Etape 5

1/ Protocole

- Le protocole utilisé pour le client/serveur est le TCP (layer Transport) car SOCK_STREAM est utilisé. Si on veut utiliser de l'UDP on utilisera SOCK_DGRAM.

- TCP : Le protocole TCP est un protocole dit connecté. Il contrôle si le paquet est arrivé à destination si ce n'est pas le cas il le renvoie.

- UDP : À la différence de TCP, UDP est un protocole en mode non connecté, il ne vérifie pas si le paquet est arrivé à destination.

2/ Format

- En ce qui concerne le format des messages evoyés, tous ces messages échangés entre le serveur et les différents clients seront stockés dans la chaîne de caractères (char) buffer que l'on a initialisé avec une taille de 2048, ce qui sera largement suffisant pour envoyer même un long message.

- Lors de l'envoi, le buffer va calculer le nombre de caractère insérés par l'utilsiateur afin d'en déduire la taille de la chaîne de caractères pour ne pas qu'elle fasse tout le temps 2048 en taille.

- On retrouve aussi du int pour le port ou la validation de socket, une structure pour les paramètres et la connexion de sockets ou encore une variable pthread_t qui va gérer les différents threads.

3/ Diagramme UML 1 serveur 2 clients

Voir fichier etape5/diagramme.png

4/ Algorithme simplifié du serveur

```
début
	ajout des librairies socket et pthread + autres librairies pour le fonctionnement du programme
	déclaration de la taille du buffer à 2048
	déclaration de la variable sub à 0
	déclaration du nombre max de client à 100
	déclaration des paramètres du socket
		AF_INET <- IPv4
		SOCK_STREAM <- TCP
		PORT <- 4444
		IP <- 127.0.0.1 (loopback)
	si mauvaise initialiation, mauvaise liaison client/serveur et mauvaise écoute du socket
		afficher erreur et quitter
	sinon
		continuer le programme
	tant que 1
	quand un client se connecte au serveur, on ajoute 1 au nombre max de client puis on crée un thread pour ce client.
		limite utilisation du CPU à 1
		si réception de username client stocké dans le buffer
			name <- username client
```

```
                envoyer à tous les clients connectés que le client est connecté
            sinon
                afficher erreur
            formatage du buffer
            tant que 1
                initialisation de la variable recep
                si réception  message stocké dans le buffer > 0
                    si longueur du buffer > 0
                        si buffer = "coucou"
                            buffer <- Bonjour
                            envoi à tous les clients le message contenu dans le
buffer (cad Bonjour)
                        sinon
                            envoi du message contenu dans le buffer au client qui a
envoyé le message
                        fin si
                sinon si réception message stocké dans le buffer == 0 ou si buffer =
"exit")
                    buffer <- <nom_client> a quitté le serveur
                    envoi à tous les clients le message contenu dans le buffer
                    on enlève 1 au nombre max de clients

                sinon si buffer="sub <username_clientX>"
                    créer un thread de subscription
                    sub <- +1
                    envoyer les messages émis par username_clientX au clients abonnés
quand il en envoie un
                sinon si buffer="unsub <username_clientX>"
                    fermer le thread de subscription
                    sub <- -1
                    ne plus envoyer les messages émis par username_clientX au clients
abonnés quand il en envoie un
                sinon
                    afficher erreur

                fermeture du thread et du socket
            fin tant que
        fin tant que
fin
```

5/ Algorithme simplifié du client (peut être exécuté plusieurs fois tant qu'on n'atteind pas le nombre max de client connectés au serveur)

```
début
        ajout des librairies socket et pthread + autres librairies pour le fonctionnement du
programme
        déclaration de la taille du buffer à 2048
        déclaration de la variable sub à 0
        déclaration du nombre max de client à 100
        saisie utilisateur de username client
        déclaration des paramètres du socket
            AF_INET <- IPv4
            SOCK_STREAM <- TCP
            PORT <- 4444
            IP <- 127.0.0.1 (loopback)
        si mauvaise connexion au serveur
            afficher erreur et quitter
        sinon
            continuer le programme
        fin si
        envoi de username client au serveur
```

```
	tant que 1
		quand le client envoi un message, on crée un thread pour cet envoi.
		quand le client reçoit un message, on crée un thread pour cette réception.
		si buffer <- saisie de "exit"
			envoi de buffer au serveur
			arrêter le programme
		sinon si buffer <- saisie de "sub <username_clientX>"
			créer un thread de subscription
			sub <- +1
			recevoir les messages émis par username_clientX au clients abonnés quand il
en envoie un
		sinon si buffer <- saisie de "unsub <username_clientX>"
			fermer le thread de subscription
			sub <- -1
			ne plus recevoir les messages émis par username_clientX aux clients abonnés
quand il en envoie un
	fin tant que
fin
```

6/ Sitographie

- https://broux.developpez.com/articles/c/sockets