****

**Ceg 3585**

**Application de bavardage**

****

3 fevrier, 2023

universite d’Ottawa

**Membres du groupe**

Waffo Tony Adams (300310088)

**Assistants d’enseignements**

Félix Nadon

Reda Rahmoune

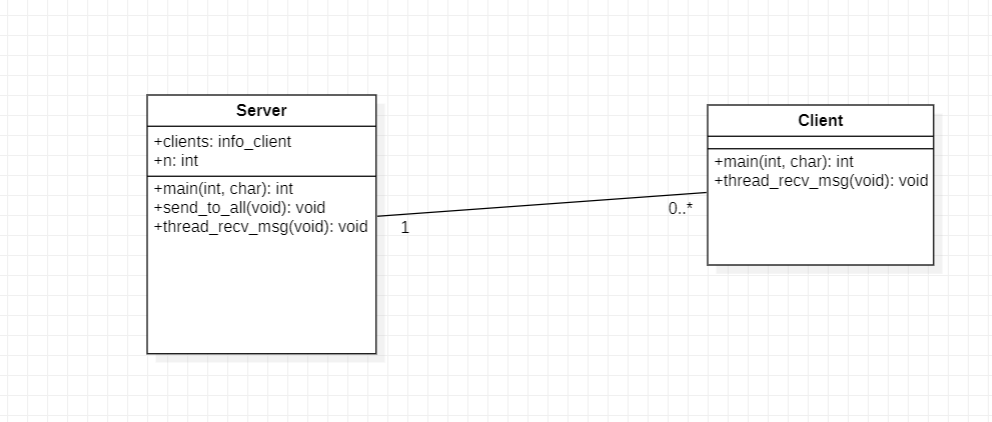
1. **Description brève du but et de la théorie du problème**

Ce tout premier laboratoire du cours de CEG3585 a pour but principale de concevoir une application de bavardage entre un client et un serveur dans une connexion TCP/IP. La théorie utilisée dans le cadre de ce laboratoire est basée sur l’implémentation des sockets. A cet effet, des sockets devront d’abord être initialisés pour le serveur et le client en question. Ainsi, après la volonté du serveur d’accepter les connexions, il pourra entrer en connexion avec le client à partir du port de ce dernier. A ce niveau, les messages entre les deux entités peuvent être transmis jusqu’à ce que la connexion soit arrêtée. La particularité de ce laboratoire serait de réaliser une programmation où le message est envoyé à une seule personne et non à toutes celles connectées dans le réseau comme il a déjà été fourni initialement. Pour ce faire, nous utiliserons la programmation C.

1. **Explication brève de votre algorithme de solution**

En effet, la majeure partie du travail avait déjà été faite dans le fichier fourni qui implémentaient déjà les fonctions d’envoie et de réception par l’utilisation des threads. Mais à la suite des enjeux énumérés plus haut auxquels nous devons faire face, la solution consiste alors à assigner un nom d’utilisateur à chaque client de telle sorte que lors de l’envoie du message par le client, le message proprement dit puisse parcourir la liste d’utilisateur dans le réseau et le transmettre uniquement au client désigné. Pour ce faire nous avons d’abord une première étape qui consiste à envoyer son nom d’utilisateur lors de la connexion au serveur, de telle sorte que ce dernier l’enregistre dans un tableau contenant son numéro de port et son prénom; raison pour laquelle on utilise un tableau de structure (struct info\_client clients[100] ). Ensuite, lorsque la connexion est établie avec un client, ce dernier se doit d’envoyer le message contenant le nom du récepteur suivi du message en question (exemple : receiverName- message). A noter qu’il n’y a pas d’espace entre le nom du récepteur et le trait d’union. A ce niveau, le serveur reçoit le message et utilise des pointeurs ainsi que des fonctions de la librairie <string.h> telles que strstr(), strcpy()et strncpy() pour extraire le nom du récepteur dans la variable « receiver » et le message dans la variable « msg ». Des lors, la fonction send\_to\_all implémente une boucle qui traverse la liste des clients du réseau et qui envoie le message au client qui est bel et bien différent de l’entité émettrice et qui correspond au nom spécifié dans le message de l’émetteur.

1. **Document de conception en utilisant les diagrammes UML**



1. **Captures d’écrans de la démonstration de l’application**

Ci-dessous correspondent les captures d’écran de la connexion des clients au serveur, du chat entre les participants puis de la déconnexion.

