Plusieurs à éditer un fichier

Louis Granboulan

27 février 2001

1 Motivations

On suppose qu'un petit groupe d'utilisateurs travaille en commun à modifier un (ou quelques) fichier(s). Il s'agit de proposer et de programmer un moyen de réaliser ceci.

Il y a principalement deux sortes de solutions :
- donner au fichier des droits Unix qui limitent les droits en écriture aux utilisateurs concernés ;
- réaliser un programme spécifique pour éditer ce fichier.

Une première idée est de créer un compte spécifique (e.g. `gourous`) qui sera propriétaire et aura le droit d'écrire dans ce fichier, et de donner le mot de passe de ce compte à tous les utilisateurs concernés. Pour un petit nombre de fichiers, c'est écraser une mouche avec un immeuble.

Une variante est de créer un groupe unix dont seront membres tous les utilisateurs concernés. Le fichier sera autorisé en écriture pour les membres du groupe.

Le problème pour ces deux solutions est qu'elles ne permettent pas de s'assurer que deux utilisateurs n'éditent pas le fichier simultanément. De plus, il faut être `root` pour créer un compte ou un groupe.

La seconde idée est de créer un programme spécifique d'édition de ce fichier. Ce programme aura les droits d'écriture (à l'aide un bit s par exemple) et vérifiera l'identité de l'utilisateur avant d'autoriser l'édition. On peut imaginer que ce programme est configurable et que le fichier de configuration associé connait, pour chaque fichier, la liste des utilisateurs autorisés.

L'avantage est que n'importe qui peut mettre en place ce système et qu'il est possible d'y rajouter une vérification pour éviter les accès simultanés (`lock`). Un inconvénient est qu'à moins de créer un compte spécifique qui ait les droits d'écriture dans le fichier, le propriétaire du fichier (a priori l'un des utilisateurs) a la possibilité de contourner le système.

NB : en pratique, on préfère n'utiliser aucun des systèmes ci-dessus, mais plutôt quelque chose comme `cvs`.

2 Exercices

Les programmes ci-dessous peuvent être réalisés indifféremment en C, en `perl` ou en shell. Pour changer des TP précédents, on préférera le C...

2.1 Vérification d'appartenance à la liste des utilisateurs autorisés

Le programme (qu'on va appeler `modif`) est appelé avec un argument, le nom du fichier qu'on veut éditer. Il lit la liste des utilisateurs autorisés dans son fichier de configuration (par exemple `.modifrc`) et renvoie un message qui dit si la modification de ce fichier est autorisée pour l'utilisateur qui a exécuté le programme.

Attention. Pour des raisons de sécurité, le fichier `.modifrc` se trouve dans le "homedir" du propriétaire du programme `modif`, et n'est lisible que par celui-ci. NB : faites des vérifications en collaboration avec votre voisin !

2.2 Modification du fichier

Une première solution est de programmer dans `modif` un éditeur de texte. C'est un peu fastidieux. Une seconde solution est d'utiliser l'éditeur préféré de l'utilisateur (variable d'environnement `VISUAL` ou `EDITOR`). Le programme `modif` doit donc appeler un programme extérieur. On utilise `system` ou `fork` et `exec` (vous verrez ces fonctions en détail plus tard).

Modifier le programme `modif` de l'exercice précédent pour qu'il appelle un éditeur. Réfléchir aux problèmes de sécurité. Regarder par exemple la description du shell `/usr/lib/rsh`.

2.3 Verrouillage du fichier

Il existe des primitives pour permettre le verrouillage d'un fichier et éviter l'édition simultanée. Regarder `flock` et `fcntl`. Une autre solution, a priori plus simple à implanter, est de créer un fichier qui s'appelle `filename.lock` et dont l'existence indique que le fichier est en cours de modification[1].

Comparer ces méthodes et programmer l'une d'entre elles.

[1] C'est par exemple ce que fait `netscape` pour ne pas être lancé plusieurs fois.