TP 4 et 5 : Projet de programmation système unix

Durant la première séance de Projet, chaque étudiant doit choisir son sujet de projet dans le premier quart d'heure. Un peu plus de temps peut être laissé aux étudiants qui souhaitent proposer un sujet personnel à traiter. Le mieux serait d'arriver avec un projet déjà préparé. Les projets doivent être traités en monôme. Lorsque le projet est terminé il est demandé de compresser dans un unique fichier ZIP nommé :
"`ProjetLinux2017_NOM_PRENOM.zip`" :

- les programmes source en langage C ;
- un rapport de quelques pages au format PDF décrivant l'architecture des programmes réalisés (avec schémas le cas échéant), les méthodes utilisées et leur justification dans le cadre du projet. Le rapport comportera une notice explicative pour la compilation des programmes et leur exécution (idéalement un makefile devrait être présent avec les fichiers sources).

Le fichier ZIP sera déposé sur la plateforme Moodle à la date limite du Vendredi 21 décembre 2018 minuit. Les programmes exécutables ne doivent pas être insérés dans le fichier ZIP.

Projet 1

Réaliser un système client/serveur permettant d'implémenter le programme *telnet*. Côté serveur le programme acceptera les connexions de clients et lancera pour le traitement de chaque client un processus fils capable de gérer le service. Toutes les demandes du client seront réalisées sous les droits du serveur. Côté client, le programme permettra d'engager une communication avec le serveur afin d'exécuter les commandes demandées par le client sur la machine distante. Pour cela le client devra s'authentifier à l'aide d'un login et d'un mot de passe. Tous les login et mots de passe seront rangés dans un fichier côté serveur, que le serveur pourra consulter afin d'accepter ou de refuser une connexion. Une fois connecté sur la machine cible, le client pourra demander l'exécution des commandes standards acceptées par le shell (gestion de fichier, parcours de répertoires, exécution de programmes, etc.). A cette fin, le client disposera en quelque sorte d'un compte sur le serveur, c'est à dire d'un répertoire local dédié auquel il aura accès et dans lequel il pourra créer d'autres répertoires et réaliser des opérations sur ses fichiers.
Le serveur sera lancé en tâche de fond. Il devra accepter le signal `SIGUSR1` réalisant l'envoi d'un message aux utilisateurs connectés annonçant la cloture immédiate de la connexion, puis la fermeture des connexions en cours, puis enfin l'arrêt de sa propre exécution. Une deuxième signal `SIGUSR2` produira l'affichage par le serveur du nombre de connexions en cours.

Projet 2

Projet identique au projet 1 en utilisant les threads : côté serveur le programme acceptera les connexions de clients et lancera pour le traitement de chaque client un thread capable de gérer le service.

Projet 3

Créer un serveur de messagerie. Le serveur met en relation les clients deux à deux à mesure qu'ils se connectent. Lorsque deux clients sont en relation, chaque client peut alternativement saisir un texte et lire un texte écrit par l'autre. Le serveur affichera chez les deux clients les messages suivants au fur et à mesure de la communication :

```
Le client distant dit: bonjour...
Donnez votre réponse: ...
etc.
```

La gestion des tâches clientes sera réalisée par un processus fils.
Le serveur sera lancé en tâche de fond. Il devra accepter le signal `SIGUSR1` réalisant l'envoi d'un message aux utilisateurs connectés annonçant la cloture immédiate de la connexion, puis la fermeture des connexions en cours puis enfin l'arrêt de sa propre exécution. Une deuxième signal `SIGUSR2` produira l'affichage par le serveur du nombre de connexions en cours.

Projet 4

Projet identique au projet 3 en utilisant les threads : la gestion des tâches clientes sera réalisée par un thread dédié.

Projet 5

Le projet concerne un jeu multijoueur (similaire à pacman) sur le réseau local. Il implique l'utilisation de : sockets, fourchettes et mémoire partagée. Le programme *serveur* contrôlera la carte (une matrice NxN) et la connexion des joueurs. Chaque joueur peut se déplacer vers le haut, le bas, la gauche et la droite. Les mouvements des joueurs seront effectués par le programme *client*. Le client affiche également l'état de la matrice obtenue à partir du *serveur*. Au début, le statut de la carte est aléatoire, il ya un certain nombre de cellules occupées (¡30%). Les joueurs ne peuvent pas traverser les cellules occupées. Les joueurs ne peuvent se déplacer que dans des cellules vides. Chaque fois qu'un joueur lecteur se connecte au serveur, le serveur lui attribuera une position aléatoire dans la carte. Vous êtes libre de choisir l'interaction entre les joueurs, par exemple si elles arrivent à la même cellule un d'entre eux perd. Je fournirais les bibliothèques graphiques pour vous permettre de visualiser la carte mais elle peut également être affiché avec du texte.

Projet 6

Sujet libre faisant intervenir, soit un système de type client/serveur réseau (la gestion des clients par le serveur s'effectuant soit par la création de processus fils, soit par le lancement de threads dédiés), soit un système de communication inter-processus utilisant la libraire IPC (files de message, mémoire partagée, sémaphores). Dans tous les cas, les signaux devront être utilisés pour réaliser une interaction avec le ou les programmes comme par exemple dans les projets 1 et 2 ci-dessus. Le sujet devra bien entendu être validé par le professeur avant d'être traité.