Sujet du TP n° 6

Préambule

Les ressources du cours (supports, exemples, et autres documents) et des TP (les sujets, les fichiers de travail ainsi que les corrigés) se trouvent à l'URL : http://www.emse.fr/~lalevee/ismin/pse, dénommé **[site]** dans les documents fournis.

Ce TP suppose que vous avez installé l'archive **PSE.tar**. Si ce n'est pas le cas, consultez le sujet du premier TP.

Placez-vous dans le répertoire **PSE/TP6**.

L'appel système "select" (transparent 114)

Pratique 1

- le répertoire **TP6** contient le fichier **ecritube.c** qui permet d'écrire des lignes de texte dans un FIFO, dont le nom est communiqué en paramètre (fichiers **tube1** ou **tube2**)
- le fichier **exercice1.c** permet d'afficher les lignes lues à partir d'un de ces deux FIFO, alimentés par **ecritube.c** jusqu'à que la ligne **fin** soit tapée
- compilez l'application en tapant la commande **make**
- pour l'exécuter, ouvrez 2 autres fenêtres **terminal** pour disposer de 3 fenêtres
- dans la première, exécutez la commande **ecritube tube1**
- dans la seconde, exécutez la commande **ecritube tube2**
- dans la troisième, exécutez la commande **exercice1**
- saisissez des lignes de texte dans les deux premières fenêtres dans un ordre quelconque

Exercice 1

- vous avez constaté que le serveur doit imposer un ordre de lecture, d'abord **tube1** puis **tube2**
- modifiez ce fichier pour qu'il sélectionne le canal prêt en lecture, en utilisant l'appel système **select()**.

Applications multithread

Dans le TP précédent, nous avons mis en œuvre un serveur multithread, dans lequel un thread est créé à chaque connexion acceptée (voir le corrigé dans le répertoire **TP5-C**). Si le nombre de clients devient important, le système peut être saturé (cela peut constituer ainsi un risque de sécurité important en cas d'attaque par déni de service).

Pour remédier à ce problème, le but de cette partie sera de mettre en œuvre une cohorte (*pool*) de taille fixe de threads *worker*. L'algorithme général du thread principal est le suivant :

```
initialiser_cohorte()
répéter
    accepter une connexion entrante
    si pas de worker libre dans la cohorte alors
        attendre
    finsi
    sélectionner un worker libre et le marquer occupé
    communiquer canal à ce worker
fin_répéter
```

Les threads *worker* reçoivent le numéro de canal à partir duquel ils réceptionnent les requêtes. Dès la réception de la ligne "**fin**", ils se déconnectent du client et mettent à jour leur statut (d'occupé à libre). Vous devrez trouver une solution pour gérer la cohorte, pour gérer l'attente et les statuts des threads *worker* (note : pour attendre n millisecondes, utilisez la fonction **usleep()**, précision en microsecondes, ou **nanosleep()**, précision en nanosecondes).

Projet

- copiez **serveur.c** et **client.c** du TP précédent dans **TP6**
- modifiez le fichier **serveur.c** pour réaliser l'application telle que décrite ci-dessus.