Permissions de fichiers

Description des permissions de fichiers

Structure des bits de mode de fichier

Les bits de mode de fichier ont 2 parties, les bits de permission de fichier et les bits de mode spéciaux.

Il y'a 3 types de permission que l'utilisateur peut avoir pour un fichier :

1. permission de lire le fichier. pour les répertoires, permission de lister le contenu.
2. permission d'écrire dans le fichier. pour les répertoire, permet de créer et supprimer des fichiers.
3. Permission d'exécuter le fichier. pour les répertoires, permet d'accéder aux fichiers.

Il y'a 3 catégories d'utilisateurs que peuvent avoir différentes permissions :

1. le propriétaire du fichier
2. les autres utilisateurs qui sont dans le groupe du fichier
3. Tous les autres.

Les fichiers ont un propriétaire et un groupe quand ils sont crées. Généralement le propriétaire est l'utilisateur courant et le groupe est le groupe du répertoire, mais celà dépend des systèmes, le système de fichier, et la manière dont il est créé.

En plus des 3 catégories de permission, les bits de mode ont 3 composants spéciaux, qui affectent seulement les fichiers exécutables et les répertoires :

1. Définis l'ID de l'utilisateur effectif du processus, appelé le bit **set-user-ID** ou bit **setuid**. Pour les répertoires, donne aux fichiers crées le même propriétaire, et définis le **set-user-ID** pour les sous-répertoire crées.
2. Définis l'ID du groupe effectif du processus, appelé le bit **set-group-ID** ou bit **setgid**. Pour les répertoires, donne aux fichiers crées le même groupe, et définis le set-group-ID pour les sous-répertoire crées.
3. Empêche les utilisateurs non privilégiés de supprimer ou renommer un fichier dans un répertoire à moins qu'ils aient leur propre fichiers dans ce répertoire. appelé le **restricted deletion flag**. Pour les fichiers réguliers sur de vieux systèmes, sauver l'image du programme dans le périphérique swap pour qu'il se charge plus rapidement. Appelé le **sticky bit**.

En plus des bits de mode de fichiers ci-dessus, il y'a des attributs de fichiers spécifiques aux systèmes de fichier. Par exemple les ACL, si un fichier est compressé, si un fichier peut être modifié, et si un fichier peut être dumpé. ex :

ext2 Les attributs spécifiques à ce système de fichier se paramètre avec **chattr**

FFS Les attributs spécifiques à ce système de fichier se paramètre avec **chflags**

Modes Symboliques

les modes symboliques représentent les changements des bits de mode des fichiers. Le format des modes symboliques est :

[ugoa...][+-=]PERMS...[,...]

u Le propriétaire du fichier

g Les utilisateur du groupe du fichier

o tous les autres

a tous les utilisateurs, identique à ugo

+ Ajouter les permissions

- supprimer les permissions

= définir les permissions

r permission en lecture

w permission en écriture

x permet d'exécution.

Exemples

donne accès en lecture/écriture pour tous les utilisateurs :
a=rw
Supprime les permissions en écriture à tous les utilisateurs autre que le propriétaire :
go-w
supprimer tous les droits à tous le monde excepté le propriétaire :
go=
ou
og-rwx

Copier les permissions existantes

Il est possible de baser les permissions d'un fichier sur des permissions existantes. Il faut utiliser **u**, **g** ou **o**. Par exemple le mode **o+g** copie les permissions du groupe aux autres utilisateurs.

