chmod

- Objet : chmod
- Niveau requis :
  débutant, avisé
- Commentaires : *Modifier les permissions des fichiers.*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  - Création par MaTTuX_ le *Pffff... On n'était même pas né alors !*
  - Testé par smolski le 26-10-2013
- Commentaires sur le forum : Lien vers le forum concernant ce tuto[1)]

Intro

La commande chmod permet de modifier les permissions aux différents types d'accès (rwx) des fichiers (et répertoire) indépendamment pour le propriétaire, le groupe ou les autres utilisateurs.
Je vais vous expliquer deux manières de les modifier, chacun sa méthode, moi j'ai commencé par la première méthode, puis maintenant je le fais avec les deux sans problème.

Bon pour modifier les permissions, on le fera en console, vous êtes un public averti maintenant.

Syntaxe

```
chmod [option] nom_du_fichier
```

TP 1

Dans cette méthode je vais vous montrer comment modifier par les lettres U G O et R W X pour un rappel aller voir le wiki **A savoir** .
Bon passons aux choses sérieuses, je veux rendre un fichier exécutable, pour cela je vérifie les droits grâce a la commande **ls -l** sur le fichier php dans mon répertoire.

```
ls -l php
```

retour de la commande

```
-rw-r--r-- 1 mattux users 92 avr 20 23:26 php
```

Pour le rendre exécutable avec une petite vérification en même temps je ferai :

```
chmod u+x php
```

```
ls -l php
```

retour de la commande

```
-rwxr--r-- 1 mattux users 92 avr 20 23:26 php
```

Voila je l'ai rendu exécutable juste pour l'user, maintenant vous pouvez permettre au groupe et aux autres d'avoir le droit d'écriture :

```
chmod go+w  php
```

```
ls -l php
```

retour de la commande

```
-rwxrw-rw- 1 mattux users 92 avr 20 23:26 php
```

Voilà je n'ai pas mis U pour l'user car il avait déjà le droit d'écriture.

Dernier test pour cette méthode :
Pour enlever le droit d'exécution et les droits d'écriture pour tous, on fera :

```
chmod ugo-wx php
```

Ou encore avec “a” signifiant “All”, tous :

```
chmod a-wx php
```

Ou, plus simplement :

```
chmod -wx php
```

Ce qui donne :

```
ls -l php
```

retour de la commande

```
-r--r--r-- 1 mattux users 92 avr 20 23:26 php
```

Option t

Le droit **sticky bit**

Ce droit a surtout un rôle important sur les dossiers.

Il réglemente le droit w sur le dossier, en interdisant à un utilisateur quelconque de supprimer un fichier dont il n'est pas le propriétaire.

Ce droit occupe par convention la place du droit x sur la catégorie other de ce dossier, mais bien entendu il ne supprime pas le droit d'accès x (s'il est accordé).
Justement, si ce droit x n'est pas accordé à la catégorie other, à la place de t c'est la lettre T qui apparaîtra.
Sa valeur octale associée vaut 1000.

Pour positionner ce droit :

```
chmod g+t dossier
```

Sur un dossier il signifie que les fichiers créés à l'intérieur ont pour groupe propriétaire le groupe propriétaire du dossier parent.

Si le dossier a le droit x pour tous il donnera :

```
d ... ... ..t  dossier
```

sinon :

```
d ... ... ..T  dossier
```

Si tu fais un u+t au lieu d'un g+t, c'est la même chose mais avec l'utilisateur propriétaire et non plus le groupe :

```
chmod u+t
```

Ça, c'est envoyé par le **captnfab** *depuis le salon df. Il est pas champion le matelot, dites ?* 😀

TP Droits sur serveur

Consultez ce post sur le forum df :

- Problème ProFTPd et répertoire Apache /var/www

Merci à **Nasedo** *et à* **nikau** *pour ces explications.* 😉

TP 2

Dans cette méthode on fera les modifications par des chiffres 4, 2 et 1, n'ayez pas peur c'est très simple, je vais reprendre les mêmes exemple que la Méthode 1, mais pour commencer je dois vous expliquer comment fonctionne cette méthode.

Une autre manière de représenter ces droits est sous **forme binaire** grâce à une **clef numérique**

fondée sur la correspondance entre :

une expression **binaire** = un nombre **octal** = une notation

Soit :

- 000 = 0 = ---
- 001 = 1 = --x (exécution)
- 010 = 2 = -w- (écriture)
- 011 = 3 = -wx
- 100 = 4 = r--(lecture)
- 101 = 5 = r-x
- 110 = 6 = rw-
- 111 = 7 = rwx

Il suffit donc de déclarer un chiffre et un seul entre 0 et 7 correspondant à toute la séquence en notation (r w x) et de l'attribuer à chacune des catégories d'utilisateur user, group, others (u, g, o).

Exemples :

- 777 = rwxrwxrwx = u g o peuvent tous lire + écrire + exécuter.
- 605 = rw----r-x = u peut lire + écrire g rien faire et o lire + exécuter.
- 644 = rw-r--r--= u peut lire + écrire g lire et o lire.
- 666 = rw-rw-rw- = u g o peuvent tous lire + écrire. Aucun ne peut exécuter.

