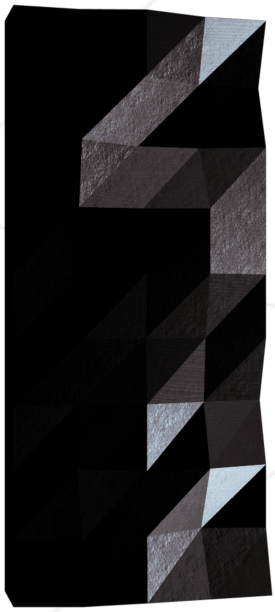

IRC

Internet Relay Chat

Résumé: L'objectif de ce projet est de reproduire le fonctionnement de la partie serveur d'un IRC en suivant une RFC

Table des matières

Chapitre I

Règles communes

- Vos fonctions de doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) 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.
- Toute mémoire allouée sur la heap doit être libéré lorsque c'est nécessaire. Aucun leak ne sera toléré.
- 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 une fichier `_bonus.{c/h}`. 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.
- 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 de un à un.

Chapitre III

Partie obligatoire

Votre IRC devra :

- Suivre le protocole de la RFC 2810, 2811, 2812 et 2813.
- La communication entre le client et le serveur se fera en TCP/IP (v4) ou (v6).
- Vous `n'avez pas` à gérer la partie client.
- Vous `n'avez pas` a gérer l'authentification client à serveur.
- Vous `devrez gérer` la communication de serveur à serveur.
- Dans le cadre de votre partie obligatoire, vous avez le droit d'utiliser les fonctions suivantes :
  - socket(2), open(2), close(2), setsockopt(2), getsockname(2)
  - getprotobyname(3), gethostbyname(3), getaddrinfo(3)
  - bind(2), connect(2), listen(2), accept(2)
  - htons(3), htonl(3), ntohs(3), ntohl(3)
  - inet_addr(3), inet_ntoa(3)
  - send(2), recv(2)
  - exit(3), signal(3)
  - lseek(2), fstat(2)
  - read(2), write(2)
  - select(2), FD_CLR, FD_COPY, FD_ISSET, FD_SET, FD_ZERO
  - et toutes les fonctions utilisées dans bircd.tar.gz qui ne seraient pas dans cette liste.
  - Utilisation de la STD ? ?
- `V`ous 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.

Chapitre IV

Serveur

- Votre binaire devra etre appelé comme ceci :

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

  - `host` correspond au hostname du serveur du réseau IRC à rejoindre
  - `port_network` correspond au port du serveur du réseau IRC à rejoindre
  - `port` correspond au numéro de port du serveur
  - Si host et `port_network` ne sont pas donnés, vous devrez `créer un nouveau réseau IRC`.
- Ne gérez pas les cas `5.7` et `5.8` du RFC 2813
- Le serveur doit supporter plusieurs clients simultanément par l'intermédiaire d'un select(2) en lecture ET écriture. Vous ne devez utiliser qu'un seul select(2) pour le serveur qui doit être `absolument non bloquant`.
- 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^Dde
\$>
```

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

`Un serveur d'exemple, bircd.tar.gz, vous est fourni avec ses sources pour vous permettre de bien démarrer. Vous pouvez utiliser et rendre ces sources, mais attention: cette base ne vous rapportera pas de points, et elle n'utilise select(2) qu'en lecture, vous devrez corriger ça.`

Chapitre V

Partie bonus

Ici libre a vous de rajouter ce qui vous semble interessant pour rendre votre IRC le plus proche du IRC actuelle, voici quelque exemple de bonus :

- La partie client d'IRC.
- Les cas que nous vous avons demandé de ne pas gérer.
- Une interface graphique.
- La gestion d'envoi de fichier.
- Creation d'un BOT.