BTS SIO SI 1	La gestion des ACL sous LINUX *Point du programme :* *Architecture et fonctions d'un système d'exploitation*	Fiche de synthèse N° 12

I- Droits d'accès sous Linux : gérer les accès aux fichiers

Les **droits d'accès** définissent la possession d'un fichier ou d'un répertoire à un utilisateur et à un groupe d'utilisateurs. Ils gèrent aussi quelles actions les utilisateurs ont le droit d'effectuer sur les fichiers, selon qu'ils sont propriétaire du fichier, membre du groupe propriétaire du fichier ou ni l'un ni l'autre. La possession et la gestion des permissions associées s'effectuent individuellement avec chaque fichier.

A- Les propriétaires

Par la propriété d'un fichier, on désigne à quel utilisateur appartient le fichier, qui le possède. À partir de cette possession (ou non), il sera ensuite possible de définir des permissions d'accès sur le fichier.

La possession d'un fichier se définit sur trois catégories :

1- l'**utilisateur** propriétaire du fichier **(*u*)**.La plupart du temps, il s'agit du créateur du fichier.

2- le **groupe** propriétaire du fichier ***(g)***. Si un utilisateur est membre d'un certain groupe qui possède la propriété d'un fichier, l'utilisateur aura aussi certaines permissions particulières sur ce fichier.

3- les autres, **other**, le reste du monde**(*o*)**. Ni propriétaire du fichier, ni membre du groupe propriétaire du fichier.

Faisons une analogie avec les voitures. Le **propriétaire** *serait la personne au nom de laquelle la voiture est immatriculée. Le* **groupe propriétaire** *est l'ensemble des personnes qui sont inscrites en tant que conducteurs secondaires de la voiture chez l'assureur. Enfin, les* **autres** *correspond à toutes les autres personnes n'étant ni détenteur de l'immatriculation ni inscrites en tant que conducteurs de la voiture chez l'assureur.*

La gestion des inclusions ou exclusions des utilisateurs à un groupe d'utilisateurs sera abordée dans le chapitre précédent.

Les **permissions** désignent ce que les diverses catégories d'utilisateurs (propriétaire d'un fichier, membres du groupe propriétaire d'un fichier et le reste du monde) ont l'autorisation d'effectuer sur un fichier donné.

Par exemple, une catégorie d'utilisateurs peut avoir accès en lecture et écriture à un fichier, alors qu'une autre catégorie a accès en lecture seulement à ce même fichier.

Les permissions se définissent sur **trois niveaux** :

1. la **lecture** d'un fichier : cette permission est nécessaire pour pouvoir accéder au contenu d'un fichier (écouter une piste audio, visionner un film, lire un texte, naviguer à l'intérieur d'un répertoire…). Cette permission est notée **r** (pour ***read***, lire).
2. l'**écriture** dans un fichier : cette permission est nécessaire pour pouvoir apporter des modifications à un fichier (corriger un texte et enregistrer les changements ; ajouter, modifier ou supprimer un fichier dans un dossier ; etc.). Cette permission est notée **w** (pour ***write***, écrire).

3. l'**<u>exécution</u>** d'un fichier : cette permission est **nécessaire particulièrement pour les logiciels**, afin qu'ils puissent être exécutés. Cette permission est notée **x** (pour ***eXecute***, exécuter).

Par exemple, l'utilisateur `toto` *dispose des droits de lecture et d'exécution sur le répertoire* `foo`*, mais pas la permission d'écriture sur ce répertoire ;* `toto` *peut donc exécuter les programmes présents dans ce répertoire et ouvrir les fichiers qu'il contient, mais ne peut pas les modifier ni en créer de nouveaux.*

Pour chacune des trois catégories d'utilisateurs (propriétaire, membres du groupe propriétaire et reste du monde) sont définies ces trois permissions :

- le propriétaire dispose ou non de la permission de lecture, d'écriture et d'exécution sur un fichier ;
- le membre du groupe propriétaire dispose ou non de la permission de lecture, d'écriture et d'exécution sur un fichier ;
- tous les autres utilisateurs disposent ou non de la permission de lecture, d'écriture et d'exécution sur un fichier.

