Programmation système II : gestion des événements

Timothée Ravier, LIFO, INSA-CVL, LIPN

1re année cycle ingénieur STI, 2013 – 2014

Pour ce TD, nous allons d'abord utiliser les code sources d'exemple disponibles à l'adresse : `https://tim.siosm.fr/cours/`.

Puis ceux à l'adresse `http://man7.org/tlpi/code/download/tlpi-140312-dist.tar.gz`.

1 Socket UNIX

Lisez le code source des programmes `unix_client.c` et `unix_server.c`. Compilez et testez.

- **Question 1 :** Expliquer le comportement de ces programmes.

2 select, poll, epoll

Lisez le code source des programmes `t_select.c`, `poll_pipes.c` et `epoll_input.c`. Compilez et testez.

- **Question 2 :** Expliquer le comportement de ces programmes.

3 signalfd

Lisez le code source du programme `signalfd_sigval.c`. Compilez et testez.

- **Question 3 :** Expliquer le comportement de ce programme.

4 Maintenant que l'on a tout vu...

Combinez tout ce que vous avez vu jusqu'à présent pour faire un programme qui écoute sur une socket réseau, qui transmet tout ce qu'elle reçoit par l'intermédiaire d'une zone de mémoire partagée à un autre processus qui se charge de mettre en majuscule tout le contenu reçu. Une fois le contenu modifié, ce programme l'envoie par l'intermédiaire d'une file de message à un autre programme qui le retransmet enfin au premier programme à l'aide d'un pipe nommé.

Vous penserez à gérer les signaux correctement avec signalfd pour tous les processus, à utiliser select/poll ou epoll, à bien vérifier tous les codes de retours de toutes les fonctions...

Vous pouvez aussi essayer de faire tout cela dans un seul processus avec des threads.

Reprenez les exemples du TD sur les threads pour les réécrire sans utiliser de threads et juste avec select/poll ou epoll.

5 Références

— The Linux Programming Interface by Michael Kerrisk : `http://man7.org/tlpi/index.html`