TP 5– Communication entre processus : les tubes

Sovannara Hak et Vincent Padois
hak@isir.fr

Concepts abordés

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

Fonctions et opérateurs utiles

- pipe(), mkfifo(), fwrite(), fclose(), fprintf()
- close(), open()
- read(), write()
- dup2()
- execvp()
- fork()
- atoi(), atof()

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 répertoire séparé (exercice1, exercice2, etc.).
- Chaque répertoire d'exercice doit contenir un fichier Makefile permettant de compiler les réponses à chaque question.
- Lorsque c'est nécessaire, ajoutez dans le repertoire de l'exercice des fichiers texte ou image pour répondre aux questions (avec un nom explicite).
- N'oubliez pas de rendre une archive « propre » (pas de .o, pas binaires, pas de fichier temporaires *.swp ou *~). La cible « clean » de vos makefiles est faite pour cela !
- Indentez vos fichiers (commande `indent` ou éditeur de texte civilisé).
- N'oubliez pas les "gardes" (#ifndef ...) dans vos fichier `.h`.
- La correction tiendra compte de la briéveté des fonctions que vous écrivez (évitez les fonctions de plus de 25 lignes) ; n'hésitez pas à découper une fonction en plusieurs sous-fonctions plus courtes.
- Votre enseigant vérifiera avec Valgrind l'absence d'erreurs (fuites mémoires, lectures invalides, ...).
- La convention de nommage des fonctions est la suivante : une fonction doit avoir un nom explicite et si son nom se compose de plusieurs mots, ces derniers sont écrits en minuscule et séparés d'un tiret bas _. Ainsi une fonction qui affiche un tableau sera appelée `affiche_tableau`.
- Le code source doit être envoyé forme d'une seule archive tar.gz [1]

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

- Votre archive doit être propre : pas d'exécutables, pas de .o, pas de fichiers temporaires. **au maximum 30 minutes après la fin du TP**
- Envoyez votre archive par e-mail à `hak@isir.upmc.fr`
- Les fichiers utiles et les transparents de cours sont sur `http://pages.isir.upmc.fr/~hak`

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

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

Exercice 2 – Tubes anonymes bidirectionnels : Exercice simple

1. Écrire 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 `"Comment␣est␣votre␣blanquette?"` au processus fils sur le tube 1 et, une fois ce message envoyé, attend le message `"Elle␣est␣bonne."` sur le tube 2. Le processus fils attend le message `"Comment␣est␣votre␣blanquette?"` sur le tube 1 et, une fois ce message reçu, envoie le message `"Elle␣est␣bonne."` sur le tube 2. Cette séquence est répétée 3 fois (le père envoie trois messages, et reçoit donc trois réponses). Chaque processus affichera la chaîne de caractère reçue. Il faudra bien sûr vérifier que les messages soient corrects avant d'envoyer une réponse.

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

1. Écrire deux programmes tels que :
   - Le processus associé au 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 processus "serveur" effectue le calcul correspondant (vous pouvez vous inspirer du code écrit au TP3) et retourne, au travers d'un autre tube nommé, le résultat au processus "client" qui l'affiche.
2. Mofifier le programme "client" de manière à ce que les opérandes et l'opérateur soient passés comme arguments du `main()`.
3. 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".