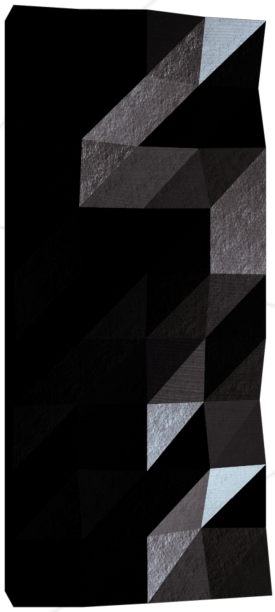

ft_irc

Internet Relay Chat

Résumé:
L'objectif de ce projet est de reproduire le fonctionnement d'un serveur IRC.
Vous utiliserez un vrai client IRC afin de vous connecter à votre serveur et ainsi de le tester.
Internet fonctionne grâce à de nombreux standards et protocoles pour permettre une interopérabilité entre les machines connectées. Il est toujours intéressant de connaître ce genre de chose.

Version: 6.1

Table des matières

I Introduction **2**

II Consignes générales **3**

III Partie obligatoire **4**

III.1 Prérequis 6

III.2 Pour MacOS seulement 7

III.3 Exemple de test 7

IV Partie bonus **8**

V Rendu et peer-evaluation **9**

Chapitre I

Introduction

Internet Relay Chat 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.

Les clients IRC se connectent à des serveurs IRC afin d'accéder à des canaux. Les serveurs IRC sont connectés entre eux afin de créer des réseaux.

Chapitre II

Consignes générales

- Votre programme ne doit en aucun cas crash (même si vous êtes à court de mémoire) ni s'arrêter de manière inattendue sauf dans le cas d'un comportement indéfini.
  Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez `0`.
- Vous devez rendre un `Makefile` qui compilera vos fichiers sources. Il ne doit pas `relink`.
- Votre `Makefile` doit contenir au minimum les règles suivantes :
  `$(NAME)`, `all`, `clean`, `fclean` et `re`.
- Compilez votre code avec `c++` et les flags `-Wall -Wextra -Werror`
- Vous devez vous conformer à la norme **C++ 98**. Par conséquent, votre code doit compiler si vous ajoutez le flag `-std=c++98`
- Dans votre travail, essayez d'utiliser en priorité des fonctionnalités `C++` (par exemple, préférez `<cstring>` à `<string.h>`). Vous pouvez utiliser des fonctions `C`, mais faites votre possible pour choisir la version `C++` quand vous le pouvez.
- Tout usage de bibliothèque externe ou de l'ensemble `Boost` est interdit.

Chapitre III

Partie obligatoire

Nom du programme	`ircserv`
Fichiers de rendu	`Makefile, *.{h, hpp}, *.cpp, *.tpp, *.ipp,` `un fichier de configuration optionnel`
Makefile	`NAME, all, clean, fclean, re`
Arguments	`port : Le port d'écoute` `password : Le mot de passe de connexion`
Fonctions externes autorisées	`Tout ce qui respecte la norme C++ 98.` `socket, close, setsockopt, getsockname, getprotobyname, gethostbyname, getaddrinfo, freeaddrinfo, bind, connect, listen, accept, htons, htonl, ntohs, ntohl, inet_addr, inet_ntoa, send, recv, signal, lseek, fstat, fcntl, poll (ou équivalent)`
Libft autorisée	`n/a`
Description	`Un serveur IRC en C++ 98`

Vous devez développer un serveur IRC en C++ 98.

Vous **n'avez pas** à développer un client.
Vous **ne devez pas** gérer la communication de serveur à serveur.

Votre binaire devra être appelé comme ceci :

```
./ircserv <port> <password>
```

- `port` : Le numéro du port sur lequel votre serveur acceptera les connexions entrantes.
- `password` : Le mot de passe permettant de s'identifier auprès de votre serveur IRC, et qui devra être fourni par tout client IRC souhaitant s'y connecter.

