Programmation Système

Fiche 3 : Communication par tubes

Cette fiche ainsi que les fichiers à télécharger sont disponibles sur le site
https://gforgeron.gitlab.io/progsys/

Exercice 1.1 Dans cet exercice on cherche à reproduire la commande shell `cmd1 | cmd2 [liste d'arguments]` afin de mettre en avant les les avantages à utiliser des tubes plutôt que des fichiers temporaires.

1. Il s'agit d'écrire une commande `faux-pipe cmd1 cmd2 [liste d'arguments]` qui utilise un fichier intermédiaire temporaire pour stocker la sortie standard de la première commande, ce même fichier servira d'entrée standard de la seconde. Tester votre commande sur `ls | cat -n` et `cat | cat -n`.
2. Écrire un programme `vrai-pipe cmd1 cmd2 [liste d'arguments]` qui met en place un tube entre deux processus pour relier la sortie de la commande `cmd1` à l'entrée de la commande `cmd2`. Tester la commande.
3. Faire en sorte que la commande `vrai-pipe` retourne la valeur de sortie de la première commande lorsque celle-ci n'est pas nulle et celle de la seconde autrement.
4. Vérifiez que l'un des processus ne se termine pas si vous ne fermez aucune extrémité du tube. Quels sont les appels impératifs à `close()` ?

Exercice 1.2 Dans quelle condition un processus peut-il être bloqué lorsqu'il cherche à écrire dans un tube ? Réaliser une expérience permettant de mesurer la taille du buffer interne d'un tube.

Exercice 1.3 Que se passe-t-il lorsqu'un processus essaye de lire sur un tube alors que celui-ci est fermé en écriture et donc qu'il n'y a plus d'écriture possible sur ce tube ? Même question lorsqu'un processus essaye d'écrire sur un tube alors que celui-ci est fermé en lecture.

Exercice 1.4 On souhaite disposer d'une commande `log` grâce à laquelle on pourrait exécuter un programme quelconque en récupérant automatiquement une copie de sa sortie standard dans un fichier. En quelque sorte, il s'agit de dédoubler chacun des caractères envoyés sur sa sortie standard : un exemplaire est reproduit sur la sortie standard par défaut alors que l'autre est enregistré dans un fichier.

Le programme `log` prend en premier argument le nom du fichier à utiliser pour la sortie. Les arguments suivants constituent la commande à exécuter :

```
log <file> <command> [ <args> ]
```

Pour fixer les idées, voici un exemple d'utilisation de ce programme :

```
[user@machine] log fichier ls -la
drwxr-xr-x  2 rnamyst rnamyst    80 2018-09-26 22:29 ./
drwxrwxrwt  7 root    root      180 2018-09-26 22:27 ../
-rw-r--r--  1 rnamyst rnamyst     0 2018-09-26 22:29 fichier
-rw-r--r--  1 rnamyst rnamyst 20041 2018-09-26 22:27 toto.pdf
[user@machine] cat fichier
drwxr-xr-x  2 rnamyst rnamyst    80 2018-09-26 22:29 ./
drwxrwxrwt  7 root    root      180 2018-09-26 22:27 ../
-rw-r--r--  1 rnamyst rnamyst     0 2018-09-26 22:29 fichier
-rw-r--r--  1 rnamyst rnamyst 20041 2018-09-26 22:27 toto.pdf
[user@machine]
```

1. Programmer la commande `log` en utilisant la commande unix `tee`.
2. En regardant attentivement la sortie donnée en exemple pour la commande `log`, on voit la ligne suivante apparaître :

   ```
   -rw-r--r--  1 rnamyst rnamyst     0 2018-09-26 22:29 fichier
   ```

   Expliquez pourquoi. Avec votre implémentation de `log`, est-il possible que le fichier n'apparaisse pas toujours ? Est-il possible que la taille observée soit non nulle ? Pourquoi ?