Une astuce permet d'associer rapidement une valeur décimale à la séquence de droits souhaitée. Il suffit d'attribuer les valeurs suivantes pour chaque type de droit.

Le droit de :

- lecture (r) correspond à 4
- écriture (w) correspond à 2
- exécution (x) correspond à 1

On additionne ces valeurs selon qu'on veuille ou non attribuer le droit correspondant pour chacune des catégories (u, g, o).

Ainsi :

- rwx = 7 (4+2+1),
- r-x = 5 (4+1) et
- r-- = 4.

Donne en séquences de droits complètes :

- (rwxr-xr--) = 754.

Pour briller en société, comment convertir un nombre binaire en nombre octal :

Il s'agit donc de passer de la base **2** (où chaque chiffre peut prendre 2 valeurs, *0* ou *1*), à la base **8** (où chaque chiffre peut prendre 8 valeurs, de *0* à *7*).

Mais pour commencer, on va commencer par une autre base, la base **10** (où chaque chiffre peut prendre, je vous le donne en mille, 10 valeurs, de *0* à *9* :-p). Vous savez tous décomposer un nombre

en base **10**. Par exemple **103**.

Le premier chiffre à droite, *3* à le rang **0**, celui juste à sa gauche, *0* le rang **1**, le suivant vers la gauche, *3*, à le rang **2**, et ansi de suite.

103 = 1 x 100 + 0 x 10 + 3 x 1
On peut aussi l'écrire comme ça, si on se rappelle que
n'importe quoi^{1} = *n'importe quoi*
et que
n'importe quoi^{0} = 1 :
$103 = 1 \times 10^{2} + 0 \times 10^{1} + 3 \times 10^{0}$

On remarque que la décompostion consiste à multiplier la valeur du chiffre du rang par la base élevée à la puissance du rang, et d'additionner toutes ces mutiplications

On va faire exactement pareil pour décomposer un nombre en base **2**. Si on dit que *n* est le rang, on multiplie la valeur du chiffre du rang par 2^{n}, et on additionne le résultat de toute ces multiplications.

Par exemple :

$010 = 0 \times 2^{2} + 1 \times 2^{1} + 0 \times 2^{0} = 0 \times 4 + 1 \times 2 + 0 \times 1 = 2$
$101 = 1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0} = 1 \times 4 + 0 \times 2 + 1 \times 1 = 5$

Oui, mais tu as dis qu'on allait apprendre à convertir en octal, pas en décimal

Et bien, puisqu'on converti un nombre binaire composé d'uniquement 3 chiffres, ça ne change rien.

Pourquoi ?

Quel est le plus grand nombre binaire à 3 chiffres possible ? **111**

$111 = 1 \times 2^{2} + 1 \times 2^{1} + 1 \times 2^{0} = 1 \times 4 + 1 \times 2 + 1 \times 1 =$ **7**

Et **7**, en octal ou en décimal, ça vaut toujours **7**

C'est une manière directe (et moins verbeuse, ATTENTION aux erreurs !) d'attribuer les droits et de les écrire sous cette forme en utilisant le code à 3 chiffres résultant.

Exemple pour un fichier :

```
chmod 754 /chemin/du/fichier/test.txt
```

`/chemin/du/fichier/test.txt` étant un fichier imaginaire destiné à illustrer l'exemple...

Exemple pour un répertoire :

```
chmod 754 /chemin/du/repertoire/test
```

où nous pouvons ajouter -R pour rendre les autorisations récursives à l'intérieur de ce répertoire, soit :

```
chmod -R 754 /chemin/du/repertoire/test
```

Il est d'usage précautionneux de SYSTEMATIQUEMENT vérifier par

```
ls -al /chemin/du/repertoire/
```

que nous avons bien obtenu ce que nous voulions modifier... C'est beaucoup, beaucoup de temps d'gagné ! 😀

Droits spéciaux - SUID

SUID - un droit sécurisé - De bonnes bases sont requises... pas cool !

Ce droit demande une connaissance préalable de **tous les droits** de bases, leur **fonctionnement**, leur **manipulation**, ainsi que de la commande de listage **ls**.
Si vous ne comprenez pas cette litanie, revoyez les commandes CHMOD CHOWN LS et concert...

POURQUOI CE DROIT ?

Observez :

Exemple de fichier où SUID s'applique nécessairement :

Saisir dans un terminal :

```
ls -l /etc/shadow
```

retour de la commande

```
-rw-r-----  root root  shadow
```

Les droits de réalisation (rw) dans le fichier **shadow** sont limités à u=root exclusivement.

et :

```
ls -l /usr/bin/passwd
```

retour de la commande

```
-rxsr-xr-x  root  root /usr/bin/passwd
```

SUID, en s'intégrant (rx**s**r) dans le fichier intermédiaire **passwd** où :

1. root = (rw-)[2] sur le fichier shadow
2. et (o)[3] = (r-x) autorisant les exécutions de commande d'utilisateurs non-root.

permet à **passwd** de créer TEMPORAIREMENT la passerelle NÉCESSAIRE à la *commande initiale* issue de l'user lambda et d'utiliser ainsi le droit root (rw-) de **passwd** sur **shadow** pour la réaliser !

