TD de Système n° 8

Exercice 1 : "filtremod"

On souhaite écrire une commande permettant de donner aux différents types d'utilisateurs des droits « raisonnables » sur les fichiers d'une arborescence donnée, pour éviter par exemple des aberrations du style :

- droits plus restrictifs pour le propriétaire que pour son groupe ou pour les autres ;
- fichier ou répertoire autorisé en lecture à certains utilisateurs dans un répertoire qui leur est inaccessible ;
- fichier de type « exécutable » interdit en exécution à des utilisateurs ayant par ailleurs le droit de le lire...

1. Écrire une fonction `int elargir_droits(char *fichier)` qui accorde à chaque catégorie d'utilisateurs les droits accordés à une catégorie « inférieure ».
2. Écrire une fonction `int restreindre_droits(char *fichier)` qui retire à chaque catégorie d'utilisateurs les droits qui ne sont accordés à aucune catégorie « supérieure ».
3. Écrire un programme permettant d'appliquer l'une des deux fonctions précédentes à tous les fichiers d'un répertoire, en fonction d'un paramètre de la ligne de commande.
4. Sous Unix, les fichiers binaires exécutables sont au format ELF, et commencent donc [1] tous par les quatre caractères suivants : le caractère de code ASCII 127 (7f en hexadécimal, 177 en octal), puis les caractères `'E'`, `'L'` et `'F'`.
   Écrire une fonction `int est_de_type_executable(char *fic)` qui détermine si un fichier a vocation à être exécuté, puis modifier votre programme pour qu'il puisse accorder des droits en exécution pertinents sur les fichiers d'un répertoire.

Exercice 2 : processus en cascade

Écrire deux programmes qui créent n processus p_i tels que :

1. pour tout i entre 2 et n, p_i soit le fils de p_{i-1} ;
2. pour tout i entre 2 et n, p_i soit le fils de p_1.

Chaque processus devra afficher son identifiant et celui de son père.

Exercice 3 : Un ordonanceur

- Écrire une fonction `void ordonanceur1(int n)`, qui devra ordonancer `n` tâches (supposément $0, \ldots, n-1$). Pour exécuter une tâche, on utilisera la fonction supposée exister `char execute(int tache)` qui renvoie le laps de temps (en secondes, inférieur à 4 minutes) que la tâche `tache` tient à voir s'écouler avant d'être réexécutée. On dispose de la fonction `int attendre(int duree)` qui attend `duree` secondes et renvoie `duree`. Notre ordonanceur commencera par lancer toutes les tâches simultanément.
- Cette fois, au lieu de renvoyer un laps de temps, `execute` renvoie l'heure à laquelle la tâche veut se réexécuter. On dispose de la fonction `int sonne()` qui "sonne l'heure" : elle renvoie, à l'heure pile, l'heure qu'il est. Implémentez `void ordonanceur2(int n)`. Si une tâche termine à 8h35 et demande à être réexécutée à 8h, il faut la relancer instantanément. Là encore notre ordonanceur commencera par lancer toutes les tâches simultanément. On suppose que votre ordonanceur est lancé à minuit (0h).

1. pour en savoir plus, `man magic` ou `man file`