Chapitre 14

Sécurité et Sûreté de fonctionnement

Comme pour une habitation il faut que votre système offre deux chose importante, d'une part que la vie dans l'habitation soit sur, pas de risque pour les utilisateurs ni pour les éléments matériel. Feux , indondation, enfermement, tremblement de terre etc. C'est la sûreté de fonctionnement.
D'autre par l'habitation est protégée ainsi que ses habitants contre des attaques plus ou moins malveillantes. La porte du jardin est fermée pour éviter que des animaux détériore le jardin. L'habitation est munie de systéme de vérouillage pour se proteger contre un cambriollage. C'est la sécurité.

La sûreté de fonctionnement est un élément statégique qui doit être gérer par la direction informatique en fonction de contraintes opérationnelles. La sécurité est le problème de tout le monde. Pour que la sécurité fonctionne, il faut que toutes les personnes ayant un accès à une ressource soient conscient du degré de sécurité associé à la ressource. La stratégie de sécurité doit bien sur être définie par la direction informatique mais c'est un travail collectif (installation d'une sérure n'a pas d'effet si tout le monde laisse la porte ouverte).

14.1 Protection des systèmes d'exploitation

Sécuriser un système, c'est protéger ce système contre un fonctionnement imprévu ou défectueux.

Il peut s'agir :
- d'erreurs de programmation (d'un utilisateur, ou du système lui-même) qui se propagent au système (du fait de contrôles insuffisants ou mal effectués).
- d'un mauvais fonctionnement du matériel.
- enfin, d'un opérateur, concepteur ou réalisateur malveillant ou peu scrupuleux (quand il s'agit d'informations financières !).

Le recensement des opérations frauduleuses aux Etats-Unis au cours d'une année a donné 339 cas de fraude, pour un coût d'un milliard de francs.

La protection des sites a également un coût très important (temps et complexité), d'où des systèmes de protection qui résultaient d'un compromis coût/efficacité.

Le coût en ressources de la protection étant resté stationnaire, les systèmes et les machines actuelles plus rapides ont rendu ce coût moins prohibitif.
L'idée d'un système de protection est de traiter les différents types de problèmes de manière générale et unitaire.

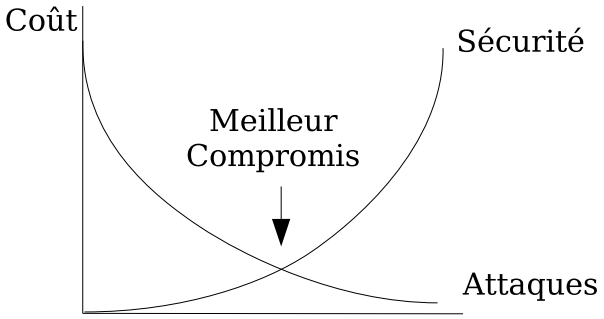

FIG. 14.1 – Un compromis entre le cout d'une attaque et celui de la sécurité

Implantés seuls, les dispositifs de protection coûtent cher.

Heureusement, si ces dispositifs permettent d'augmenter les performances du logiciel, dans des domaines comme celui de la fiabilité ou de la résistance aux erreurs, leur coût relatif diminue. Si, de plus, ces dispositifs permettent une gestion des ressources partagées plus facile et plus sûre, ils peuvent devenir compétitifs d'un point de vue commercial.

Il est difficile de définir précisément ce que l'on entend par protection d'un système d'exploitation (et d'information en général), tant les facteurs qui peuvent influer sur cette notion (humains, sociaux, économiques), sont nombreux. On peut dire cependant que la protection se rapporte à tout ce par quoi l'information peut être modifiée, divulguée ou détruite. Dans certains cas, la gestion du trafic aérien par exemple, elle peut être la garantie des performances du système. La confidentialité d'enregistrements financiers, médicaux ou personnels relève aussi de la protection, comme le fait qu'un processus utilisateur ne puisse être exécuté en mode système. La protection exige enfin la correction des processus système.