SUID - Conté...

Il était une fois...

SUID permet la réalisation d'une commande d'exécution (x) à partir d'un utilisateur-lambda, *via* un fichier d'utilisateur exclusif (u = root par exemple...) *vers* un fichier de réalisation finale de propriété exclusive identique sans attribuer des droits permanents hors de la commande présentée !

En effet, le droit SUID est un droit *TEMPORAIRE* de fichier.
Il s'agit en fait d'un **dispositif de sécurité** essentiel qui autorise un utilisateur à bénéficier de droits plus étendus que les siens et obtenir les droits sur un fichier exclusif (réservé à root en général...) par une commande exécutive (–x).

Pour préserver la sécurité permanente du fichier de réalisation, il fonctionne juste le temps et sous le contrôle de la commande sollicitée.
Ce droit est noté symboliquement par s (sa valeur octale est 4000).

Dans la notation, il se met en lieu et place du **x**, celui-ci prenant dans le listage des droits celle du **w** au centre de la notation habituelle.

En fait, la notation de ce droit pour user se présente :

- rxs - - = (*temporairement*) rwx - -)

ou en binaire :

- 47 - - = 7 - -.

Exemple

Saisir dans un terminal :

```
ls -l /etc/shadow
```

retour de la commande

```
-rw-r-----  root root  shadow
```

Ce qui indique toutes les limites de ce fichier réservé à root. Sécurité maxi !

Puis saisir :

```
ls -l /usr/bin/passwd
```

retour de la commande

```
-rxsr-xr-x  root  root /usr/bin/passwd
```

Ici, le droit sur le fichier passwd est accordé à tous les exécutants lambda (valeur x pour tous).

Le fichier **passwd** propriété de *root*, a bien sûr accès de réalisation (rw) dans le fichier **shadow**, qui est de propriété *root* également.

Le positionnement du SUID permet à passwd d'utiliser ses droits de réalisation root (rw-) dans le fichier shadow pour l'exécution de la commande lambda initié, en la prenant à son compte !

Exit l'utilisateur non-root initiateur de la commande ! C'est donc root qui la prend à son compte juste pour l'exécution à l'intérieur de cette commande.

Final

Il est ainsi possible de permettre à tout exécutant non-root d'agir dans un fichier en préservant la haute sécurité de celui-ci dans le cadre **UNIQUE** de l'exécution d'une commande qui serait non autorisée au départ sans la rendre permanente pour toute autre commande.

suid permet en fait de ne pas accorder de droits sur un fichier sensible de manière dangereuse ! \\

Sécurité préservée, santé assurée..!

Astuce - find

Créé sous l'inspiration de Melodie, avec la tutelle attentive de Mattux_ ! Qu'ils en soient remerciés ici !

Astuce délicate ! Règles de Sécurité en jeu !

L'utilisation de **find** permet d'établir une liste de répertoires ou de fichiers afin de leur appliquer en série et de manière typée une commande chmod, chown ou autres...

Exemple :

```
find -type d /home/monrepertoire -exec chmod -R 775 {} ;
```

se décompose ainsi :

- “find -type d” où -type d indique de ne lister et d'agir que sur des répertoires.
- (Pour des fichiers, et que des fichiers, écrire -type f)
- “/home/repertoire” le chemin où débute la commande
- “-exec chmod -R 775” exécution de la commande (ici droits en octales...)
- “{} ;” pour finaliser find. Voir commentaires **man find** ligne 804 -exec commande

L'exécution judicieuse de cette ligne permet :

sur un ensemble de répertoires ou de fichiers typés,

C'est à dire définis au préalable dans la ligne de commande,

ATTENTION AUX RISQUES DE SECURITE !

de placer ou replacer une série de droits que l'on désire harmoniser d'un seul coup !
Regardez-y à deux fois dans cett' manip, sous peine de fragiliser votre machine !

D'une manière générale :
Tout ce qui est listé par l'emploi de find demande une forte connaissance de ce que l'on fait **ET** une circonspection absolue !

Sinon, find , par ses possibilités de typage pointu[4] permettra un listage au p'tits oignons sans écritures laborieuses, ce qui est le soucis de tout linuxien averti au final !

Ne sautez pas les étapes et prenez conseils, sur DF si ça vous dit...

À la place de chmod ou de -type, plusieurs options ouvrent un grand jeu de quilles, attention donc à certains strikes définitifs !!! ././/._._

Pour aller plus avant :

- Consultez le :

```
man find
```

Encore lui ? Oui encore..!

- Le tuto find : Le tuto df sur la commande find

— **smolski** 2009/01/11 07:46 *ça nous rajeuni pas ça !*

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
lecture + écriture/modif
3)
propriétaire
4)
voir le man find.... **1687** ligne !!! C'est pas pour rien !