Conception d’un scanner réseau

1) Lancement du scanner

```
make all
sudo ./cmake-build-debug/serveurTCP
./cmake-build-debug/clientTCP
```

2) Format des requêtes entre client et serveur

Les requêtes sont uniquement des **chaînes de caractères**.
Pour envoyer des informations du client au serveur on envoie sous la forme :

- Client envoie : **chiffre|information**
- Serveur **lit le chiffre** et sait comment **traiter** l'information.

Codification des données :

- Pour le scan horizontal, le client envoie : **1**
- Pour le scan vertical, le client envoie : **2|adresse_ip**
- Pour l'aide, le client envoie : **3**
- Pour quitter, le client envoie : **4** (signal de fin de connexion)

Pour envoyer des informations du serveur au client on envoie sous la forme :

- Serveur envoie : donnée (pas de mise en forme particulière)
- Client lit la donnée et l'affiche.

3) Scanner horizontal

Ce scanner horizontal effectue un **balayage des adresses IP actives** dans le réseau local. Il scanne toutes les adresses dans la **plage définie par l'adresse du réseau** et **l'adresse de diffusion**(broadcast) pour chaque interface réseau active qui n'est pas de type loopback. Le scan s'appuie sur des **paquets ICMP** pour déterminer si une hôte est actif ou non.

Détecter les adresses réseau (coté client)

Le programme commence par **récupérer la liste des interfaces** réseau disponibles. Pour chaque interface de type IPv4 qui n'est pas une interface loopback :

L'**adresse IP** et le **masque de sous-réseau** de l'interface sont affichés.

L'adresse réseau et l'adresse de diffusion sont **calculées à partir de l'adresse IP de l'interface et du masque de sous-réseau**.

Balayage des adresses (coté client)

Le client exécute ensuite le **scan sur la plage d'adresses définie** entre l'adresse du réseau et l'adresse de diffusion. Il utilise la fonction `is_host_active` pour envoyer des **requêtes ICMP**(type Echo) et attend une **réponse pour chaque adresse IP** dans la plage. Chaque adresse IP répondant positivement est considérée comme **active**.

Pour chaque adresse détectée comme active, l'information est **affichée sur la console**.

En cas d'échec de réponse (**timeout**), l'adresse est **considérée comme inactive**.

Des **contrôles d'erreur sont effectués** pour s'assurer que les interfaces réseau peuvent être **correctement analysées** et que les adresses IP peuvent être **balayées sans interruption**. Les erreurs potentielles dans la création de sockets ou pendant l'envoi et la réception de paquets ICMP sont gérées et des **messages appropriés sont affichés**.

Fermeture de la session de scan

Une fois le scan terminé, toutes les ressources réseau utilisées, telles que les sockets et les structures d'information d'interface, sont **libérées pour éviter les fuites de mémoire** et garantir que le programme se termine proprement.

4) Scanner vertical

Ce scanner vertical va balayer **les ports pour une adresse IP donnée**. Ce scan consiste à **vérifier l'état de chaque port** dans une plage donnée (par défaut de 1 à 1024) pour une adresse IP spécifique.

Lecture et vérification de l'adresse IP (coté client)

Tout d'abord, le programme vérifie le **format de l'adresse IP** entrée par l'utilisateur. Cette vérification est effectuée en essayant de convertir l'adresse IP en un format numérique. Si la **conversion réussit**, cela signifie que l'adresse IP est au format `x.x.x.x` avec `x` compris entre 0 et 255. Si l'adresse est valide, le programme envoie au serveur la commande `2|xxx.xxx.xxx.xxx`. Si l'adresse **n'est pas valide**, le programme demande à l'utilisateur d'entrer une nouvelle adresse.

Balayage des ports (coté serveur)

Le serveur **traite ensuite la demande** en appelant la fonction `scan_vertical`. Cette fonction prend en paramètres **l'adresse IP à scanner, le port de départ et le port de fin de la plage à scanner, ainsi qu'un buffer pour stocker les résultats du scan**. Par défaut, la fonction scanne les ports de 1 à 1024.

La fonction `scan_vertical` effectue ensuite le scan des ports en utilisant une boucle qui itère **sur chaque port de la plage à scanner**. Pour chaque port, la fonction crée un **socket**, le configure en **mode non-bloquant**, et essaye de se connecter au serveur à l'adresse IP et au port spécifiés. Si la **connexion échoue**, le port est considéré comme **fermé**. Si la **connexion réussit**, le port est considéré comme **ouvert** et les informations sur le port ouvert sont ajoutées à la chaîne de caractères `buffer`.

Gestion des délais d'attente (coté serveur)

La fonction `scan_vertical` gère également le **délai d'attente du scan**. Si le délai est trop long, cela signifie généralement que l'adresse IP est invalide et la fonction arrête le scan et retourne un message de timeout. Si le délai n'est pas dépassé, la fonction poursuit le scan des ports.

Réception et affichage des données (coté client)

La liste des ports ouverts est envoyée au client et celui ci l'affiche telle que reçue pour l'utilisateur. Ensuite, l'utilisateur peut choisir dans le menu l'action à réaliser.

5) Exemple de fonctionnement :

On lance le serveur avec `sudo` de manière à pouvoir faire le scanner horizontal :

```
make all
sudo ./cmake-build-debug/serveurTCP
./cmake-build-debug/clientTCP
```

Dans cet exemple, nous avons effectué un scan horizontal, suivi d'un scan vertical de l'adresse `172.20.10.1` (en partage de connexion).

Scan horizontal : deux adresses actives ont été détectée.
Scan vertical : deux ports sont ouverts à cette adresse.

```
Menu:
1. Scan horizontal
2. Scan vertical
3. Help
4. Quitter
1
Serveur : Interface: en0
IP Address: 172.20.10.3
Subnet Mask: 255.255.255.240
Network Address: 172.20.10.0
Broadcast Address: 172.20.10.15
Scanning IP: 172.20.10.15
Host 172.20.10.1 is active.
Host 172.20.10.2 is inactive.
Host 172.20.10.3 is active.
Host 172.20.10.4 is inactive.
Host 172.20.10.5 is inactive.
Host 172.20.10.6 is inactive.
Host 172.20.10.7 is inactive.
Host 172.20.10.8 is inactive.
Host 172.20.10.9 is inactive.
Host 172.20.10.10 is inactive.
Host 172.20.10.11 is inactive.
Host 172.20.10.12 is inactive.
Host 172.20.10.13 is inactive.
Host 172.20.10.14 is inactive.
Scan horizontal terminé.

Menu:
1. Scan horizontal
2. Scan vertical
3. Help
4. Quitter
2
Entrez l'adresse IP : 172.20.10.1
Serveur : Port 21
Port 53
Timeout

Menu:
1. Scan horizontal
2. Scan vertical
3. Help
4. Quitter
```