Monôme : Julien ELBAZ

Mini-Projet LWT
Jeu de Pendu

Ce qui a été réalisé :

Le projet se compose de deux parties, un client simulant telnet qui consiste en une boucle lisant les messages reçus et envoyant ceux tapés par l'utilisateur, et un serveur qui émule un jeu de pendu 'en réseau'.
Chaque partie est contenue dans un module séparé, et un module contenant des fonctions utilitaires est également présent.
Les scores, identifiants et mots de passe des utilisateurs sont tous contenus dans le fichier id.txt présent à la racine.
Un script shell est de plus fourni pour simplifier la compilation.
Tous les appels système ont été réalisés en LWT.
Une aide est disponible une fois loggué en tapant /aide.

La connection au serveur ainsi que les actions à effectuer se passent exactement comme dans le sujet. Ce rapport a pour but de mettre l'accent sur les quelques variations du projet par rapport à l'énoncé.

Précisions :

1. Un seul fichier pour les scores

Dans le sujet on peut lire que les informations de chaque utilisateur doivent être stockées dans un fichier différent. Je trouve personnellement plus simple de tout placer au même endroit, j'ai donc tout mis dans un seul fichier dont je parse le contenu.

Chaque ligne du fichier correspond à un utilisateur différent, et a le format suivant :

```
[Nom de l'utilisateur] [Mot de passe] [Score]
```

Les champs sont séparés par des espaces. Le mot de passe n'est pas crypté, il n'est jamais envoyé par tcp/ip.

2. Déroulement d'une partie

Un joueur peut soit créer un défi, soit en relever un.
Les défis sont stockés dans une pile.
Si le joueur crée un défi, il choisit le mot ainsi que le nombre de points associé. Cette mise ne peut excéder son score actuel, car si un joueur remporte le défi elle sera déduite. En revanche si un joueur perd le défi cette somme est ajoutée au score de l'auteur du défi.
Remarque : relever son propre défi est possible, mais ne change pas le score.
Le mot est ensuite envoyé à un dictionnaire en ligne par le protocole RFC 2229 (DICT), en l'occurence dict.org. Si le mot existe, on exécute ensuite un script qui regarde le nombre d'occurences du mot sur google.com. On enregistre alors le défi dans le pile.

Un joueur peut alors relever le défi, et une partie de pendu se lance comme décrit dans l'énoncé.

3. Structure du code

Le projet n'étant pas particulièrement expansif en lignes de code, la séparation en module est extrèmement restreinte. Seuls trois modules ont été utilisés.

- Server : pour les opérations liées au serveur
- Client : pour les opérations du client
- Utils : pour des fonctions utilitaires « passe partout »

Le code est entièrement commenté.

4. Robustesse

Un effort a été fait pour que le serveur soit capable de gérer la plupart des exceptions liées à une mauvaise utilisation du protocole. Néanmoins, même si beaucoup de cas sont gérés il est possible qu'un envoi puisse crasher le serveur en de rares occasions.