TP 4– Communication entre processus : les tubes

Concepts abordés

- Création de tubes anonymes et nommés.
- Lien entres les tubes et la fonction | (pipe) du shell.

Principaux appels systèmes et fonctions utiles

- pipe(), mkfifo()
- fopen(), fread(), fwrite(), fscanf(), fprintf()
- open(), read(), write()
- dup2()
- execvp()
- fork()
- atoi()

N'hésitez pas à utiliser la commande man pour obtenir la documentation de chaque fonction.

Contraintes et remise des TPs

- **Le code de chaque exercice doit être dans un repertoire séparé (exercice1, exercice2, etc.).**
- **Chaque répertoire d'exercice doit contenir un fichier Makefile permettant de compiler les réponses à chaque question.**
- Une note de propreté et style du code est attribuée à votre travail et prise en compte lors de l'évaluation. Elle inclue : la lisibilité du code (taille des fonctions, noms de variables explicites, etc.), sa propreté (libération de mémoire, contrôle des erreurs, réemploi de variables, etc.), sa qualité algorithmique, la présence et la pertinence des commentaires, les gardes `#ifndef` dans les headers, l'indentation et mise en page. L'ensemble de ces consignes est détaillé dans le document "propreté et lisibilité du code" distribué au premier TP et disponible sur le site de l'UE.
- Le code source doit être envoyé forme d'une archive tar.gz[1]. Votre archive doit être propre : pas d'exécutables, pas de .o, pas de fichiers temporaires. **pour le mercredi soir suivant la fin du TP** par e-mail à l'adresse `alexandre.coninx@upmc.fr`
- Les supports de cours, les exemples et les fichiers utiles aux TPs sont sur la page du cours : `http://enseignement.coninx.org/rob4-infosys/`.

Conseils supplémentaires

- Evitez les déclarations de tableaux à taille variable (`int x; x = ...; int tab[x];`). Elles ne sont supportées que pour le standard C99 (elles n'existent pas dans les standards antérieurs et leur support est optionnel en C11).

1. Pour faire une archive tar.gz : `tar cvzf mon_archive.tar.gz mon_repertoire`

Barème de notation

Exercice 1 : coefficient 1. Exercice 2 : coefficient 1. Exercice 3 : coefficient 2. Style, propreté et lisibilité du code : coefficient 1.

Exercice 1 – Tubes et pipe du shell : Nombre de fichiers dans un répertoire

1. Ecrire un programme affichant le nombre de fichiers dans un répertoire. Pour ce faire, vous devez combiner les appels aux commandes `ls` et `wc -l` en utilisant un tube, deux processus parents et les redirections de l'entrée et de la sortie standard comme indiqué en cours.

Exercice 2 – Tubes anonymes bidirectionnels : Exercice simple

1. Ecrire un programme permettant la création de deux processus (père et fils) s'échangeant des informations au travers de deux tubes anonymes. Le processus père envoie le message `"hello␣!"` au processus fils sur le tube 1 et, une fois ce message envoyé, attend le message `"hello␣OK!"` sur le tube 2. Le processus fils attend le message `"hello␣!"` sur le tube 1 et, une fois ce message reçu, envoie le message `"hello␣OK!"` sur le tube 2. Cette séquence est répétée 3 fois.

Exercice 3 – Tubes nommés bidirectionnels : Client / Serveur de calcul

Le but de cet exercice est d'ecrire deux programmes tels que :
- Le premier programme, appelé "client", envoie, au travers d'un tube nommé, deux nombres entiers et un opérateur $(+, -, \times, /)$ au processus associé au second programme, appelé "serveur".
- Le programme "serveur" effectue le calcul correspondant et retourne, au travers d'un autre tube nommé, le résultat au processus "client" qui l'affiche.

1. a) Deux choix sont possibles pour réaliser la lecture et l'écriture dans les tubes, lesquels sont-ils ? Quels sont les principaux appels systèmes ou fonctions associés à chaque possibilité ?

   b) De facon analogue, deux possibilités sont offertes pour le format des données échangées par les deux processus, lesquelles sont-elles ? Quelles sont les avantages et les inconvénients associés à chacune ?

   Vous pouvez choisir n'importe lesquelles de ces possibilités pour réaliser la suite de l'exercice, à votre choix. Le choix de l'une par rapport à l'autre n'aura pas d'impact sur la notation tant que vos programmes fonctionnent et répondent aux questions.
2. Écrire le programme "serveur" qui crée (si besoin) et ouvre les deux tubes nommés, attend de recevoir deux nombres entiers et un operateur sur le premier, effectue le calcul correspondant (vous pouvez réemployer le code écrit au TP3 ou vous en inspirer) et retourne le résultat dans le second tube.

3. Écrire le programme “client” qui ouvre les deux tubes nommés, envoie au travers du premier deux nombres entiers et un opérateur ($+, -, \times, /$) au processus associé au second programme, puis attend la réponse sur le second tube et l'affiche.
4. Mofifier le programme “client” de manière à ce que les opérandes et l'opérateur soient passés comme arguments du `main()` .
5. Mofifier le programme “serveur” et le programme “client” de manière à ce que le caractère `'f'` passé comme argument au programme “client” engendre la fin du processus “serveur”.