**Documentation for Client-Server Chat Application**

Ce document fournit une documentation pour une application de chat client-serveur simple écrite en C. L'application facilite la communication entre un client et un serveur via une connexion TCP/IP. Le client et le serveur communiquent à l'aide de sockets, et l'utilisateur interagit avec le client via une interface console.

**Utilisation**

Pour utiliser l'application de chat, les arguments de ligne de commande suivants sont nécessaires :

./client <server\_ip> <server\_port>

* <server\_ip> : L'adresse IP du serveur.
* <server\_port> : Le numéro de port sur lequel le serveur écoute.

Exemple :

./client 127.0.0.1 8080

**Implémentation**

**Dépendances**

* Bibliothèques standard C : stdio.h, stdlib.h, string.h, unistd.h
* Bibliothèques réseau : arpa/inet.h
* Bibliothèques de threads : pthread.h
* Bibliothèques de contrôle de fichiers : fcntl.h
* Bibliothèques d'appels système : sys/types.h, sys/stat.h

**Constantes**

BUFFER\_SIZE : Définit la taille du tampon de message pour l'envoi et la réception de messages.

**Fonctions**

**main**

La fonction principale est le point d'entrée de l'application client. Elle effectue les tâches suivantes :

1. Analyse les arguments de ligne de commande pour obtenir l'adresse IP et le port du serveur.
2. Crée un socket pour la communication avec le serveur.
3. Configure l'adresse du serveur en utilisant struct sockaddr\_in.
4. Se connecte au serveur.
5. Invite l'utilisateur à saisir un nom d'utilisateur et l'envoie au serveur.
6. Crée deux threads :
   * userInputThread : Gère la saisie utilisateur et envoie des messages au serveur.
   * serverListenerThread : Écoute les messages du serveur et les affiche.

**userInputThread**

Ce thread est responsable de la capture de la saisie utilisateur et de l'envoi de messages au serveur. Il s'exécute dans une boucle infinie, invitant l'utilisateur à saisir des messages et les envoyant au serveur.

**serverListenerThread**

Ce thread écoute les messages du serveur. Il s'exécute dans une boucle infinie, recevant des messages et les affichant. S'il y a une déconnexion du serveur, il avertit l'utilisateur et se termine.

**Interaction avec l'Utilisateur**

* Après la connexion au serveur, l'utilisateur est invité à saisir un nom d'utilisateur.
* L'utilisateur peut ensuite saisir des messages dans la console, qui sont envoyés au serveur.
* Les messages reçus du serveur sont affichés dans la console.

**Gestion des Erreurs**

L'application fournit des messages d'erreur et se termine correctement en cas d'échecs, tels que la création du socket ou les erreurs de connexion.

**Nettoyage**

L'application ferme le socket avant de se terminer.