Les droits sont affiché pars une série de 9 caractères, associé 3 par 3 **(rwx rwx rwx)** définissent les droits des 3 identités **(u, g et o).**

Les répertoires : ***un cas particulier***

Pour **accéder au contenu d'un répertoire** (pour ouvrir un fichier ou se déplacer dans un sous-répertoire), un utilisateur doit disposer de la permission d'***exécution*** **(*x*)** sur ce répertoire.
Pour être en mesure de **lister le contenu d'un répertoire**, l'utilisateur doit avoir la permission de ***lecture*** **(*r*)** sur ce répertoire.
Pour **écrire dans le répertoire**, la permission d'***écriture*** **(*w*)** doit être accordée.

L'utilisateur peut disposer de ces permissions selon qu'il est propriétaire du répertoire, membre du groupe propriétaire du répertoire ou faire partie du reste du monde.

- Un utilisateur ne disposant ni des permissions de lecture ni d'exécution ne pourra aucunement accéder au contenu du répertoire.
- Un utilisateur ne disposant que de la permission de lecture pourra **lister** *le contenu du dossier. (Par exemple, avec la commande* ***ls*** *dans une fenêtre de terminal.) Il ne pourra pas accéder au dossier avec son navigateur de fichiers.*
- Un utilisateur ne disposant que de la permission d'exécution peut ouvrir un répertoire, mais ne peut pas en voir le contenu. ***C'est utile, par exemple, pour traverser un répertoire dont on ne doit pas pouvoir lister le contenu.***
- Un utilisateur disposant des droits de lecture et d'exécution pourra lister le contenu d'un dossier et y entrer avec son navigateur de fichiers.

B- Droits attribués automatiquement à un fichier

Lorsqu'un nouveau fichier est créé, celui-ci obtient automatiquement certains paramètres :

1. **Propriétaires :** Par défaut, le propriétaire d'un nouveau fichier est son créateur et le groupe propriétaire, le groupe principal de son créateur. Par exemple, si l'utilisateur `toto`, dont le groupe principal est `utilisateurs`, crée un nouveau fichier ou dossier, celui-ci appartient à `toto:utilisateurs` ;
2. **Permissions :** Les permissions accordées par défaut sont celles déterminées par un paramètre particulier appelé le ***masque utilisateur***.

   Dans Ubuntu, le masque utilisateur par défaut accorde les permissions **`rwx-rx-rx`** :

   - le **propriétaire du fichier** dispose des permissions de **lecture, d'écriture et d'exécution** ;
   - le **groupe propriétaire** dispose des droits de **lecture et d'exécution, mais pas d'écriture** ;
   - le **reste du monde** dispose des droits de **lecture et d'exécution, mais pas d'écriture.**

C- Commandes chown *ou* chmod

`chown` et `chmod` ont des buts différents : le premier **modifie les propriétaires d'un fichier**, alors que le second **modifie les permissions sur ce fichier**.

Imaginons un fichier `foo.txt` appartenant à l'utilisateur 'root' et au groupe 'root'. Ce fichier a le mode 420 (soit lecture seule pour le propriétaire, écriture seule pour le groupe propriétaire et aucun accès pour le reste du monde).

- Avec uniquement `chown`, le propriétaire d'un fichier est changé, mais les permissions sur ce fichier sont maintenues. Exécutons la commande suivante :

```
sudo chown toto foo.txt
```

L'utilisateur 'toto' devient désormais le propriétaire du fichier. 'toto' a maintenant accès en lecture seule à ce fichier (puisque les permissions pour le propriétaire du fichier sont limitées à la lecture seule pour le propriétaire du fichier). Les permissions ne sont pas modifiées.

- Avec uniquement `chmod`, les permissions d'un fichier sont modifiées, mais les propriétaires et groupe propriétaire sont maintenus. Exécutons la commande suivante :

```
sudo chmod ug+rw foo.txt
sudo chmod o+r foo.txt
```

Le propriétaire du fichier et le groupe propriétaire du fichier disposent désormais de l'accès en lecture et écriture sur le fichier `foo.txt`, et le reste du monde y a accès en lecture seule.

L'utilisateur 'toto' n'a donc accès qu'en lecture au fichier, puisqu'il n'est ni le propriétaire ('root') ni membre du groupe propriétaire (groupe 'root') ; il n'a donc que les accès du reste du monde.