- pérennité du système
- confidentialité des données (système, utilisateur, etc.)
- correction du système

A l'opposé, nous ne parlerons pas de :

- protection physique de l'ordinateur (feu, vol, coupures, etc.)
- malveillance ou incompétence de l'opérateur (il est éventuellement possible de limiter soigneusement les privilèges du super-utilisateur afin de préserver le système).

Le degré de protection du système dépend de deux facteurs :

- le degré de protection des informations qu'il manipule
- le degré de confiance en ses logiciels, en particulier le système d'exploitation.

Un logiciel est fiable quand il satisfait correctement ses spécifications et quand, de plus, il est capable de résister à un environnement imprévu (données erronées, pannes, etc.), soit en corrigeant l'anomalie, soit en la signalant, mais en évitant que les erreurs ne se propagent et ne contaminent le système tout entier.
La protection, l'intégrité et l'authenticité des données qui transitent dans un système d'information sont réalisées par les systèmes cryptographiques (ATHENA et Kerberos au MIT).
Le confinement des erreurs est obtenu en contrôlant les accès aux entités du système d'exploitation, par les domaines de protection.

14.2 Généralités sur le contrôle d'accès

Contrôle très précis de l'utilisation des ressources par les processus.

Sujets \ Objets	Fichier 1	Segment 1	Segment 2	Processus 2	Editeur
Processus 1	Lire	Executer	Lire Ecrire		Entrer
Processus 2	Lire Ecrire				Entrer
Processus 3		Lire Ecrire Executer		Entrer	Entrer

FIG. 14.2 – Matrice d'accès

Deux niveaux :
- un niveau logique (soft), celui du modèle de protection, ensemble de règles qui définissent quels accès (aux ressources) sont autorisés et quels accès sont interdits. Ces règles sont définies soit à la conception du système, soit par les utilisateurs.
- un niveau matériel qui permet d'appliquer le modèle réellement. C'est le rôle des mécanismes de protection.

Le premier doit être dynamique. Par contre, le deuxième doit être stable pour faciliter l'implémentation, le contrôle et la fiabillisation.
Les deux doivent de surcroît être indépendants du modèle pour offrir un vaste ensemble de règles possibles.
Un exemple de protection simple est celui des répertoires sous unix, pour éviter qu'ils soit corompus (ou que l'arborescence soit corompue), il sont identifiés comme des fichiers spéciaux et il n'est possible d'y accèder que par le truchement d'appels systèmes spécifiques. Biensur il est toujours possible de les manipuler si l'on peut accèder en mode **raw** au disque dur mais cette option est réservée au super utilisateur.

14.2.1 Domaines de protection et matrices d'accès

On formalise le système comme un ensemble d'entités actives, les sujets, un ensemble d'entités accessibles, les objets. Le modèle de protection définit quels sujets ont accès à quels objets et comment (modalités d'accès).

On parle alors de droit d'accès, définis par le couple (objet, modalités)

Exemple : (fichier, lire)

Le modèle doit fixer à tout instant les droits d'accès dont dispose chaque processus. Cet ensemble de droits est le domaine de protection du processus. Voire un exemple de matrice d'accès dans la figure 14.2

14.2.2 Domaines de protection restreints

Il est souhaitable que la matrice d'accès puisse évoluer dynamiquement. En effet, un même processus peut avoir, au cours de son existence, des besoins variables afin que chaque module

qui compose un processus ne mette pas en danger des ressources non utilisées. Par exemple : un module de lecture de données, un module de calcul, un module d'impression. On va donc exécuter chaque module dans un domaine de protection le plus réduit possible.

C'est le *principe du moindre privilège* : un programme ne peut endommager un objet auquel il n'a pas accès !

Pour mettre en place ces domaines dynamiques, une possibilité est de changer les droits d'accès du processus au cours de son exécution. Une autre possibilité est d'ajouter aux objets le type "domaine" et de contrôler les accès à la matrice. L'édition de cases de la matrice devient une opération protégée.

14.2.3 Avantages des domaines de protections restreints

Avantages de cette souplesse :
- le maillon faible : un système rigide laisse souvent des "poternes" (portes dérobées) pour pouvoir implémenter certaines opérations ;
- si les mesures de protection sont trop pesantes, l'expérience prouve que l'on crée souvent des moyens "exceptionnels" pour les contourner ;
- il est intéressant de faire varier les contrôles suivant les utilisateurs ;
- on peut réaliser des accès à la carte sur certains objets ;
- enfin, certains problèmes de protection nécessitent des mesures souples, ce sont : "le cheval de Troie" et le confinement.

