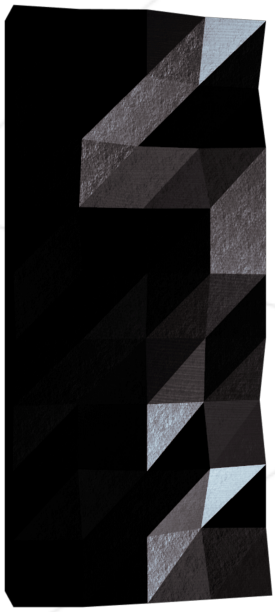

IRC

Internet Relay Chat

Résumé: L'objectif de ce projet est de vous faire reproduire le fonctionnement de la partie serveur d'un IRC. Pour cela vous devrez suivre des RFC et vous interagirez avec de vrai clients IRC. Internet fonctionne grâce à de nombreux standards et protocoles pour permettre un interopérabilité entre les machines connectées. C'est toujours intéressant de connaitre ce genre de chose.

Table des matières

Chapitre I

Règles communes

- Votre programme ne doit en aucun cas crash (même si il n'y a plus de mémoire) ni s'arrêter de manière inattendue mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Si le projet le demande, vous devez rendre un Makefile qui compilera vos sources pour créer la sortie demandée, en utilisant les flags `-Wall`, `-Wextra` et `-Werror`. Votre Makefile ne doit pas relink.
- Si le projet demande un Makefile, votre Makefile doit au minimum contenir les règles `$(NAME)`, `all`, `clean`, `fclean` et `re`.
- Pour rendre des bonus, vous devez inclure une règle `bonus` à votre Makefile qui ajoutera les divers headers, librairies ou fonctions qui ne sont pas autorisées dans la partie principale du projet. Les bonus doivent être dans des fichiers séparés, identifiables par leur nom spécifique. L'évaluation de la partie obligatoire et de la partie bonus sont faites séparément.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs. C'est notamment intéressant durant la soutenance où vous pouvez utiliser vos tests, les tests de l'autre étudiant, ou tout autre test auquel vous pensez.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

Chapitre II

Introduction

Internet Relay Chat ou IRC est un protocole de communication textuel sur Internet. Il sert à la communication instantanée principalement sous la forme de discussions en groupe par l'intermédiaire de canaux de discussion, mais peut aussi être utilisé pour de la communication directe entre deux personnes.

Un client IRC se connecte à un serveur IRC pour accéder à un `channel` particulier. Les serveurs IRC sont connectés entre eux pour proposer un réseau global avec des `channel` uniques.

Chapitre III

Partie obligatoire

Nom du programme	`ircserv`
Fichiers de rendu	
Makefile	`Oui`
Arguments	
Fonctions externes autorisées	`socket, open, close, setsockopt, getsockname, getprotobyname, gethostbyname, getaddrinfo, bind, connect, listen, accept, htons, htonl, ntohs, ntohl, inet_addr, inet_ntoa, send, recv, exit, signal, lseek, fstat, read, write, fcntl, select, FD_CLR, FD_COPY, FD_ISSET, FD_SET, FD_ZERO`
Libft autorisée	
Description	`Ecrire un serveur IRC en C++`

- Votre serveur IRC doit etre en C++
- Vous devez suivre les RFC 1459, 2810, 2811, 2812, 2813 et 7194.
- La communication entre le client et le serveur se fera en TCP/IP (v4) ou (v6).
- Vous `n'avez pas` à coder de client.
- Vous `devrez gérer` la communication de serveur à serveur.
- Vous pouvez inclure et utiliser tout `iostream - string - vector - list - queue - stack - map - algorithm`
- Vous pouvez utiliser une librairie cryptographique pour gérer les communication TLS.
- Vous avez l'autorisation d'utiliser d'autres fonctions dans le cadre de vos bonus, à condition que leur utilisation soit dûment justifiée lors de votre correction. Soyez malins.
- Votre binaire devra etre appelé comme ceci :

```
./ircserv [host:port_network:password_network] <port> <password>
```

  - `host` correspond au hostname du serveur d'un réseau IRC à rejoindre
  - `port_network` correspond au port du serveur `host` sur lequel se connecter
  - `password_network` correspond au mot de passe nécessaire pour se connecter au serveur `host`
  - `port` correspond au numéro de port de votre serveur sur lequel il acceptera les connections entrantes
  - `password` est le mot de passe nécessaire aux clients et serveurs pour se connecter sur votre serveur
  - Si `host`, `port_network` et `password_network` ne sont pas donnés, vous devrez `créer un nouveau réseau IRC`.

- Le serveur doit supporter plusieurs clients simultanément, de même que la connection éventuelle au serveur, et ne jammais bloquer. fork est interdit, et toutes les opérations d'entrées-sorties doivent être non bloquantes, vous devez utiliser un unique `select` pour tout (read, write, mais aussi listen, ...).

Nous vous laissons utiliser fcntl parce que MacOS X n'implémente pas write de la même façon que les autres Unix.
Vous devez utiliser des FD non bloquants pour obtenir un résultat similaire aux autres OS.

Comme vous utilisez des FD non bloquants, vous pourriez utiliser les fonctions read/recv ou write/send sans select, et votre serveur serait non bloquant. Ne le faites pas. Et cela consommerait des ressources système pour rien.
A nouveau, utiliser read/recv ou write/send sur n'importe quel FD sans vérifier la possibilité avec select vous menera directement à la note de 0 et la fin de la soutenance.

Vous ne pouvez utiliser fcntl que de la façon suivante : fcntl(fd, F_SETFL, O_NONBLOCK);
Tout autre flag est interdit.

- Il vous appartient de produire un code propre, vérifiant absolument toutes les erreurs et tous les cas pouvant amener des problèmes (envoi ou réception partielle de commandes, faible bande passante...).
- Pour vérifier que votre serveur utilise correctement tout ce qui vient d'être dit, un premier test est d'utiliser `nc` avec Control+D pour envoyer des parties de commandes :

```
\$> nc 127.0.0.1 6667
com^Dman^Dd
\$>
```

Ceci aura pour effet d'envoyer d'abord les lettres `com` puis `man` puis `d\n`. Il vous faudra donc agréger les paquets afin de recréer la commande `command` pour pouvoir la traiter.

- Il existe plusieurs clients IRC qui peuvent être utilisés pour vos tests et qui seront utilisés durant la soutenance.
- Un serveur IRC public sera également utilisé durant la soutenance pour tester les connections entre serveurs.

Chapitre IV

Partie bonus

Libre à vous de rajouter ce qui vous semble interessant pour rendre votre IRC le plus proche de l'IRC standard, voici quelque exemple de bonus.

- Un client IRC.
- Un fichier de configuration, avec les modifications des paramètres en ligne de commande que cela implique.
- Les cas indiqués comme non obligatoires dans la RFC.
- Une interface graphique.
- La gestion d'envoi de fichier.
- Creation d'un BOT.
- Connection fonctionnelle avec un vrai réseau IRC public.