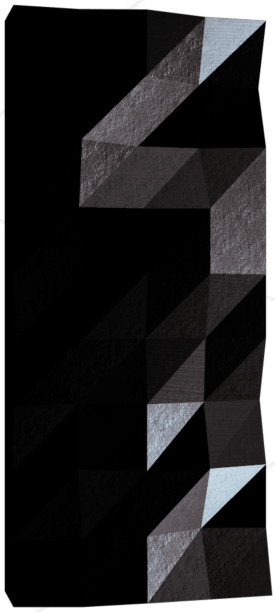

Projet Environnement UNIX II

ft_irc

42 staff staff@42.fr

Résumé: Ce projet consiste à implémenter un client et un serveur IRC permettant l'échange de messages en réseau TCP/IP

Table des matières

Chapitre I

Préambule

Voici ce que Wikipédia a à dire au sujet de l'Aligot :

L'aligot (oligot ou olicouót en aveyronnais est dérivé du verbe haligoter "déchirer, mettre en lambeaux") est une préparation culinaire de fromage spécial (tomme d'Aligot) et de purée de pommes de terre à la texture très élastique. C'est une spécialité rurale originaire de l'Aubrac (Aveyron, Cantal et la Lozère) qui s'est répandue dans le dernier quart du xxe siècle au reste du Massif central et plus largement en France notamment à travers l'exode rural des bougnats à Paris.

Préparation : L'aligot est une préparation faite avec une purée de pommes de terre à laquelle sont mélangés de la crème, du beurre et de la tomme d'Aligot. On ajoute un peu d'ail pilé ou haché finement. Cette purée doit être longuement travaillée afin d'obtenir une texture très élastique. L'aligot ne doit pas être confondu avec la truffade, plat traditionnel des monts du Cantal voisins.

Histoire : Ce mets a été à l'origine une soupe préparée avec du bouillon, des morceaux de pain et avec de la tomme fraîche. Les moines de l'Aubrac le servaient, dit-on, aux pèlerins qui traversaient ces montagnes sur la Via Podiensis, pour se rendre à Saint-Jacques-de-Compostelle. Le monastère sur la commune de Saint-Chély-d'Aubrac disparaît à la Révolution mais les agriculteurs qui ont récupéré les terres de l'abbaye perpétuent la tradition, il est notamment confectionné et consommé dans les étables-abris d'estives (les mazucs ou burons) tenues par des vachers-fromagers pour la fabrication du fromage de Laguiole. Au xixe siècle, une mauvaise récolte de blé remplace le pain par des pommes de terre. Au xxe siècle, les Auvergnats spécialisés dans la restauration ont rendu cette préparation populaire.

Préparation de l'aligot familial : L'aligot, autrefois plat de subsistance familial, est devenu pour certains un mets dit "de fête", relativement exceptionnel, sauf dans les restaurants se revendiquant rouergats ou auvergnats. Il est assez fréquemment accompagné de saucisse paysanne. Il ne doit pas être confondu avec la truffade, plat vraiment plus cantalien où des pommes de terre en lamelles sont sautées dans du lard et auxquelles on ajoute la tomme fraîche.

Chapitre II

Sujet - Partie obligatoire

Ce projet consiste à faire un client et un serveur IRC (Internet Relay Chat) qui permettent d'envoyer et recevoir des messages entre plusieurs personnes, au sein de groupes de discussion appelés "channel".

Vous êtes néammoins libre du choix du protocole à utiliser (vous n'êtes pas obligé de respecter les RFC définissant IRC, vous pouvez inventer votre propre protocol de chat). Vous devrez par contre quelque soit votre choix absolument obtenir une cohérence entre votre client et votre serveur. Ils doivent communiquer correctement ensemble.

La communication entre le client et le serveur se fera en TCP/IP (v4).

