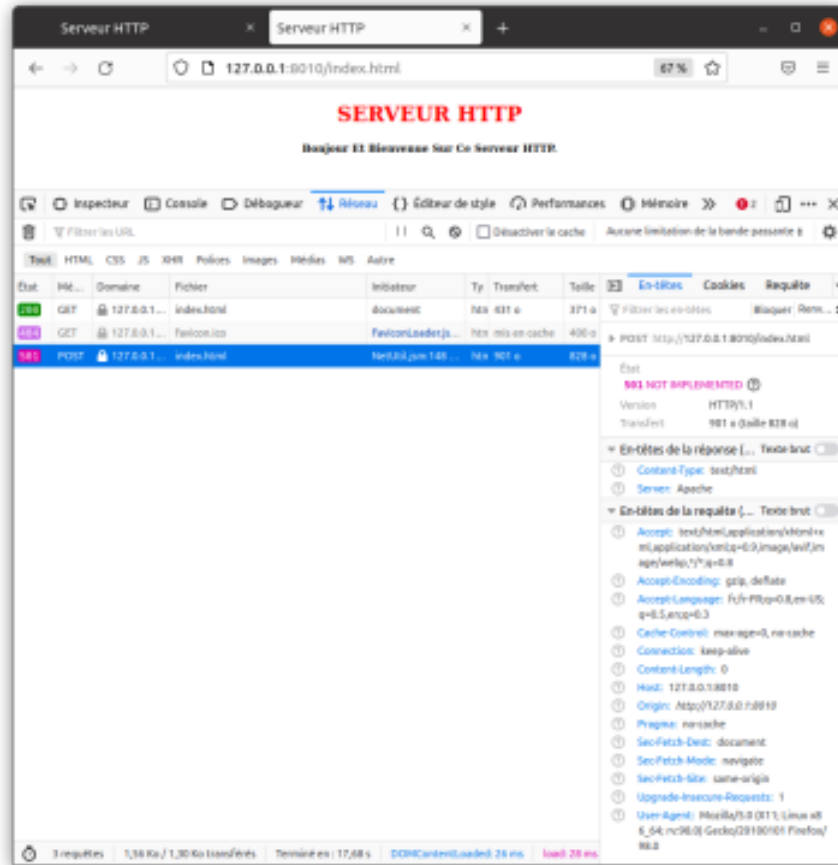


FIGURE 2.8 – Serveur HTTP

On ignorera les en-têtes de la requête. Il faudra gérer le cas d'erreur 404, 400 et 501 on faiant les appels système suivants :

400.html	0	0	Microsoft Edge HT...	10/03/2022 01:51	00000000
404.html	0	0	Microsoft Edge HT...	10/03/2022 01:50	00000000
501.html	0	0	Microsoft Edge HT...	10/03/2022 01:50	00000000

FIGURE 2.9 – Les Cas D'Erreurs

Maintenant, on expliquera chaque erreur avec une demonstration de l'interface apres son execution :

1. [L'erreur 404](#) : Le client tente d'accéder à une ressource qui n'existe pas , il lui affiche une page HTML contenant une description "**PAGE NOT FOUND**"

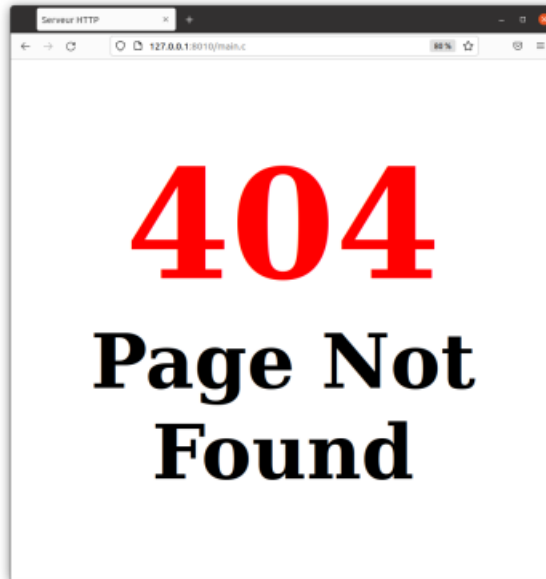


FIGURE 2.10 – Cas D'Erreur 404

2. [L'erreur 400](#) : le client tente d'accéder à une ressource qui existe mais la synthaxe n'est pas correct, il lui affiche une page HTML contenant une description "**BAD REQUEST**"

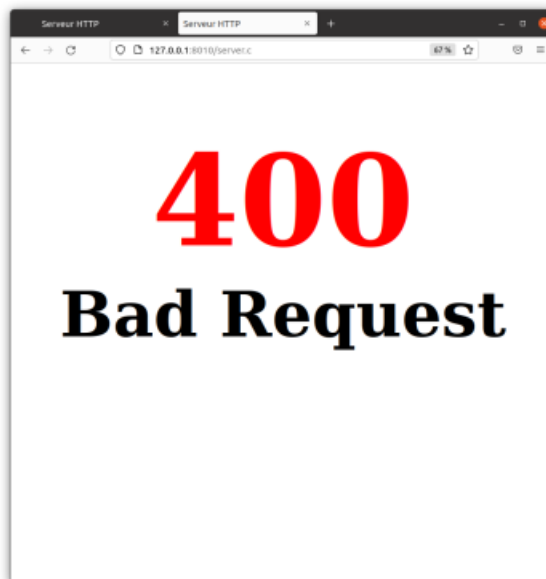


FIGURE 2.11 – Cas D'Erreur 400

3. [L'erreur 501](#) : , le client effectue un autre type de requête, par exemple une requête POST ou HEAD, il lui affiche une page HTML contenant une description "NOT IMPLEMENTED"



FIGURE 2.12 – Cas D'Erreur 501

2.5 Etape 5 :

Le serveur gère aussi les fichiers images et textes et cela en basant sur l'extension du fichier demandé par le client et en répondant avec le type MIME qui correspond (en-tête Content-Type) la fonction qui le permet en faisant les conditions suivantes :

```
if (strcmp(ext, "html") == 0 || strcmp(ext, "htm") == 0) { return "text/html"; }
if (strcmp(ext, "jpeg") == 0 || strcmp(ext, "jpg") == 0) { return "image/jpg"; }
if (strcmp(ext, "css") == 0) { return "text/css"; }
if (strcmp(ext, "js") == 0) { return "application/javascript"; }
if (strcmp(ext, "json") == 0) { return "application/json"; }
if (strcmp(ext, "txt") == 0) { return "text/plain"; }
if (strcmp(ext, "gif") == 0) { return "image/gif"; }
if (strcmp(ext, "png") == 0) { return "image/png"; }
```

FIGURE 2.13 – Fonction Mime

est bien " MIME "

Conclusion

Tout au long de ces 2 chapitres qu'on a traité, on a pu voir qu'est-ce qu'un Serveur HTTP 1.0, Les fonctionnalités utilisées un petit historique sur ce dernier.

Nous avons ensuite passé à l'étape de réalisation du projet ou on avait affaire à des questions/Reponses . On a parlé de la creation d'un serveur HTTP 1.0 multithreads , en envoyant des heads et des affichages

Après, nous avons vu les cas d'erreurs 400, 404 et 501 et enfin la structure MIME (Multipurpose Internet Mail Extension) qui a role de définir des règles de codage pour les messages non ASCII..

Comme perspective, nous souhaitons que ses serveurs réseau seront étudiés par plusieurs développeurs en améliorant leurs fonctionnalités comme on dit "La preuve de la valeur d'un système informatique est son existence."