14.3 Le cheval de Troie

Un utilisateur fait souvent appel à un certain nombre de programmes qu'il n'a pas écrit lui-même (heureusement), un éditeur par exemple. Ce programme peut être un cheval de Troie : il va profiter des droits donnés par l'utilisateur pour consulter, copier, modifier ou altérer des données auxquelles il n'est pas censé accéder.

14.4 Le confinement

Le problème ici est tout simplement le fait que le programme ne manipule pas de données de l'utilisateur mais simplement enregistre ses paramètres d'appels (les utilisateurs à qui vous envoyez du courrier par exemple). Le problème du confinement est donc de vous protéger contre ce type d'extraction d'informations (ce qui peut par exemple être utilisé en bourse pour connaitre votre comportement d'achat).

14.5 les mécanismes de contrôle

Accès hiérarchiques
UNIX (4)/ MULTICS (8) / VMS
Listes d'accès
UNIX/MULTICS
Capacités

Les capacités sont des triplets (UTILISATEUR, DROITS, POINTEUR). La manipulation des capacités est réalisée de façon protégée. Le pointeur n'est pas directement utilisable par l'utilisateur de la capacité. La capacité donne le droit d'accès à certains utilisateurs d'une certaine ressource. Pour qu'un autre utilisateur puisse utiliser votre ressource, vous devez lui donner une capacité.

Changer de protection revient à changer de C-liste.

La notion de domaine se matérialise par une simple indirection sur une autre C-liste.

Comme les capacités donnent un accès sans contrôle aux objets, la protection des capacités doit être absolue. Elle est donc réalisée de façon matérielle.

