

## Les permissions sur les fichiers

par Jean-Christophe

Dis Papa, c'est quoi `rwrx-xr-x` ? Tais-toi et nage !

---

### Un peu de théorie

Dans ce chapitre, nous allons étudier les **permissions sur les fichiers**. Nous allons voir rapidement sur quoi la gestion des permissions se base.

#### Les utilisateurs et les groupes

Les fichiers appartiennent à un **utilisateur** à l'intérieur d'un **groupe d'utilisateurs**.

- L'**utilisateur**, c'est vous ! C'est un identifiant avec un mot de passe, qui sont définis dans le fichier `/etc/passwd`. On peut ajouter des utilisateurs à l'aide de la commande `useradd` ou `adduser` (voir `man adduser`). Certaines distributions fournissent des outils graphiques pour ce faire, comme `drakuser` de Mandrake.
- Le **groupe d'utilisateurs**, défini dans le fichier `/etc/group`, permet de regrouper des utilisateurs dans des groupes (un même utilisateur peut faire partie de plusieurs groupes) afin par exemple de partager des permissions entre plusieurs utilisateurs. Par exemple, le groupe `quakeusers` peut être défini afin d'y placer les utilisateurs qui auront le droit d'utiliser le jeu Quake : on donnera par exemple la permission aux membres de ce groupe de lancer l'exécutable de ce jeu. De même, le groupe `cdrecording` aura les droits d'utiliser le graveur de CD.

#### Les droits possibles : R, W et X

Les droits que l'on peut attribuer à un fichier concernent :

- la lecture (**r** comme read),
- l'écriture (**w** comme write),
- l'exécution (**x** comme execute).

On peut attribuer ces droits pour :

- un utilisateur,
  - les membres d'un groupe,
  - le monde entier (i.e. les autres utilisateurs).
- 

### Visualiser/modifier les permissions

#### Ligne de commande

En ligne de commande (voir la rubrique [Shell](#)), tapez :

```
[username@taz username]$ ls -la
total 144
drwxr-xr-x 18 username users 2048 jan 7 19:22 .
drwxr-xr-x  7 root root 1024 fév 6 1996 ..
-rw-----  1 username users 147 jan 7 19:22 .Xauthority
-rw-r--r--  1 username users 1899 jui 28 21:01 .Xdefaults
-rw-----  1 username users 5860 jan 7 19:22 .bash_history
-rw-r--r--  1 username users 24 jui 28 21:01 .bash_logout
-rw-r--r--  1 username users 262 jui 29 18:15 .bash_profile
-rw-r--r--  1 username users 434 jui 28 21:01 .bashrc
-rw-r--r--  1 username users 2626 jui 28 21:01 .emacs
-rw-r--r--  1 username users 532 jui 28 21:01 .inputrc
drwxr-xr-x  3 username users 1024 jui 28 21:01 .kde
-rw-r--r--  1 username users 1546 jan 7 19:04 .kderc
-rwxr-xr-x  1 username users 1166 jui 28 21:01 .kderc.rpmorig
```

Pour une explication détaillée des différentes colonnes, voir la rubrique [Shell](#). Nous n'allons ici nous intéresser qu'aux éléments relatifs aux permissions.

1. La première colonne `-rw-r--r--` représente les permissions associées au fichier.

(le premier caractère est le type du fichier : un *d* pour un répertoire, un *l* pour un lien, etc.)

Ensuite, on a trois groupes de trois caractères : **rwX**. La présence de la lettre *r* *w* ou *x* accorde la permission, un tiret '-' la dénie.

**r** signifie : possibilité de lire ce fichier / dans ce répertoire,

**w** signifie : possibilité d'écrire dans ce fichier / répertoire,

**x** signifie : possibilité d'exécuter ce fichier / d'aller dans ce répertoire.

Les trois groupes de caractère s'appliquent, dans l'ordre, à :

1. l'utilisateur auquel appartient le fichier,
2. le groupe d'utilisateurs auquel est rattaché le fichier,
3. les autres utilisateurs.

```
- r w x r - x r - x
  uti-
  sateur  groupe  autres
```

2. La 3<sup>ème</sup> colonne est l'utilisateur à qui appartient le fichier. À cet utilisateur s'appliquent les permissions représentées par les trois premiers caractères de permissions de la première colonne (`-rwXr-xr-x`).

3. La 4<sup>ème</sup> colonne est le groupe d'utilisateurs auquel appartient le fichier. A ce groupe s'appliquent les permissions représentées par le deuxième groupe de trois caractères de permissions de la première colonne (`-rwXr-xr-x`).

### Exemples :

1. La ligne suivante :

```
drwxr-xr-x 18 username users 2048 jan 7 19:22 .
```

signifie pour le répertoire '.' (le répertoire HOME de l'utilisateur username) que tout le monde a le droit de lire le contenu du répertoire (le dernier 'r'), et que tout le monde peut y accéder (le dernier 'x'). Par contre, seul l'utilisateur username peut y écrire (caractère 'w'), c'est à dire y créer des fichiers, les modifier ou les supprimer.

2. La ligne :

```
-rw-r----- 1 username wwwadm 1728 jan 7 19:22 projet-www
```

signifie que seul 'username' et les utilisateurs du groupe 'wwwadm' peuvent lire ce fichier, que seule username peut le modifier, et que les autres utilisateurs n'ont aucun droit dessus (le dernier groupe de caractères '-----').

### Modification

#### Les droits : **chmod**

Il existe deux façon de changer les droits d'un fichier ou répertoire.