Généralement :

1. Si un fichier ou dossier contient des informations relativement sensibles ou privées, utilisez conjointement `chown` et `chmod` pour régler des propriétaires adéquats et des permissions qui ne permettent pas l'accès total au fichier.
2. Si un fichier ou dossier ne contient que de l'information généraliste ou publique, autorisez simplement l'accès en lecture, écriture et exécution au fichier avec `chmod` et ne vous préoccupez pas des propriétaires.

Nous allons approfondir notre analyse sur les ACL.

II- Le « umask »

Pourquoi à la création d'un fichier ou d'un répertoire j'ai automatiquement les droits suivants ? :

- Quand on crée un fichier : `rw- r-- r--`
- Quand on crée un répertoire : `rwx r-x r-x`

Tout cela dépend de la valeur du mask des droits, c'est-à-dire la valeur de son **umask**.

On retrouve la valeur de son umask, simplement en tapant la commande umask dans un terminal.

Pour la modifier, il suffit d'exécuter cette commande :
`umask 002` (par exemple)

La valeur par défaut est : **umask 0022**.

Quand on crée un fichier, les droits de celui-ci sont 666 (soit `rw-rw-rw`) + masque.
Quand on crée un répertoire c'est le même principe mais les droits de celui-ci sont de 777 + masque.

Alors comment retrouvez les droits associés à un fichier ou à un répertoire par défaut ?

Avec cet exemple de mask positionné à 022, les fichiers créés auront par défaut les droits `rw-r--r--`, cela s'explique (un peu d'algèbre booléen) :
Quand on crée un fichier, les droits de celui-ci sont 666 (soit `rw-rw-rw`) + masque
Si on note ça en bit, cela fait :
110 110 110 pour les rw-rw-rw-
000 010 010 pour le masque de 022
---
110 100 100 pour 666 et masque
`rw- r-- r--`

Ainsi, dans certains cas, il est plus intéressant de changer la valeur de umask que de faire des chmod à répétition ensuite.

Attention une mauvaise gestion des droits sous Linux peut s'avérer **très dangereux** !

Un exemple simple, vous montez un site web (avec un serveur LAMP) et vous rencontrez une erreur : "*You don't have permission to access /www/dossier/ on this server*".
Vous n'avez donc pas les droits, la solution de facilité serait de faire un chmod -R 777 (tous les droits à tout le monde sur tous les fichiers).

Cette opération est risquée car elle autorise l'exécution de tous les fichiers. Une personne avec des mauvaises intentions pourrait donc y insérer du code malveillant et l'exécuter sur votre machine.

Il faut donc faire très attention aux droits que vous accordez à vos fichiers.

III- Les ACL (Access Control Lists)

Les ACLs offrent la possibilité de positionner des droits d'accès supplémentaires.
Le propriétaire d'un fichier peut grâce aux ACLs accorder des privilèges à un ou plusieurs utilisateurs et/ou groupes **qui se substitueront aux droits d'accès de base**.

Avec les ACLs c'est possible de donner des droits à un utilisateur qui ne fait pas partie du groupe en ne modifiant pas les droits pour les autres.
De même on peut autoriser des droits d'accès pour un groupe d'utilisateurs qui n'est pas le groupe du fichier.
Il n'y a pas des limites concernant le nombre d'utilisateurs ou groupes à ajouter avec les ACLs.

Un ACL est composé de plusieurs entrées de type ACL. Une entrée spécifie les permissions d'accès pour un objet associé pour un utilisateur ou groupe d'utilisateurs en utilisant une combinaison des droits traditionnels r, w et x.

A- Vérifier la configuration du noyau

Tout d'abord, sachez que les ACL ne peuvent être utilisées que si le noyau le supporte; pour savoir si c'est votre cas, loguez-vous en tant que root. Tapez ensuite :

```
grep ACL /boot/config-*

CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_EXT4DEV_FS_POSIX_ACL=y
…
CONFIG_FS_POSIX_ACL=y
```

La ligne suivante indique que le support général des ACL est présent :

```
CONFIG_FS_POSIX_ACL=y
```

Ensuite des lignes du type suivant permettent de savoir pour quels systèmes de fichiers les ACL sont disponibles :

```
CONFIG_EXT3_FS_POSIX_ACL=y
```

On remarque que les ACL fonctionnent sur les volumes formatés en EXT3 par exemple.