Vous devrez implémenter les fonctionnalités suivantes :

- `/nick <nickname>` : Implémenter la notion de nickname (un surnom, le login par défaut, maximum 9 charactères, comme dans la RFC)
- `/join <#chan>`, `/leave [#channel]` , etc... : Implémenter la notion de channel
- `/who` : Qui est loggué sur le channel ?
- `/msg <nick> <message>` : Envoi de messages à un utilisateur particulier sur le serveur
- Passage du code en mode non-bloquant : le code fourni est bloquant dans certains cas. Si un client ne répond plus, le serveur peut se bloquer après un cours délai. Attention, un seul Select dans votre code est autorisé.

`ATTENTION: cela n'a rien à voir avec des sockets non bloquantes qui elles sont interdites (donc pas de fcntl(s, O_NONBLOCK))`

Chapitre III

Serveur

Usage :

```
$> ./serveur <port>
```

où “port” correspond au numéro de port du serveur.

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.

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.

Votre code sera non seulement non bloquant, mais devra aussi utiliser des tampons tournants circulaires afin de sécuriser et optimiser l'envoi comme la réception des différentes commandes et réponses. Il vous apartient 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, une 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.

Chapitre IV

Client

Usage :

```
$> ./client <machine> <port>
```

où “machine” correspond au nom d’hôte de la machine sur laquelle se trouve le serveur et “port” le numéro de port.

Vous pouvez également utiliser ce synopsis si votre client implémente la commande `/connect <machine> [port]`

```
$> ./client [machine [port]]
```

D’une façon générale on veillera à utiliser des commandes clients proches de celles d’un client irc classique

Attention à ne pas confondre les commandes clients et le protocol IRC: http://en.wikipedia.org/wiki/List_of_Internet_Relay_Chat_commands

Chapitre V

Sujet - Partie bonus

Les bonus ne seront évalués que si votre partie obligatoire est PARFAITE. Par PARFAITE, on entend bien évidemment qu'elle est entièrement réalisée, et qu'il n'est pas possible de mettre son comportement en défaut, même en cas d'erreur aussi vicieuse soit-elle, de mauvaise utilisation, etc ... Concrètement, cela signifie que si votre partie obligatoire n'obtient pas TOUS les points à la notation, vos bonus seront intégralement IGNORÉS.

Des idées de bonus :

- gestion d'un prompt dans le client
- transfert de fichier (via le serveur)
- transfert de fichier (sans passer par le serveur)
- prompt dans le client
- respect de la RFC 1459 ou mieux 2813

Chapitre VI

Consignes

- Ce projet ne sera corrigé que par des humains. Vous êtes donc libres d'organiser et nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes listées ici.
- Le binaire du serveur doit se nommer `serveur`
- Le binaire du client doit se nommer `client`
- Vous devez rendre un Makefile. Il devra compiler le client et le serveur, et contenir les règles : client, serveur, all, clean, fclean, re. Il ne doit recompiler les binaires qu'en cas de nécessité.
- Votre projet doit être à la Norme.
- Vous devez gérer les erreurs de façon raisonnée. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, etc...).
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un '\n' :

```
$>cat -e auteur
xlogin$
$>
```

- 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), printf(3), signal(3)
  - mmap(2), munmap(2), lseek(2), fstat(2)
  - les fonctions autorisées dans le cadre de votre libft (read(2), write(2), malloc(3), free(3), etc... par exemple ;-))
  - 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.
- 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.
- Vous pouvez poser vos questions sur le forum dans l'intranet dans la section dédiée à ce projet.

Chapitre VII

Notation

- La notation de `ft_irc` s'effectue en deux temps :
  - La partie obligatoire : le serveur sur 10 points et le client sur 10 points
  - Les bonus :
    - Ils ne seront évalués que si votre partie obligatoire est PARFAITE (Tout doit fonctionner comme attendu, et la gestion d'erreur doit être sans faille).
    - Vous obtiendrez entre 1 et plusieurs points par fonctionnalité bonus distincte et correctement réalisée (À la discrétion de vos correcteurs)
    - Également, l'optimisation de la qualité de certains éléments de votre code seront évalués et pourront donner lieu à des points supplémentaires dans cette partie bonus.
- Bon courage à tous !