

Les ChMod

Par Natim



www.openclassrooms.com

*Licence Creative Commons 7 2.0
Dernière mise à jour le 16/10/2012*

Sommaire


Sommaire	2
Les ChMod	3
Présentation du chmod	3
Modification des droits sur les fichiers	4
De manière octale	5
De manière symbolique	5
Les droits différents entre un répertoire et un fichier	6
Pour un fichier	6
Pour un répertoire	6
D'autres droits	6
Partager	7



Les ChMod



Mise à jour : 16/10/2012

Difficulté : Facile 



Vous avez sûrement entendu parler des **chmod** ! Cette commande qui pose souvent problème en PHP. Eh bien pour commencer, sachez que c'est de la faute de Linux.

Pour faire simple, je dirai seulement que Linux est un système multi-utilisateurs et donc qu'il ne faut pas que tous les utilisateurs puissent lire les fichiers de tous, sinon, l'intérêt est limité.

Les *chmod* définissent les droits d'un utilisateur sur un fichier.

Sommaire du tutoriel :



- [Présentation du chmod](#)
- [Modification des droits sur les fichiers](#)
- [Les droits différents entre un répertoire et un fichier](#)
- [D'autres droits](#)

Présentation du chmod

Linux est un système multi-utilisateurs.

Chaque utilisateur appartient au moins à un groupe.

Les fichiers ont donc des permissions visant 3 types d'utilisateurs :

- celle concernant le propriétaire du fichier
- celle concernant le groupe du propriétaire du fichier
- celle concernant les autres utilisateurs.

Pour chaque type de personnes visées, il y a trois types de droits :

- r : droit de lecture
- w : droit d'écriture
- x : droit d'exécution.

A ces types de droits on associe un bit, prenant :

- la valeur 0 si la personne n'a pas ce droit
- la valeur 1 si la personne a ce droit.

On a donc des triplets du genre 111 (par exemple) pour chacun des types d'utilisateurs.

Ce qui nous donne **111 100 101** (encore par exemple).

Dans ce cas,

- le propriétaire a les droits **111**, c'est-à-dire lecture, écriture et exécution
- le groupe a les droits **100**, c'est-à-dire le droit de lecture
- les autres ont les droits **101**, c'est-à-dire les droits de lecture et d'exécution.

Le fait d'avoir un nombre contenant des 0 et des 1 est ce qu'on appelle un nombre binaire.

Nous avons pour habitude de compter en *décimal* avec des chiffres de 0 à 9.

Mais on peut compter aussi en *octal* (chiffres de 0 à 7), ou en *hexadécimal* (de 0 à F, A = 10; B = 11; C = 12 ...; F = 15).

Or, les nombres binaires vont de 000 à 111, ce qui fait en octal de 0 à 7. Et nous pouvons ainsi donner à chaque type d'utilisateur son droit sous la forme d'un chiffre entre 0 et 7 (0 = aucun droit : 7 = tous les droits).

Une petite table pour s'y retrouver :

Position binaire	Valeur octale	Les droits	Commentaire
000	0	- - -	Aucun droits
001	1	- - x	Executable
010	2	- w -	Ecriture
011	3	- w x	Ecrire et executer
100	4	r - -	Lire
101	5	r - x	Lire et executer
110	6	r w -	Lire et ecrire
111	7	r w x	Lire ecrire et executer

Modification des droits sur les fichiers

Maintenant que nous en savons un peu plus sur les droits, nous allons voir comment les appliquer.

On peut appliquer les droits sur un fichier de plusieurs façons :

- de manière symbolique
- de manière octale.

De manière octale

Si je veux changer tous les droits du fichier ou m'assurer qu'ils sont comme je le veux, c'est la meilleure manière.

Prenons un exemple.

J'ai un script *bash* que je veux être le seul à pouvoir modifier, mais que les personnes de mon groupe pourront lire. Et que tous pourront exécuter.

Je devrais mettre les droits suivants :

Utilisateur	U	G	O
Droits d'accès	r w x	r - x	- - x
Position binaire	1 1 1	1 0 1	0 0 1
Octale	7	5	1

On va donc exécuter la commande suivante :

Code : Console

```
chmod 751 script.sh
```

De manière symbolique

Si maintenant, je me rends compte que je ne peux pas modifier un fichier texte, ce que pourtant je voudrais.

Avec la méthode octale, il me faudrait tout décomposer pour seulement supprimer un droit.

Mais il existe la méthode symbolique.

Elle est de type : `chmod [ugoa][+][rwX]`.

C'est l'une de ces lettres **u** (propriétaire du fichier), **g** (groupe), **o** (les autres), **a** (tout le monde = **u + g + o**), suivie de + ou - pour respectivement ajouter ou supprimer les permissions, et la forme symbolique des permissions est de la forme **r** (*read* : lecture), **w** (*write* : écriture), **x** (exécution).

Par exemple, pour pouvoir modifier ce fichier texte qui nous appartient :

Code : Console

```
chmod u+w fich.odt
```

on peut mettre plusieurs droits symboliques en les séparant par des virgules :

Code : Console

```
chmod u+rw,g+r,o+r,a-x fich.odt
```

Dans cet exemple, on ajoute les droits en lecture et en écriture au propriétaire, on ajoute les droits de lecture au groupe, les droits de lecture aux autres, et on enlève les droits d'exécution à tous.



Seul le propriétaire du fichier ou le super-utilisateur **ROOT** peut modifier les droits sur les fichiers et répertoires.

Les droits différents entre un répertoire et un fichier

Les droits de lecture et d'écriture sur un fichier ou un répertoire sont facilement conceptualisables.

Pour un fichier

- **r** : permission de lire le contenu du fichier
- **w** : permission de modifier le contenu du fichier
- **x** : sous Windows, le .exe permet d'exécuter un fichier, mais sous Linux, c'est ce droit qui permet de rendre un fichier exécutable.

Pour un répertoire

- **r** : permission de lister les fichiers
- **w** : permission d'ajouter ou de supprimer des fichiers
- **x** : sur un répertoire, ce droit empêche de rentrer dans le répertoire, et donc de lister les fichiers et répertoires qu'il contient.

D'autres droits

Le *chmod* permet aussi de donner d'autres droits moins connus.

- Le Sticky bits.

Il permet :

- lorsqu'on l'applique à un exécutable, de le garder en mémoire lors de sa première exécution
- lorsqu'on l'applique à un répertoire, seul le propriétaire du fichier ou le propriétaire du répertoire a le droit d'effacer les fichiers.

On le met ainsi :

Code : Console

```
chmod u+t fich
```

- Le SUID, le SGID.

Le SUID permet d'avoir accès aux droits du propriétaire à l'intérieur du programme, pour avoir accès aux fichiers de configuration, par exemple. Pour des raisons de sécurité, le SUID ne s'applique qu'aux programmes binaires compilés à l'exception des scripts Perl.

Le SGID permet de déterminer le groupe des fichiers créés dans le répertoire.

On le met ainsi :

Code : Console

```
chmod u+s prog ou chmod 4755 prog # Pour le SUID  
chmod g+s rep/ ou chmod 2755 rep/ # Pour le SGID
```

Voici donc la fin de cette petite introduction au *Chmod*.

Partager

