Université Joseph Fourier, Grenoble
Licence d'Informatique L3

Introduction aux Systèmes et Réseaux

TP n°3 : Fichiers, flots d'entrées-sorties et tubes

1 Obtention des métadonnées d'un fichier

Exercice 1 'Ecrire un programme `fileinfo` basé sur la primitive `stat` (ou `fstat`), qui affiche les informations suivantes sur le fichier dont le nom lui est passé en paramètre :

- nom ;
- type (fichier normal, répertoire, autre) ;
- droit d'accès (lecture, écriture) pour le propriétaire ;
- date de la dernière modification [1].

Exemples d'utilisation :

```
$> fileinfo csapp.h
csapp.h regular r- Fri Jan 11 18:54:07 2007
$> fileinfo /tmp
/tmp directoy rw Sat Mar  1 21:13:55 2007
$> fileinfo /dev/zero
/dev/zero other rw Wed Feb 27 11:48:54 2008
```

Les informations sur le type du fichier et les droits d'accès de son propriétaire sont stockées dans le champs `st_mode` de la structure (de type `struct stat`) intialisée par l'appel à la primitive `stat`. La date de dernière modification est stockée dans le champ `st_mtime` de cette même structure. Voir `man 2 stat` et la documentation de la Glibc [2] pour les détails.

2 Redirections

Exercice 2 Tester/comprendre les commandes suivantes :

```
touch n.txt
echo 1 > n.txt
echo -1 >> n.text
echo 3 >> n.txt
sort < n.txt
sort < n.txt > outfile 2>&1
ls -l | less
```

1. Pour l'affichage de la date, on pourra s'appuyer sur la primitive `ctime`.
2. `http://www.gnu.org/software/libc/manual/html_node/File-Attributes.html#File-Attributes`

```
man -s2 stat | tail
cat n.txt | wc -l
echo "Hello" | tr "[:lower:]" "[:upper:]"
```

3 Tubes

Exercice 3 Écrire un programme qui crée un tube, puis un fils (rappelons que le fils hérite des descripteurs du père). Le père doit fermer la sortie du tube, puis effectuer une boucle de lecture sur le clavier. Chaque ligne lue est envoyée en entrée du tube. Cette boucle s'arrête lorsque l'utilisateur presse ctrl-d.

Le fils doit fermer l'entrée du tube, puis effectuer une boucle de lecture sur la sortie du tube. Chaque ligne lue doit être affichée à l'écran (les lignes sont donc affichées progressivement, une par une). Cette boucle s'arrête lorsque l'on détecte que l'entrée du tube a été fermée par le père.

Exercice 4 On reprend le principe de la question précédente en considérant trois processus. Le père crée un tube, puis deux fils. Le fils 1 envoie dans le tube ce qu'il lit au clavier, et le fils 2 affiche à l'écran ce qu'il lit dans le tube.

Le père doit se terminer (et rendre la main au shell) après la fin de ses deux fils. Quelles sont les conditions d'arrêt pour chacun des fils ? Quelle précaution particulière faut-il prendre par rapport à celle du fils 2 ?

Exercice 5 Reprendre le programme qui relie le père et le fils par un tube et le changer afin de rediriger la sortie standard du père vers l'entrée du tube et la sortie du tube vers l'entrée standard du fils. Lancer ensuite dans chacun des processus une commande au moyen d'`execve` (ou de l'une de ses variantes) [3] :

— pour le père, (émetteur dans l'entrée du tube), lancer la commande `cat` [4] ;

— pour le fils, (récepteur à la sortie du tube), lancer la commande `nl` [5].

Remarque : la primitive `execve` et ses variantes préservent (par défaut) la table des descripteurs ouverts pour un processus [6].

Rappels et élements de solution : La primitive `dup2` sert à dupliquer un descripteur de fichier. Elle prend en arguments deux descripteurs de fichiers ouverts (source et destination de la copie). Pour rediriger la sortie standard de l'écran vers un fichier (resp. l'entrée standard du clavier vers un fichier), l'idée est d'appeler `dup2` pour recopier le descripteur du nouveau fichier de sortie (resp. d'entrée) vers celui de la sortie standard (resp.

3. Attention à bien passer les arguments au programme lancé, soit sous forme de liste (variantes `execl`), soit sous forme de vecteur (variantes `execv`). Il n'y a ici qu'un seul argument non nul : le nom du programme (`cat` ou `nl`).

4. Lorsqu'elle est lancée sans arguments, la commande `cat` lit des données (ligne par ligne) sur l'entrée standard et les affiche sur la sortie standard.

5. Lorsqu'elle est lancée sans arguments, la commande `nl` lit des données (ligne par ligne) sur l'entrée standard et les affiche sur la sortie standard en appliquant un formatage (espaces et numéro de ligne).

6. Ce comportement est modifiable via un paramètre (*flag*) passé à la primitive `open` lors de l'ouverture du fichier.

entrée standard) (**attention à l'ordre ! ! !**. Dans les deux cas, il ne faudra pas oublier de fermer également le descripteur original (celui qu'on vient de dupliquer), puisqu'il ne sert alors plus à rien.

4 Implémentation de commandes

Exercice 6 Écrire un programme qui exécute `ls -l > toto` (exercice vu en TD).