B- Installation du paquet acl

Les ACL sont activées, mais nous ne pouvons toujours pas les modifier, pour cela nous devons installer le paquet acl :

```
# apt-get install acl
```

C- Modifier les ACL : setfacl

1- Ajouter une ACL

Pour ajouter une ACL, vous devez utiliser la commande setfacl avec l'option -m :

```
setfacl -m permissions fichierOuDossier
```

Les permissions s'écrivent sous cette forme :

```
préfixe:[utilisateurOuGroupe:]droits
```

Rappel :

- *Les préfixes disponibles sont :*
  - ***u:** : Pour modifier les droits d'un **u**tilisateur*
  - ***g:** : Pour modifier les droits d'un **g**roupe*
  - ***o:** : Pour modifier les droits du reste du monde (**o**ther)*
- *Les droits s'écrivent sous la forme d'un triplet rwx que vous devez déjà connaitre :*
  - ***r** = droit de lecture*
  - ***w** = droit d'écriture*
  - ***x** = droit d'exécution pour les fichiers, pour les dossiers, c'est le droit "d'entrée" dans le dossier*

Exemple :

```
setfacl -m u:kamel:rw- test
```

... donnera les droits de lecture et d'écriture à kamel pour **le fichier** test.

Ajouter **l'option -R** permet d'appliquer des droits à tout un répertoire :

```
setfacl -Rm u:kamel:rw RepertoireDeTest/
```

... effectuera la même opération que tout à l'heure mais sur **tout le dossier** RepertoireDeTest

L'option -R doit être spécifiée avant l'option **-m**

Vous pouvez bien sur spécifier des permissions pour plusieurs utilisateurs/groupes à la fois, pour cela, séparez-les par une virgule :

```
setfacl -m u:kamel:rw,u:quentin:rwx,g:sio1:r,o:--- test
```

setfacl permet aussi de modifier les droits classiques (comme chmod) Il faut spécifier un nom vide :

```
setfacl -m u::rwx,g::r--,o:--- test
```

... donnera les droits rwxr----- au fichier test

2- Doits par défaut et héritage

Si vous appliquez une ACL à un dossier, les fichiers créés ensuite dans se dossier n'hériteront pas de son ACL. Heureusement, **l'héritage des ACL** est possible, il suffit de rajouter le préfixe **d:** (comme **d**efault) au début de l'ACL :

```
setfacl -m d:u:kamel:rw RepertoireDeTest/
```

```
setfacl -m d:u:kamel:rw,o:--- RepertoireDeTest/
```

Dans cette ACL, seul **u:kamel:rw** sera un droit par défaut ; si vous souhaitez que les fichiers héritent aussi de **o:---**, vous devez taper :

```
setfacl -m d:u:kamel:rw,d:o:--- RepertoireDeTest/
```

Il cependant possible de se passer du préfixe **d:**, grâce à **l'option -d**, dans ce cas, toutes les permissions spécifiées seront des permissions par défaut :

```
setfacl -dm u:kamel:rw,o:--- RepertoireDeTest/
```

... aura le même effet que le code précédent.
Une fois encore, l'option -d doit être spécifiée avant l'option -m

Ajouter des droits par défaut ne modifie pas les droits existants, si vous souhaitez ajouter une ACL à tout un répertoire et ses sous-répertoires **ET que cette ACL soit héritée par la suite**, vous devez le faire de cette manière : (notez la présence de l'option -R)

```
setfacl -Rm d:u:kamel:rwx,d:g:sio1:r--,d:o:---,u:kamel:rwx,g:sio1:r--,o:---
 RepertoireDeTest/
```

3- Supprimer une ACL

Pour supprimer une ACL, il suffit d'utiliser l'option **-b** ...

```
setfacl -b test
```

... supprimera toute l'ACL du fichier test

Vous pouvez supprimer une partie de l'ACL avec l'option **-x** :

```
setfacl -x u:quentin,g:sio1 test
```

... supprimera les permissions de l'utilisateur quentin et du groupe sio1 du fichier test

Pour supprimer UNIQUEMENT les autorisations par défaut, vous devez utiliser l'option **-k**, TOUTES les permissions par défaut seront supprimées

