Rémy Malgouyres
Université Clermont 1
http : //laic.u − clermont1.fr/~mr/

Programmation système et réseaux

TP n° 4
Système client serveur (sur une machine)

Durée 1 semaine

Objectifs :
Le but de ce TP est d'implémenter un squelette d'application avec système client-serveur, avec plusieurs processus sur une même machine. Comte-tenu du temps imparti, seul le système de communication entre processus sera mis en place.

Note. Chaque semaine, le listing commenté du TP est à rendre en début de la scéance de la semaine suivante.

1 Exécuter `xterm` dans un fils

Dans une console, tapez `xterm`. Qu'observez-vous ? Pour ouvrir un `xterm` qui exécute une commande, utilisez l'option `-e`.

Exemple. Écrire une script shell `monscript` qui prend en paramètre le nom de fichiers ou de répertoires, et qui liste ces fichiers et répertoires avec leurs droits. Avant de ce terminer, le script exécute un `read` pour attendre qu'on appuie sur retour-chariot. Lancer un `xterm` qui exécute le script `monscript`.

```
$ xterm -e ./monscript ~ /tmp .
```

2 Créer un stsème client serveur

Exercice 1 a Écrire un programme *C* `serveur.c` qui crée deux tubes permettant de communiquer dans les deux sens avec un processus fils créé par `fork`. Dans le fils, on lancera par `execv` un `xterm` qui exécute un deuxime programme *C* `client.c`. De plus, on passera en argument à `client.c` les descripteurs de tubes qui lui permettront de communiquer avec `serveur.c`

b) Dans le programme `client.c`, saisir une chaîne de caractère au clavier et la transmettre à `serveur.c` par un tube. On transmettra aussi le *PID* du programme `client.c` en cours d'exécution et le *PID* du `xterm` créé. Le processus père (dans `serveur.c`) affiche son *PID* le *PID* de son fils, et transmet son *PID* au (petit-)fils (qui exécute `client.c`) via le tube. Le petit fils affiche le message obtenu.

c) Faire une copie de sauvegarde des programmes du *a*) et du *b*) qui marchent. Dans le processus père (qui exécute `serveur.c`), on incluera la création du `xterm` client dans une boucle `while` qui saisit un acractère au clavier, et crée un xterm à chaque fois qu'on tape un $'n'$. De plus, suite au `fork`, dans le processus père (cas où `fork` $\neq$ `0`), on appellera à nouveau `fork` pour créer un nouveau fils (frère du `xterm`). c'est ce frère du `xterm` qui dialoguera avec le programme qui exécute `client.c` avant de se terminer par `exit`, tandis que le processus père (le processus original) reviendra dans le while pour créer d'aures `xterm`