Bien que poll() soit mentionné dans le sujet et la grille d'évaluation, vous pouvez utiliser un équivalent tel que select(), kqueue(), ou epoll().

III.1 Prérequis

- Le serveur doit pouvoir gérer plusieurs clients simultanément sans jamais bloquer.
- Le `forking` est interdit. Toute les opérations entrées/sorties doivent être non bloquantes.
- Vous n'avez le droit qu'à **un seul** `poll()` (ou équivalent) pour gérer toutes ces opérations (read, write, mais aussi listen, etc.).

`Comme vous pouvez utiliser des FD en mode non bloquant, il est possible d'avoir un serveur non bloquant avec read/recv ou write/send tout en n'ayant pas recours à poll() (ou équivalent).`
`Mais cela consommerait des ressources système inutilement.`
`Ainsi, si vous essayez d'utiliser read/recv ou write/send avec n'importe quel FD sans utiliser poll() (ou équivalent), votre note sera de 0.`

- Il existe plusieurs clients IRC. Vous devez choisir l'un d'eux comme **référence**. Votre client de référence sera utilisé pour l'évaluation.
- Votre client de référence doit pouvoir se connecter à votre serveur sans rencontrer d'erreur.
- La communication entre le client et le serveur se fera en `TCP/IP` (v4 ou v6).
- Utiliser votre client de référence avec votre serveur devra être similaire à l'utiliser avec un serveur IRC officiel. Cependant, seules les fonctionnalités suivantes sont obligatoires :
  - Vous devez pouvoir vous authentifier, définir un `nickname`, un `username`, rejoindre un `channel`, envoyer et recevoir des messages privés, avec votre client de référence.
  - Tous les messages envoyés par un client dans un `channel` doivent être transmis à tous les clients ayant rejoint ce `channel`.
  - Vous devez avoir des `operators` et des utilisateurs basiques.
  - Vous devez donc implémenter les commandes spécifiques aux `operators`.
- Bien entendu, on attend de vous un code propre.

III.2 Pour MacOS seulement

Vu que MacOS n'implémente pas write() comme les autres Unix, vous pouvez utiliser fcntl().
Vous devez utiliser des descripteurs de fichier en mode non bloquant afin d'obtenir un résultat similaire à celui des autres Unix.

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

III.3 Exemple de test

Vérifiez absolument toutes les erreurs et tous les problèmes possibles (donnée partiellement reçue, bande passante faible, etc.).

Afin de vous assurer que votre serveur traite tout ce que vous lui envoyez, voici un test basique qui peut être fait avec `nc` :

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

Utilisez `ctrl+D` pour envoyer la commande en plusieurs parties : '`com`', puis '`man`', puis '`d`\n'.

Afin de traiter une commande, vous devez en premier lieu la reconstituer en concaténant les paquets reçus.

Chapitre IV

Partie bonus

Voici quelques fonctionnalités supplémentaires que vous pouvez ajouter à votre serveur afin qu'il soit encore plus proche d'un vrai serveur IRC :

- L'envoi de fichier.
- Un bot.

Les bonus ne seront évalués que si la partie obligatoire est PARFAITE. Par parfaite, nous entendons complète et sans aucun dysfonctionnement. Si vous n'avez pas réussi TOUS les points de la partie obligatoire, votre partie bonus ne sera pas prise en compte.

Chapitre V

Rendu et peer-evaluation

Rendez votre travail sur votre dépot `Git` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

Nous vous recommandons de créer des programmes de test pour votre projet bien ceux-ci **ne seront ni rendus ni notés**. Ces tests pourraient s'avérer particulièrement utiles durant votre soutenance, mais également si vous avez à évaluer un autre `ft_irc` un jour. En effet, vous êtes libre d'utiliser les tests de votre choix en évaluation.

`Votre client de référence sera utilisé pour l'évaluation.`