Systèmes d'exploitation
TD 6: Système de fichiers UNIX.

Frédéric Vivien, bureau 306, e-mail: fvivien

18 mars 1997

Note: se référer au `man` pour la syntaxe et l'utilisation des fonctions évoquées dans ce TP.

Mécanismes de protection

La commande shell *umask* permet d'indiquer les droits d'accès aux fichiers ultérieurement crées. La commande *chmod* permet de modifier les droits d'accès d'un fichier.

Exercices

Les exercices suivant demandent la participation de deux utilisateurs.

1. Écrire un programme qui renvoie le numéro et le nom de l'utilisateur «effectif» et de l'utilisateur «réel». Se servir des fonctions `getuid` et `getpwuid` et de leurs variantes.

2. Un utilisateur α recopie chez lui le fichier `/bin/sh`. Il en modifie les droits d'accès en positionnant le bit *set-uid*. Il crée un fichier interdit en lecture et en écriture à tout autre que lui. L'utilisateur β exécute le programme *sh* modifié, puis essaye de modifier le fichier de α. Que se passe-t-il? Que renvoie le programme précédent? Que se passerait-t-il si le bit *set-uid* du shell standard était positionné?

   N.B.: ne laissez pas traîner le fichier *sh*.

3. Écrire un programme C et un programme shell qui affichent «bonjour» à l'écran. Définir les droits d'accès aux deux programmes de sorte que seul l'utilisateur α puissent les lire mais que les deux utilisateurs puissent les «exécuter». L'utilisateur β essaye d'exécuter les deux programmes. Commentaires?

Liens

Nous manipulons ici les liens physiques (commande `ln`) et les liens symboliques (commande `ln -s`).

Utilisez les commandes `ls`, `mv`, `cp`, `ln`, `chmod`, `cat` sur un fichier, un lien physique sur celui-ci, et un lien symbolique. Quelles sont les différences?

Fonction `fseek`

La fonction système `fseek` permet de modifier la position courante dans un fichier.

1. Écrire au moyen de la fonction `stat` un programme qui renvoie les principales données concernant un fichier: numéro d'inode, taille, taille des blocs, nombre de blocs alloués, etc.

2. Écrire un programme qui crée un fichier et qui écrit un mot après avoir déplacé le pointeur sur le fichier de 256 Ko (par exemple). Visualisez le fichier au moyen des commandes `cat` et `od -c`, et sa taille au moyen du programme précédent et des commandes `du -a` et `ls -l`. Comment UNIX gère-t-il ce type de fichiers?

Tubes de communication

Un tube (*pipe*) est le moyen normal offert par UNIX pour la communication entre processus.

Un tube est créé par un appel à la fonction `pipe`, il est lu au moyen des fonctions `read` et `write`. Il est fermé (seulement en lecture ou seulement en écriture) par la fonction `close`.

Écrire un programme qui crée un tube et un processus fils. Faites écrire le père dans le tube, et lire le fils. Que se passe-t-il si le fils veut lire alors que le tube est vide? Un tube est de taille bornée. Que se passe-t-il si le père veut écrire dans un tube plein? Que se passe-t-il si le père essaye d'écrire dans le tube avant que le fils ne soit créé?

Un tube ordinaire n'a pas de références dans le système de fichiers. Il a par contre un certain nombre de caractéristiques auxquelles ont peu accéder au moyen de la primitive `fstat`. Affichez ces caractéristiques.

Un tube ordinaire ne permet la communication qu'entre deux processus ayant un ¡«lien de filiation». Les tubes nommés permettent à deux processus créés indépendamments de communiquer. Reprendre l'exercice avec deux processus indépendants et un tube nommé crée au moyen de la commande `mknod`.