Projet

Client-Serveur avec Graceful Shutdown

Le projet est à rendre avant le dimanche 16 mai à 23h59.

Le but de ce projet est d'écrire un client/serveur avec un système de graceful shutdown pour le serveur. Ce projet comporte deux parties : une partie imposée et une partie libre. Nous vous donnons des contraintes à respecter pour la partie client/serveur, mais libre à vous de décider ce que le client et le serveur échangent. Vous pouvez par exemple décider de faire :

- Une plateforme de streaming (audio ou vidéo), où les clients demandent un fichier au serveur et celui-ci leur envoie.
- Un jeu en ligne, où le serveur envoie au clients l'état du jeu, et les clients envoient au serveur leurs actions.
- ce que vous voulez qui réponde aux impératifs décrits ci-dessous.

1 Le serveur

- Le serveur est une application multi-tâches :
  1. une tâche reçoit les connexions des utilisateurs. A chaque connexion, une nouvelle tâche est créée, qui sera chargée de gérer la communication avec un utilisateur spécifique.
  2. les tâches de communications avec les utilisateurs permettent d'échanger des informations via le réseau, jusqu'à ce que la tâche soit terminée, à la demande de l'utilisateur ou côté serveur.
- Le serveur doit pouvoir faire un graceful shutdown. Dans ce cas, la tâche principale ne prends plus de connexion et attends que les autres tâches soient terminées afin de libérer proprement l'espace mémoire. Les tâches utilisateurs informent les clients que la connexion doit être interrompue. Lors du redémarrage du serveur, le client peut se connecter et demander la reprise de la tâche là ou elle s'était arrêtée. La sauvegarde des informations nécessaires peut être effectuée par le serveur ou le client, suivant l'application que vous choisissez de développer. **Vous devez en revanche impérativement implanter un graceful shutdown avec un système de sauvegarde et de chargement**.

2 Le client

- Le client est aussi une application multi-tâches :
  1. une tâche (au minimum, adapter selon vos besoins) aura pour objectif de communiquer avec le serveur.
  2. une tâche gérera les interactions au clavier avec l'utilisateur.
- Le client peut se terminer à la demande de l'utilisateur, si celui-ci l'indique via une commande au clavier (et non un signal). Le client peut aussi se terminer si le serveur envoie un message indiquant la fermeture de la connexion.
- Comme indiqué plus haut, il est possible que le client doive faire des sauvegardes et des chargements, cela dépend de votre application.

3 Impératifs pour le rendu

Voici les consignes pour le rendu :

1. le projet est effectué seul ou en binôme, le binôme étant fortement recommandé étant donné le travail à effectuer. La méthode de notation ne changera pas selon que vous soyez seul ou à deux. En cas de binôme, vous devez **impérativement être avec quelqu'un de votre groupe de TP**, c'est à dire que vous avez le même correcteur.
2. Étant donné l'UE projet libre vous demande de faire un projet sur de plus grands groupes, **il n'est (malheureusement) pas possible de faire le même projet pour les deux U.E.**.
3. le rendu se fera via une archive, nommée d'après vos noms et prénoms, aux formats `tar.gz`, `tgz` ou `zip`.
4. l'archive contiendra un répertoire, dont le nom sera le même que celui de l'archive (moins l'extension).
5. dans répertoire, vous placerez :
   — un repertoire `server`, contenant les fichiers sources du serveur.
   — un repertoire `client`, contenant les fichiers sources du client.
   — si besoin, un répertoire `commun`, contenant le code utilisé par le client **et** le serveur.
   — un fichier `RAPPORT.md`, dans lequel vous décrirez **succintement** votre projet.
   — un fichier `README.md`, dans lequel vous décrirez comment compiler et exécuter votre programme.
   — un fichier `Makefile`, qui servira à compiler votre projet.
6. Votre code sera commenté, comme vous avez appris à le faire dans la matière programmation orientée objet. Dans l'en-tête de chaque fichier, vous ajouterez le champs `@author` pour indiquer qui est l'auteur de la fonction.
7. Un programme qui ne compile pas vaut 0, il vaut mieux rendre un programme qui ne répond pas à toutes les questions qu'un programme qui ne compile pas. N'hésitez donc pas à mettre les zones problématiques en commentaires.
8. Tout **soupçon** de fraude sera immédiatement sanctionné d'un 0 pour **l'ensemble** des personnes impliquées, voire d'un conseil de discipline. Toute justification du type « on en a discuté ensemble » ne sera pas admise. Discuter n'est pas interdit. Rendre des devoirs dont le code est identique l'est. Changer le nom des variables est un très mauvais moyen de camoufler une fraude.

4 Comment le projet sera noté

Une partie des points concernera :
— le respect des consignes indiquées plus haut.
— la qualité du rapport.
— la qualité des commentaires.
— la qualité du code (indentation, nom des variables, clarté, nous utiliserons valgrind pour tester l'absence d'erreurs). Une fonction ne doit pas faire plus de 30 lignes (ce qui est déjà très long).
— les applications multi-tâches (lancement des tâches, fermetures, libération de l'espace mémoire, ...), côté serveur et côté client.
— le graceful shutdown (signaux, sauvegarde, chargement)
— le projet que vous aurez imaginé.