Changer les bits de mode spéciaux

Pour définir le bits **setuid**, utiliser **u** pour la partie user et **s** dans la partie permissions

Pour définir le bits **setgid**, utiliser **g** pour la partie user et **s** dans la partie permissions

pour définir les bits **setuid** et **setgid**, omettre la partie user (ou utiliser **a**), et **s** dans la partie permissions

Pour définir le **restricted deletion flag** ou le **sticky bit**, omettre la partie **user** (ou utiliser **a**) et **t** dans la partie permissions

Exemples

définir le setuid :
u+s
supprimer le setuid et setgid :
a-s
définit le restricted deletion flag ou le sticky bit :
+t

Exécution conditionnelle

utiliser '**X**' au lieu de '**x**' affecte le droit d'exécution/recherche si le fichier est un répertoire ou avait déjà les permissions d'exécution. Par ex : **a+X** donne à tous les utilisateurs le droit de recherche dans les répertoires, ou d'exécuter les fichiers si quelqu'un pouvait les exécuter avant.

Effectuer plusieurs changements

Il y'a 2 manières d'effectuer des changement multiples.

La première manière est de spécifier plusieurs opérations. Par exemple, donner aux utilisateurs autre que le propriétaire la permission de lire et si c'est un répertoire ou si quelqu'un avait le droit de l'exécuter, de le faire également. et refuse l'écriture :
og+rX-w
La deuxième manière est de spécifier plus d'un mode symbolique simple. Par exemple, donner à tous le droit en lecture mais refuser l'écriture pour tous les utilisateurs hormis le propriétaire :
a+r,go-w
Définir explicitement toutes les permissions
u=rwx,g=rx,o=
Les 2 méthodes peuvent être combinées
a+r,g+x-W
Donner à tous les utilisateurs le droit en lecture, donner au groupe le droit d'exécution/recherche, mais pas en écriture
u+r,g+rx,o+r,g-w

Umask et protection

Si la partie user d'un mode symbolique est omis, le défaut est **a**, excepté pour les permissions qui sont définies dans la variable système **umask**. La valeur de **umask** peut être définie en utilisant la commande **umask**. Sa valeur par défaut varie d'un système à l'autre.

Omettre la partie user d'un mode symbolique n'est généralement pas très utile sauf avec '**+**', cela permet d'utiliser **umask** comme protection facilement personnalisable.

Par exemple, si umask a la valeur 2, qui supprime les permissions en écriture pour les utilisateurs qui ne sont pas dans le groupe du fichier, alors le mode
+w
ajoute la permission au fichier et son propriétaire et au groupe, mais pas aux autres utilisateurs
a+w
ignore umask, et donne le droit à tous les utilisateurs

Modes numériques

Il est possible de spécfier le mode symbolique en octal. Ce nombre est toujours interprété en octal. Il n'est pas necessaire de rajouter des 0, comme en C. 0055 est le même que 55.

Tous les autres utilisateurs
1 Exécution/recherche
2 Écriture
4 Lecture
Les utilisateurs dans le groupe du fichier
10 Exécution/recherche
20 Écriture

40 Lecture

Le propriétaire

100 Exécution/recherche

200 Écriture

400 Lecture

bits de mode spéciaux

1000 Restricted deletion flag ou sticky bit

2000 Set group ID

4000 Set user ID

Répertoires avec Set-User-ID et Set-Group-ID

Sur beaucoup de systèmes, si un répertoire a le **setgid** de mis, les fichiers nouvellement crées héritent du même groupe que le répertoire, et les sous-répertoire nouvellement crées héritent du **setgid**. Sur certains systèmes, le **setuid** a le même effet. Ce mécanisme permet aux utilisateurs de partager des fichiers facilement. Les commandes comme **chmod** et **mkdir** effacent ces bits généralement.

ces commandes laissent le setuid et setgid des sous-répertoires :
mkdir A B C
chmod 755 A
chmod 0755 B
chmod u=rwx,go=rx C
mkdir -m 755 D
mkdir -m 0755 E
mkdir -m u=rwx,go=rx F
ces commandes essayent de définit setuid et setgid sur les sous-répertoires :
mkdir G H
chmod 6755 G
chmod u=rwx,go=rx,a+s H
mkdir -m 6755 I
mkdir -m u=rwx,go=rx,a+s J
cette commande essaye de supprimer setuid et setgid du répertoire D :
chmod a-s D