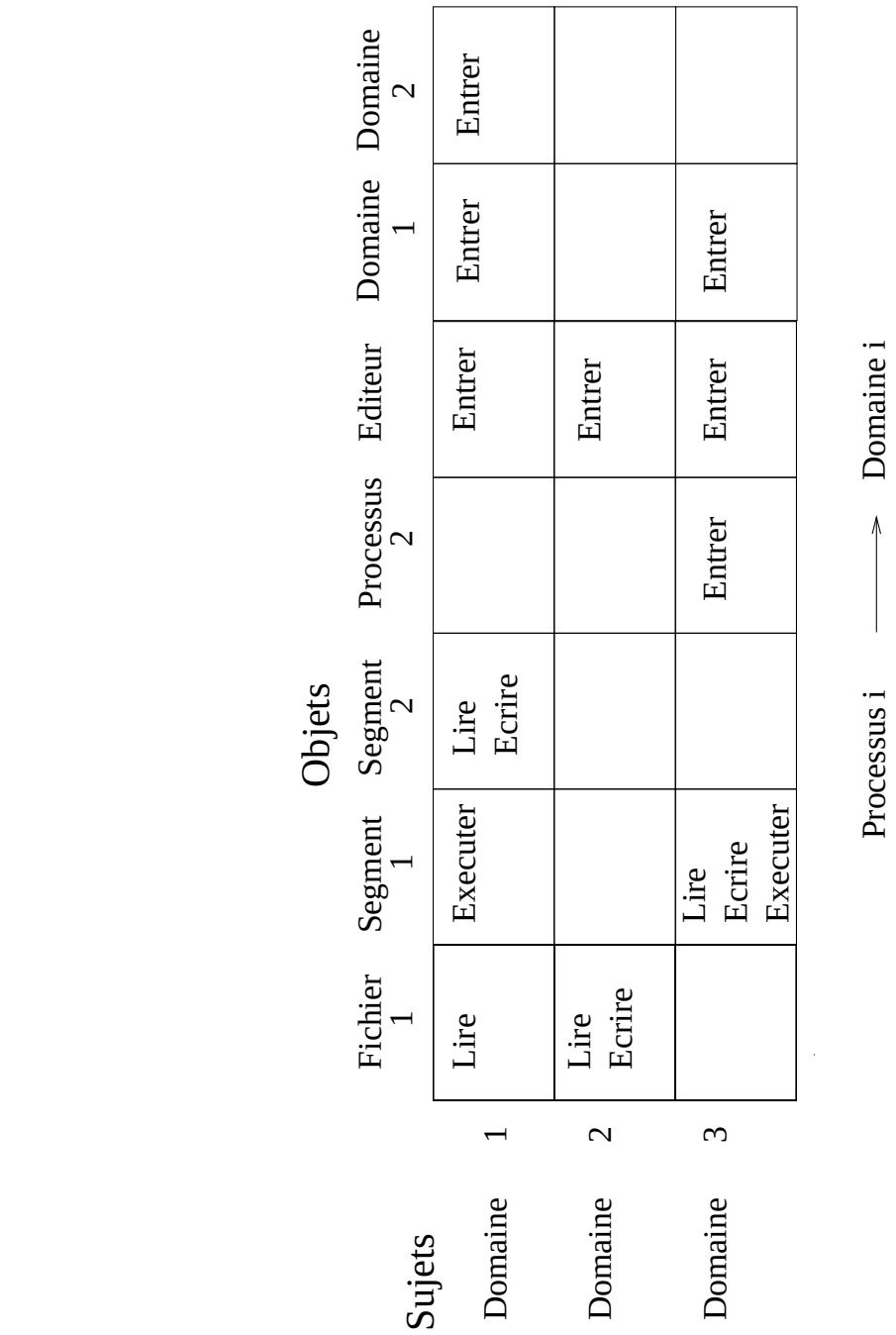

FIG. 14.3 – Matrice d'accès

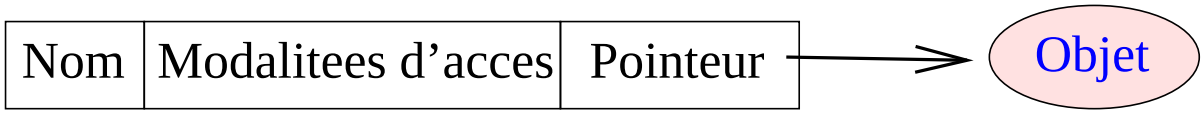

FIG. 14.4 – Une capacité

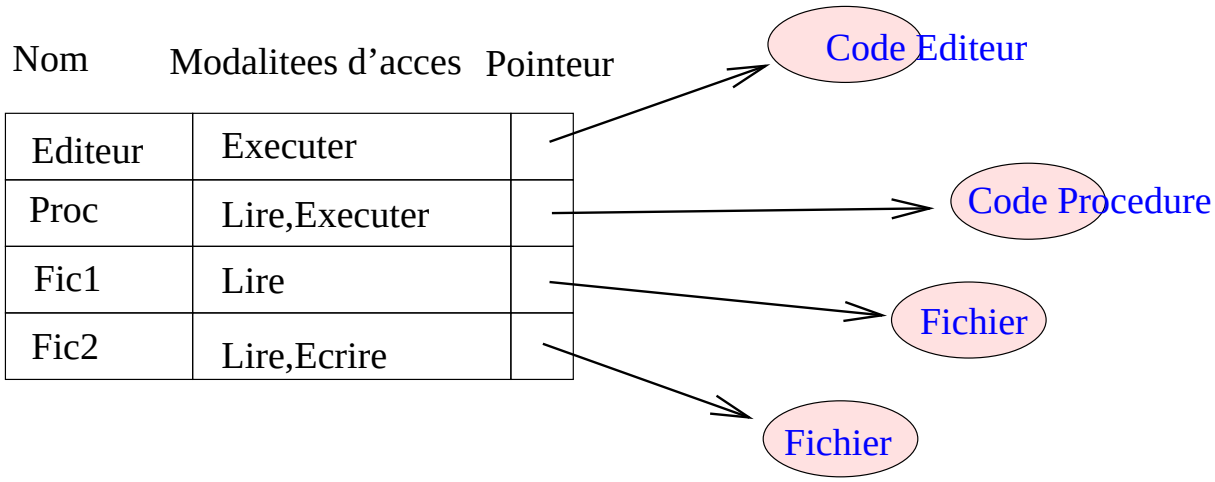

FIG. 14.5 – Une liste de capacités

14.5.1 Application des capacités au domaines de protection restreints

Les C-listes sont des objets d'un type n'ayant qu'un droit d'entrée, la C-liste contenant le droit réel.

