Travaux Dirigés Programmation Système: Feuille 2

Informatique 2ème année. ENSEIRB 2018/2019

—*Mathieu Faverge* - `mfaverge@enseirb.fr` —

Gestion des inodes et descripteurs de fichiers

▸**Exercice 1.** *Fichiers FIFO :*

Un fichier FIFO est un type spécial de fichier qui correspond à un tube : ce qu'on écrit d'un coté du tube peut être lu de l'autre coté. Il peut être ouvert par plusieurs processus, tant en lecture qu'en écriture. Lorsque des processus échangent des données par le biais d'une file FIFO, le noyau transfère les informations de manière interne, sans passer par une écriture réelle dans le système de fichier. Ainsi, le fichier spécial FIFO n'a pas de véritable contenu, c'est essentiellement un point de référence pour que les processus puissent accéder au tube en employant un nom dans le système de fichiers.

Le but de l'exercice est de créer un "chat" à l'aide d'un fichier FIFO.

1. *Créez un programme C* `serveur.c` *qui crée une FIFO et affiche sur la sortie standard les caractères qui sont lus par la fifo. On appellera la FIFO 'versserveur' et le programme se fera simplement en quatre étapes :*
   - *Utilisation de* `mknod` *pour créer le FIFO*
   - *Ouverture du FIFO (à l'aide de ...* (cf TD précédent)*)*
   - *Lecture sur le FIFO (à l'aide de ...* (cf TD précédent)*)*
   - *Écriture sur la sortie standard*

   La lecture sur la FIFO est bloquante : votre programme sera bloqué en attente que des caractères arrivent dans la FIFO. Vous devez écrire le programme suivant pour que ca fonctionne.
2. *Créez un programme C* `client.c` *qui ouvre une FIFO 'verserveur' en écriture et y écrit le texte lu sur l'entrée standard. On utilisera les fonctions* `open(2)`, `read(2)` *et* `write(2)` *pour ouvrir la FIFO, lire sur l'entrée standard et écrire dans la FIFO.*
3. *Lancez dans un terminal le serveur puis dans un autre terminal, le client. Que se passe-t-il ? Que se passe-t-il si vous lancez plusieurs client et un serveur ? Écrivez soigneusement les gestions d'erreurs.*
4. *Modifiez votre programme pour pouvoir transmettre des caractères ainsi que des fichiers entiers (binaires, images, ...). Vous pouvez utiliser les redirections en entrée et en sortie pour vérifier son fonctionnement.*
5. *** Pour faire un chat, il est nécessaire que l'échange entre les deux programmes soit bi-directionnel. Modifiez les programmes précédents en créant 2 FIFOs (*`versclient` *et* `versserveur`*) pour que cela fonctionne. Attention, toute lecture sur une FIFO est bloquante !*
6. *** On voudrait faire un chat entre n clients ($n > 2$), tel que tout ce qui est tapé par un client est vu par les autres. Comment feriez-vous ?*

▸**Exercice 2.** *Descripteurs de fichier et dup :*

1. *créez un fichier* `donnees` *qui contient la chaîne de caractères "abcdefgh"*
2. *écrivez un programme C qui lit les quatre premiers octets de ce fichier et les affiche*
3. *écrivez un programme C qui duplique avec* `dup(2)` *le descripteur de fichier correspondant à ce fichier, qui lit 4 octets depuis le premier descripteur et les affiche, puis lit 4 octets depuis le deuxième descripteur et les affiche*
4. *écrivez un programme C qui fait exactement la même chose mais qui refait un* `open(2)` *du même fichier au lieu du* `dup(2)`*.*
5. *Comparez l'exécution des deux derniers programmes et expliquez ce que vous constatez.*

▸**Exercice 3.** *ls :*
A l'aide des appels `opendir(3)`,`readdir(3)`,`closedir(3)` *écrivez un programme C qui affiche la liste des fichiers (au sens large) contenus dans le répertoire courant, un fichier par ligne.*

En utilisant les fonctions `stat(2)`/`lstat(2)`/`fstat(2)`*, ajoutez devant chaque fichier une lettre indiquant son type : "-" si régulier, "d" si répertoire, "l" si lien... Créez des liens symboliques dans votre compte et testez votre programme.*

Pour finir, modifiez votre programme de telle sorte que la liste des répertoires dont on veut afficher le contenu soit fournie en argument :

```
$ ./myls . .. / /tmp
```

Si aucun argument n'est fourni au programme, alors il utilisera le répertoire courant (i.e. `$./myls` *équivaut à* `$./myls .`*)*

▸**Exercice 4.** *ls, la suite :*
Pour aller plus loin :

- *Ajoutez une option* `-l`*, permettant d'activer l'affichage du type des fichiers (par défault celui-ci sera inactif). On utilisera la fonction* `getopt(3)` *pour gérer les options.*
- *Ajoutez une option* `-R`*, permettant d'effectuer un affichage récursif du contenu du(es) répertoire(s).*
- *Ajoutez l'affichage des droits, de la date de création (*`ctime`*), du nom de l'utilisateur (*`getpwuid`*), . . .*

▸**Exercice 5.** *le chat, la suite :*
Pour aller plus loin : On reprend le chat à plusieurs du premier exercice. Pour éviter qu'un client recoive ses propres messages en echo, on peut envoyer à message un premier octet qui identifie le client. Un client recevant son message ne l'imprimera alors pas. Proposez une solution pour implémenter ce protocole.