19/11/2023

Université Mohamed Khider Biskra
Faculté SESNV
Département d'informatique

Niveau: M1 **Module: SD** **Année: 2023-2024**

TP n°3. **Clients_Serveur / Threads.**

3 séances (2 + 1 pour consultation)

Dans ce TP, on reprend le même travail demandé dans le TP2 en prenant en compte les quatre exercices : l'application (TCP) réalise une conversation (chat : échange de messages variables) entre le serveur et les clients.

Lorsque le serveur reçoit un message de la part d'un client, il calcule et affiche le nombre de mots. Le chat se termine par l'envoi d'un message « quit » par l'une des deux machines : soit par le serveur, soit par le client. Si un client envoie un message « quit », il quitte le groupe et le serveur ferme la session uniquement avec lui. Ce dernier cas est similaire à la décision du serveur de mettre fin à la connexion avec un client donné. Cependant, lorsque le serveur diffuse un « quit » à tous les clients connectés, il supprime l'ensemble du groupe.

L'exécution de l'application de TP2 avec plusieurs clients montre que le serveur ne peut communiquer avec un client que lorsqu'il termine la connexion avec un client précédent (précédence en termes de connexion avec le serveur). Ce problème se pose à cause de l'effet bloquant de la fonction accept(…) qui mène à une exécution séquentielle des communications.

Pour résoudre ce problème et garantir des communications parallèles au niveau du serveur avec les différents clients, on introduit le concept de threads. L'idée est de créer et d'associer un thread pour chaque client connecté au serveur.

Lorsque le serveur décide de clôturer la session et de supprimer le groupe, il envoie « quit » à tous les clients connectés et n'accepte plus de nouvelles connexions. Après la fermeture, il affiche le nombre de mots de tous les messages reçus (la somme de tous les nombres calculés lors des différentes communications).

Remarques:

1. Avant d'utiliser les threads et pour bien comprendre le problème posé par la fonction accept(…), vous devez d'abord exécuter un ancien code avec plusieurs clients.
2. Eviter l'utilisation de la fonction fork (**n'est pas accepté**)
3. Faire attention dans le cas de l'utilisation d'une variable commune pour calculer le nombre des mots pour chaque message car il s'agit d'une variable partagée nécessitant une synchronisation entre les threads.