Cette technique sur les C-listes permet d'implanter facilement le principe de moindre privilège.

Les mécanismes d'accès mémoire modernes permettent aisément de réaliser les capacités.

Un problème important est la révocation

En effet, une fois que vous avez donné une capacité, l'accès est définitivement donné. Pour régler ce problème, on ne fournira pas la capacité d'accès à un objet mais à un domaine, et on détruira ce domaine si l'on veut de nouveau interdire l'accès à l'objet. On crée deux capacités en chaine et l'on détruit celle que l'on possède quand ont veut retirer l'accès.

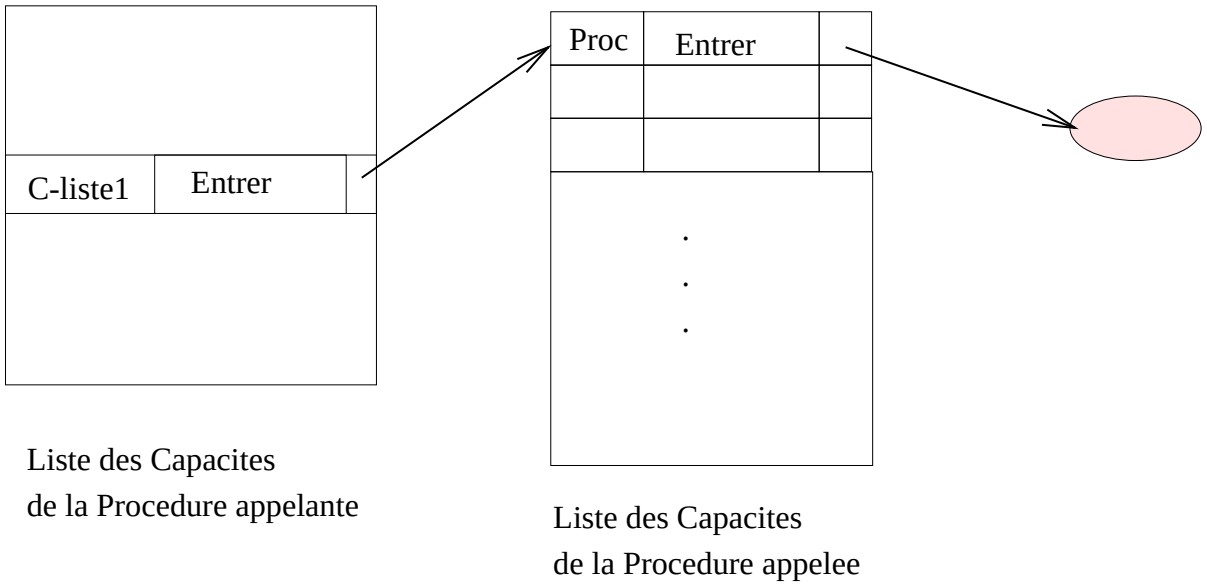

FIG. 14.6 – Changement du domaine de protection

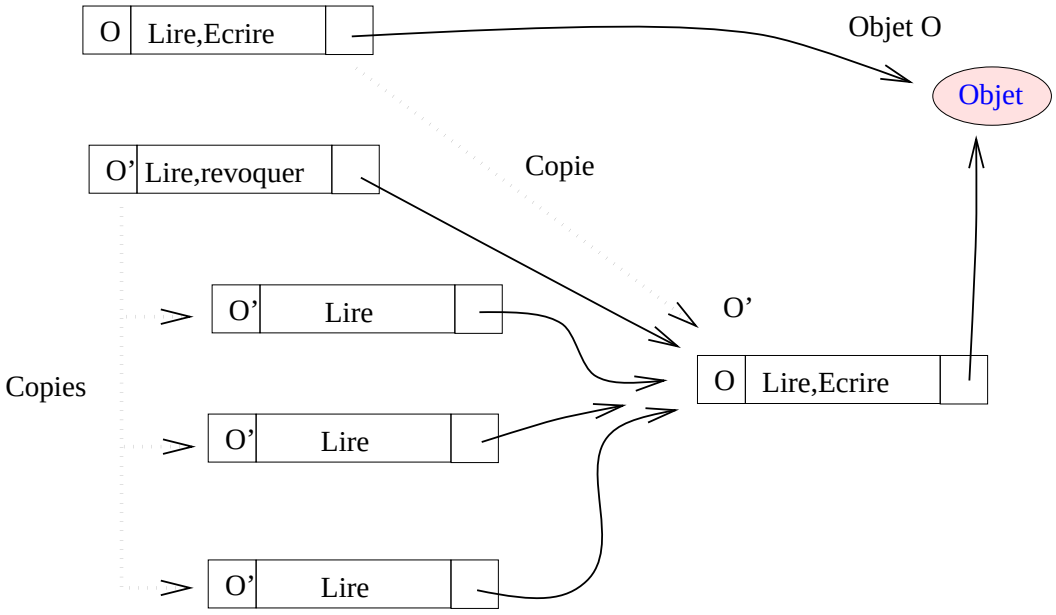

FIG. 14.7 – Transmission d'une capacité

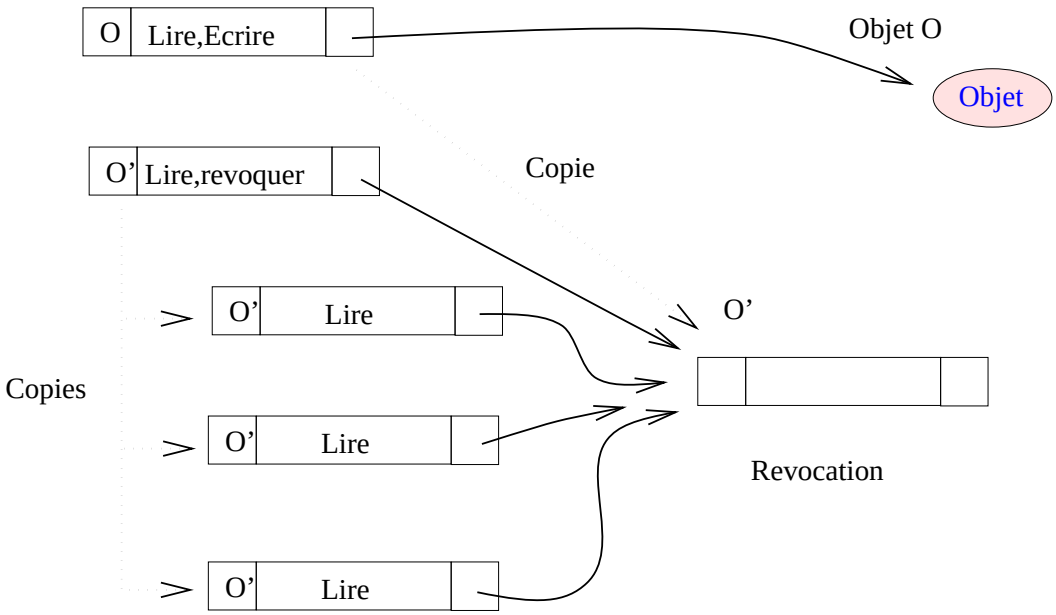

FIG. 14.8 – Révocation d'une capacité

14.6 Les ACL

Les ACL (access control lists) sont une extension des modes de protection standard d'UNIX. Les ACL sont des droits que l'on définit en plus des 9 bits de protection classiques, ils permettent en particulier d'autoriser l'accès ou de le refuser, à un utilisateur donné, ou à un groupe donné.