Une image contenant texte, noir, équipement électronique, ordinateur

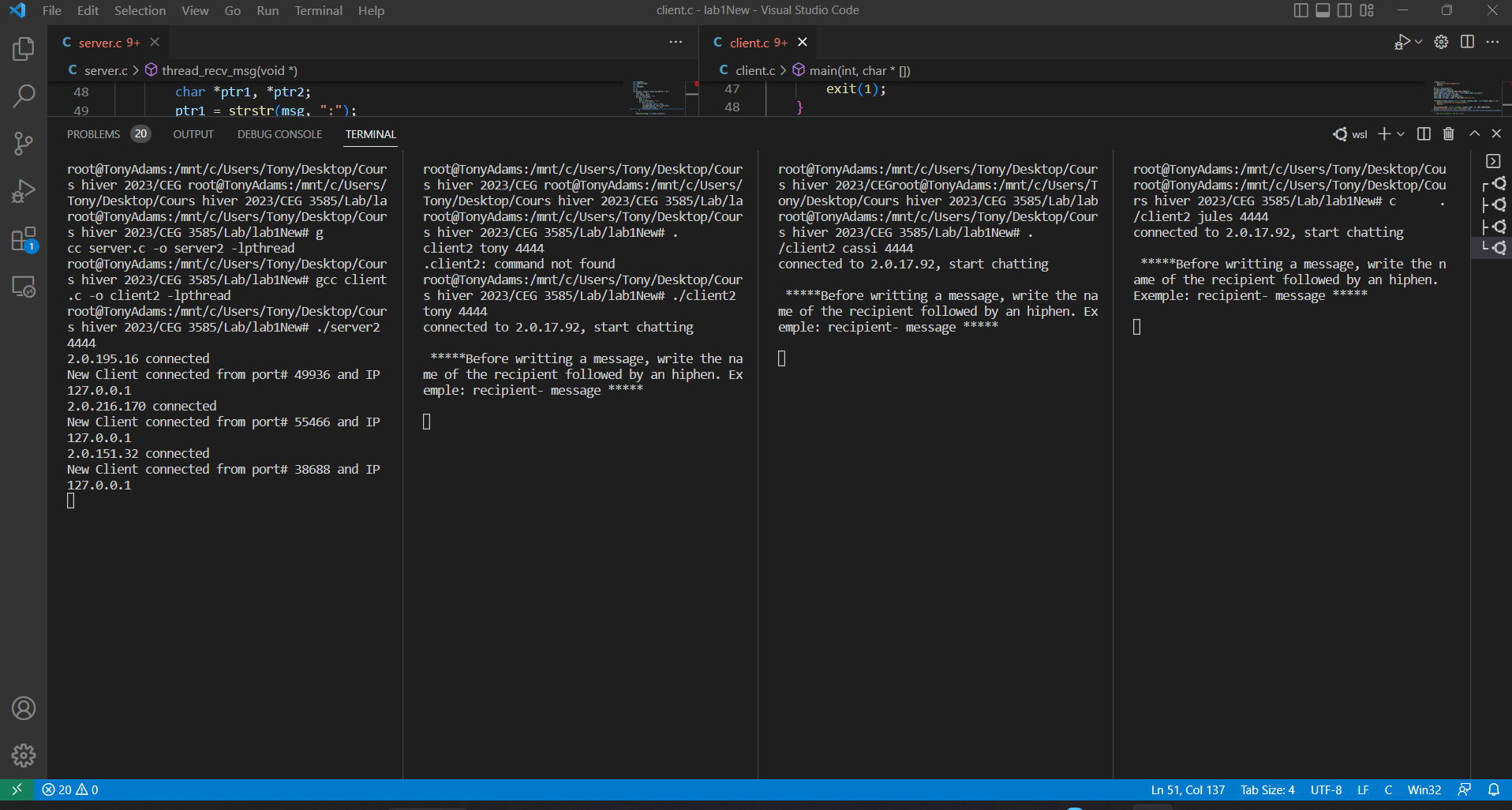
Description générée automatiquement

Figure 2 Chat de discussion entre clients

Figure 1 Connexion des clients au serveur

1. **Une image contenant texte

   Description générée automatiquementDiscussion**

Figure 3 Déconnexion des clients du réseau

La communication de nos jours est quelque chose dont nous ne pouvons-nous en passer. La programmation des sockets nous permet donc d’accomplir cette tâche. Encore plus important, l’utilisation de thread est nécessaire, voire primordiale dans la mesure où on n’a pas besoin qu’un client finisse son processus avant que le prochain client soit pris en compte. Le serveur peut alors gérer plusieurs clients simultanément, permettant alors d’augmenter les performances du réseau. Bien que le code implémenté donne des résultats attendus, il a été difficile de trouver le bon algorithme qui détecte le nom du récepteur et le message proprement dit à partir de l’input du client. Toutefois, après quelques recherches, une solution a été trouvée.

1. **Conclusion**

En définitive, ce laboratoire m’a largement instruit sur le processus utilisé entre des entités d’un réseau pour établir des communications. De plus, j’ai non seulement appris de nouvelles choses sur la programmation des sockets en langage C que je n’avais jamais utilisé au paravent, mais j’ai aussi approfondi mes connaissances en ce qui concerne l’utilisation des threads et leur importance.