SYR2 TP2-3

Processus et pipes

Merci d'envoyer le résultat de votre TP à votre encadrant avant le prochain TP. Vous devrez envoyer:

- Trois fichiers sources respectivement appellés `shell1.c`, `shell2.c` et `shell3.c` (pas d'autres noms de fichier svp!). Merci d'attacher chaque fichier à votre mail sans créer de fichiers tar, zip ou autre.
- Un fichier Makefile (conçu dans les règles de l'art).
- Un fichier texte appellé `README` avec les noms des deux étudiants du binôme.

Un "shell" est un programme qui lit des commandes tapées par un utilisateur et qui crée des processus pour exécuter ces commandes. Un vrai shell tel que `bash` possède de très nombreuses fonctionnalités comme le support de variables, la possibilité d'exécuter des scripts etc. Le but de ce TP est d'écrire un petit shell minimaliste qui lit simplement des commandes au clavier et qui les exécute.

1. Ecrivez un premier programme appellé `shell1.c` qui lit simplement le nom d'un programme au clavier (vous pouvez utiliser la fonction `fgets()` pour effectuer cette lecture). Le shell crée un nouveau processus qui exécute le programme demandé par l'utilisateur. Le shell attend que le programme soit terminé avant de demander une commande suivante. Par exemple, si l'utilisateur tape "ls", il doit voir la liste des fichiers du répertoire courant. La commande "exit" signifie que le shell doit s'arrêter.

   Bien entendu vous devrez utiliser les fonctions vues en cours (`fork()`, différentes versions d'`exec()`). L'utilisation de fonctions de haut niveau telles que `system()` est interdite.

2. Copiez votre programme `shell1.c` dans un nouveau fichier `shell2.c`. Etendez `shell2.c` pour permettre à l'utilisateur de passer des paramètres dans sa ligne de commande. Par exemple votre shell doit correctement exécuter un commande telle que "`ls -l /tmp`"

   Pour interpréter une ligne de commande et la découper mot par mot vous pouvez au choix implémenter l'interprétation à la main ou utiliser la fonction `strtok()`.

3. Copiez votre programme `shell2.c` dans un nouveau fichier `shell3.c`. Etendez `shell3.c` pour permettre à l'utilisateur de taper des commandes avec un pipe. Par exemple, "`ls /tmp | wc -l`" retourne le nombre de fichiers présents dans le répertoire `/tmp`. Vous aurez besoin d'utiliser la fonction `dup2()`.

   Je vous suggère **fortement** de ne pas vous focaliser sur l'interprétation de la ligne de commande. Commencez par une version où le shell demande deux commandes l'une après l'autre puis crée les deux processus et le pipe. Par exemple, pour effectuer la commande "`ls /tmp | wc -l`" vous pouvez réaliser quelque chose comme ceci:

```
> ./shell3
commande1> ls /tmp
commande2> wc -l
15
commande1> exit
>
```

   OPTIONNEL: Vous pouvez ensuite étendre votre programme pour supporter la vraie syntaxe "`ls /tmp | wc -l`", voire permettre d'enchaîner davantage de processus: "`sort foo | uniq | wc -l`".

— fin —