Deux commandes permettent de manipuler les ACL, ce sont `chacl` et `lsacl`.
La syntaxe de la commande shell `chacl` :

```
chacl '(dr.staff,r-x)(zipstein.%,r-x)(%.licence,---)' proj
```

qui donne sur le fichier `proj` les droits de lecture et d'écriture à l'utilisateur `dr` du groupe `staff` et à l'utilisateur `zipstein` quelque soit son groupe et qui refuse cet accès aux utilisateurs du groupe `licence`.

```
chacl '(binome.%,rwx)(%.@,--x)(%.%,---)' catalogue_projet
```

qui donne le droit d'accès total à l'utilisateur `binome` (quelque soit son groupe), permet le parcours du répertoire aux membres du groupe propriétaire et refuse l'accès à tous les autres utilisateurs.

Deux symboles spéciaux :

`%` pour n'importe qui (utilisateur ou groupe)

`@` pour le propriétaire ou le groupe propriétaire

On retrouve aussi les autres syntaxes de `chmod` par exemple :
`chacl %.%=r fichier`
ou
`chacl @.%=5 fichier`

Attention les acl sont détruits par la commande `chmod` et la commande `chacl` ne permet pas de positioner les autres bits définis dans l'inode ; seuls les 9 bits de protections sont positionnables par `chacl`.

Pour positionner les droits standard et des acl, il faut donc réaliser en succession un `chmod` puis un `chacl`.

On utilisera :
`chacl '(prof.%,rwx)' catalogue_projet`
pour les projets de C ou de système.

La commande `lsacl [fichiers]` permet de connaître les acl associés aux fichiers, remarquer qu'à l'inverse de `/bin/ls` cette commande n'a pas de paramètres par défaut.

14.6.1 Appels systemes `setacl` et `getacl`

On trouvera deux appels systèmes correspondant :

```
#include <sys/acl.h>

int setacl(
     const char *path,
     size_t nentries,
     const struct acl_entry *acl
);

int fsetacl(
     int fildes,
     size_t nentries,
     const struct acl_entry *acl
```

```
);
```

Un bon exercice : récrire `lsacl` de façon qu'il fonctionne d'une manière similaire à `/bin/ls`. Utilisation de la commande script pour montrer le comportement des acl.

```
Script started on Fri May  5 10:33:20 1995
$ lsacl *
(dr.%,rw-)(%.staff,---)(%.%,---) fich
(dr.%,rw-)(%.staff,---)(%.%,---) file
(dr.%,rwx)(%.staff,---)(%.%,---) projet
$ chacl  '(prof.%,rwx)' fich
$ lsacl *
(prof.%,rwx)(dr.%,rw-)(%.staff,---)(%.%,---) fich
(dr.%,rw-)(%.staff,---)(%.%,---) file
(dr.%,rwx)(%.staff,---)(%.%,---) projet
$ chacl '(%.staff,rx)' fich
$ lsacl *
(prof.%,rwx)(dr.%,rw-)(%.staff,r-x)(%.%,---) fich
(dr.%,rw-)(%.staff,---)(%.%,---) file
(dr.%,rwx)(%.staff,---)(%.%,---) projet
$ chacl '(illouz.staff=' fich
$ lsacl fich
(illouz.staff,---)(prof.%,rwx)(dr.%,rw-)(%.staff,r-x)(%.%,---) fich
$ chacl '(prof.%,rx)' . ..
$ su prof
Password:
$ cat fich
$ touch fich
$ chacl '(dr.staff,x)' fich
chacl: file "fich": Not owner (errno = 1)
$ lsacl *
(illouz.staff,---)(prof.%,rwx)(dr.%,rw-)(%.staff,r-x)(%.%,---) fich
(dr.%,rw-)(%.staff,---)(%.%,---) file
(dr.%,rwx)(%.staff,---)(%.%,---) projet
$ exit # du su
$ exit # du script

script done on Fri May  5 10:37:18 1995
```

14.6.2 Autres pistes sur la sécurité

- crack et autres logiciels d'attaque de mots de passe
- root-fix et autres Pots de Miel
- virus et logiciels antivirus (sous linux ? ? ? ?)
- Gestion des mots de passe
- Honey Pot
- Haute disponibilité
  - Redondance
  - RAID (cf chapitre)
  - Distribution de disques
  - SAN
  - Clusters
  - keep alive
- Sauvegardes et systèmes de sauvegarde

– Les attaques par déni de services