- Soit en précisant les droits en octal (base 8). La correspondance est la suivante :

- ◆ 1 : droit d'exécution
- ◆ 2 : droit d'écriture
- ◆ 4 : droit de lecture

Donc, pour préciser les droits en exécution et lecture, le chiffre octal est :  $1 + 4 = 5$ . Pour préciser les droits en exécution, écriture et lecture, le chiffre octal est :  $1 + 2 + 4 = 7$ .

Ensuite, il faut savoir que le chiffre des unités (en octal) correspond 'au reste du monde' que le chiffre des 'huitaines' (deuxième chiffre en octal) correspond 'au groupe' et que le chiffre des 'soixante-quatraines' (troisième chiffre en octal) correspond 'à l'utilisateur'. Ainsi, si on veut que l'utilisateur ait les 3 droits (rwX), que le groupe ai les 2 droits (r-x) et que le reste du monde n'ai aucun droit (---), le nombre octal est : 750 ( $7 = 1 + 2 + 4$  ;  $5 = 1 + 4$  ;  $0 = 0$ ). Pour donner ces droits à un fichier on tape alors :

```
[username@localhost ~] $ chmod 0750 /chemin/vers/fichier
```

Le premier 0 dans 0750 signifie qu'on donne le nombre en octal (sinon la correspondance en base 10 ou autre est complexe à déterminer).

- Soit en précisant les droits qu'on ajoute (+) ou soustrait (-) au fichier (ou répertoire). Pour ajouter des droits à l'utilisateur, on ajoute un droit (r,w, x ou toute combinaison des 3) à 'u'. Par exemple : `u+rw` signifie qu'on ajoute les droit de lecteur et d'écriture à l'utilisateur. Pour le groupe, on ajoute ou soustrait à 'g' et pour le reste du monde on ajoute ou soustrait à 'o' (other). Par exemple pour ajouter les droits de lecture et d'écriture à l'utilisateur on tapera :

```
[username@localhost ~] $ chmod u+rw /chemin/vers/fichier
```

Et :

```
[username@localhost ~] $ chmod go-rwx /chemin/vers/fichier
```

Pour retirer tous les droits (rwx) au groupe (g) et aux autres (a).

Remarque : à la place de u, g et o on peut utiliser 'a' (all) qui veut dire qu'on change les droits de tout le monde (utilisateur, groupe et reste du monde).

Il faut savoir que la commande peut s'appliquer de manière récursive (c'est bien pratique pour les répertoires), en lui passant l'argument '-R' et qu'à la place du droit 'x', on peut préciser 'X' (majuscule) ce qui signifie que parmi les fichiers et répertoires dont on modifiera les droits d'exécution, seuls les répertoires sont concernés. Ainsi :

```
[username@localhost ~] $ chmod -R u+rwX /chemin/vers/repertoire
```

rendra tous les fichiers contenus dans ce répertoire (ainsi que dans tous les sous-répertoires de celui-ci) lisibles et écrivables et que tous les sous-répertoires (et leurs sous-répertoires) seront 'navigables' (le droit d'exécution pour un répertoire autorise à se rendre dans le dit répertoire).

### La propriété : chown

Les droits de propriétés sont très simples à modifier. Il suffit de donner le nom du nouveau possesseur (et éventuellement le nom du nouveau groupe) et le nom du fichier. Ainsi :

```
[username@localhost ~] $ chown username /chemin/vers/fichier
```

donne le fichier /chemin/vers/fichier à l'utilisateur 'username'. Et :

```
[username@localhost ~] $ chown username.groupe /chemin/vers/fichier
```

donne le fichier /chemin/vers/fichier à l'utilisateur 'username' et au groupe 'groupe'. Comme pour `chmod` le paramètre '-R' permet de rendre récursive l'application de la nouvelle propriété (i.e. répertoires et sous-répertoires).

### Le groupe : chgrp

Le changement de groupe uniquement peut être obtenu par :

```
[username@localhost ~] $ chgrp groupe /chemin/vers/fichier
```

qui donne /chemin/vers/fichier au groupe 'groupe'. L'argument '-R' rend cette application récursive.

### Interface graphique

Dans votre gestionnaire de fichiers préféré, sous KDE ou Gnome par exemple, vous pouvez consulter et modifier les permissions sur un fichier en affichant la boîte de dialogue "Propriétés" du fichier/répertoire (en général par un clic droit).

### Pour aller plus loin ...

Il existe encore deux types de droits (en fait 3 mais le troisième est devenu obsolète) : 's' (SUID bit) et 'g' (SGID bit). Si l'on active le SUID bit d'un programme, il s'exécute sur le compte du possesseur du fichier (si c'est root, sur le compte de root, d'où un danger potentiel de sécurité). Si l'on active le SGID bit, c'est sur le compte de l'utilisateur normal, mais en tant que membre du groupe du fichier. Cela rend un fichier exécutable. À ce propos consultez l'article [SUID Scripts](#) par Xavier GARREAU sur Léa.

Ces droits s'octroient de la même manière que les autres. Par exemple :

```
# chmod +s /usr/bin/xmms
```

fera en sorte que xmms puisse obtenir les privilèges de root (le fichier /usr/bin/xmms appartenant à root. Un `ls -l` sur le fichier donnera ensuite :

```
-rwsr-xr-- 1 root  root  172812 dec 12 12:12 xmms
```

Dans ce cas particulier, cela permet à xmms d'obtenir une priorité temps réel qui peut être nécessaire pour un son parfait.

