Chloé DESDOUITS
Guillerme DUVILLIE

M1 Informatique MOCA

Projet de réseau

23 décembre 2011

UM2

Table des matières

1 Mode d'emploi

Tout d'abord, notre application nécessite l'installation de la bibliothèque Ncurses qui gère l'interface console. Il suffit ensuite de compiler grâce au Makefile.

L'exécutable du serveur se nomme "serveur" et accepte les paramètres suivants :

- -n nom de l'hote (par default 'localhost')
- -a adresse ip de l'hote (par default '127.0.0.1')
- -p numero de port qu'utilisera le serveur d'envoi (par default '13321')
- -ps numero de port secondaire qu'utilisera le serveur de réception (par default '13322')
- -h ou - -help affichage de l'aide

L'exécutable du client se nomme "client" et accepte les paramètres suivants :

- -n nom de l'hote (par default 'localhost')
- -N nom de l'hote du serveur (par default 'localhost')
- -a adresse ip de l'hote (par default '127.0.0.1')
- -A adresse ip du serveur (par default '127.0.0.1')
- -P numero de port qu'utilise le serveur d'envoi (par default '13321')
- -S numero de port qu'utilise le serveur de réception (par default '13322')
- -h affichage de l'aide

Par conséquent, pour lancer l'application en local, il suffit d'appeler les exécutables du serveur et du client sans paramètres. Voici un exemple de ligne de commande pour lancer l'application en réseau :
$ serveur -a 192.168.1.131
$ client -a 192.168.1.96 -A 192.168.1.13

Une fois le client lancé, la demande de contrôle de la caméra s'effectue en pressant la touche 'c'. On peut relâcher le contrôle prématurément en appuyant sur 'q'. La direction est donnée à la caméra grâce aux touches directionnelles.

Les exécutables se quittent en envoyant le signal SIGINT (ctrl-c).

2 Architecture de l'application

Commençons par examiner l'architecture du serveur. Notre serveur est divisé en deux processus, les deux étant multi-thread.

Le processus père est le serveur d'envoi ; il gère le broadcast de la grille. Son thread principal accepte les connexions des clients et crée un thread secondaire par client connecté. Les threads secondaires envoient la grille à leur client respectif.

Le processus fils est le serveur de réception ; il traite les demandes de déplacement de la caméra. Son thread principal gère la file d'attente des clients qui veulent déplacer la caméra. Il lance un thread secondaire pour le premier client qui demande la main. Ce thread secondaire gère le déplacement du pointeur dans la grille. Le thread principal tue le thread secondaire à la fin du temps imparti et relance un thread secondaire pour le client suivant.

Le client, quant à lui, est également multi-threads. Son thread principal gère la réception et l'affichage de la grille. Tandis que son thread secondaire gère les entrées clavier et envoi les demandes de contrôle de la caméra au serveur de réception.

3 Protocoles d'échange

4 Schémas algorithmiques

5 Difficultés et solutions

Lorsqu'un thread secondaire utilise les appels systèmes recv ou send en mode connecté alors que la socket a été fermée de l'autre côté, un signal SIGPIPE est émis. Ce signal doit être intercepté par le thread principal sous peine de terminer l'application.

Réalisé avec LaTeX