D- Voir les ACL en place : getfacl

La commande getfacl vous permet de connaitre les ACL en place :

getfacl reperoireDeTest/

```
# file: reperoireDeTest/
# owner: op414
# group: op414
user::rwx
user:kamel:rwx
user:quentin:r--
group::rwx
mask::rwx
other::---
default:user::rwx
default:user:kamel:rwx
default:user:quentin:r--
default:group::rwx
default:mask::rwx
default:other::---
```

Le masque

Le masque vous permet de savoir quelles sont les autorisations maximales accordées à un fichier ou dossier (utilisateurs et groupes confondus), les droits classiques (chmod) ne sont pas comptabilisés.

getfacl test

```
# file: test
# owner: op414
# group: op414
user::rwx
user:kamel:rwx
user:quentin:r--
group::rwx
mask::rwx
other::--
```

Ici, le masque est **rwx** car kamel possède les droits rwx.
L'utilité du masque est de pouvoir enlever des permissions à tous les utilisateurs et groupes (sauf de l'utilisateur propriétaire, dont les droits sont définis par chmod):

setfacl -m m:r-- test

Vous remarquez qu'il faut utiliser le préfixe **m:** (comme **m**ask). Refaisons un coup de **getfacl** sur le fichier :

getfacl test

```
# file: test
# owner: op414
# group: op414
user::rwx
user:kamel:rwx              #effective:r--
user:quentin:r--            #effective:r--
group::rwx                  #effective:r--
mask::r--
other::---
```

On remarque que les droits de kamel, quentin et du groupe propriétaire n'ont pas été modifiés (ce qui permet de les rétablir en ré-augentant le masque). En revanche, il est maintenant écrit **#effective:r--** en face de leurs lignes. Cela signifie que leurs **droits réellements appliqués** sont r-- !

Annexes (suppléments)

Copie des ACL (cp et mv)

Les commandes cp et mv sont capables de conserver les ACL. Il suffit de spécifier l'option -a lors de l'utilisation de cp . mv le fait au-to-ma-ti-que-ment. Bien entendu, le répertoire cible doit être situé sur une partition gérant les ACL.

Un dernier point : quand un fichier possède une ACL et que vous faites un ls -l, l'ACL ne peut être écrite en entier ; le signe + s'affiche pour signifier le présence de l'ACL : -rw-rw----+

Monter les partitions avec l'option acl

Si vous lisez cette sous-partie, c'est que les commandes setfacl et getfacl n'ont pas fonctionné chez vous. Ces manipulations doivent effectuées en tant qu'utilisateur root

Pour que vous puissiez utiliser les ACL, les partitions doivent êtres montées avec l'option correspondante...

```
# mount -t ext3 -o defaults,acl /dev/hda1/ /home
```

... pour monter la partition 2 du premier disque, formatée en ext3 dans le répertoire /home

Vous pouvez remonter un partition déjà montée :

```
# mount -o remount,acl /home
```

Si vous voulez que le volume soit monté automatiquement avec l'option acl, vous devez modifier le fichier /etc/fstab

```
# nano /etc/fstab
```

Vous devez rajouter ,acl dans la colonne "options" de la partition concernée :

```
/dev/sda1       /               ext3    errors=remount-ro 0       1
```

devient chez moi :

```
/dev/sda1       /               ext3    errors=remount-ro,acl 0       1
```

Voilà, ma partition principale sera automatiquement montée avec l'option acl au démarrage Si vous avez une partition /home séparée, vous devez bien entendu modifier sa ligne.

Il ne vous reste plus qu'à redémarrer ou remonter les partitions concernées comme vu précédemment

Exemple d'utilisation concret

Voici un exemple d'utilisation des ACL, il est issu de ma propre utilisation de vos nouvelles meilleures amies

Si vous créez un serveur web grâce à ce tuto, vous risquez d'avoir un petit problème de permissions. En effet, votre machine ne possède que deux **vrais** comptes utilisateur (le votre et le compte root), si vous vous loggez en FTP, le serveur vsFTPd utilise l'uilisateur www-data. Le compte **virtuel** d'administration à accès à tout le répertoire /home mais ne pourra pas entrer dans votre répertoire personnel car celui-ci appartient à votre compte utilisateur et non pas à www-data

Pour palier ce problème, il suffit d'utiliser cette ACL :

```
setfacl -Rm d:u:www-data:rwx,d:u:op414:rwx,u:www-data:rwx,u:op414:rwx /home/op414
```

... dans le cas ou votre compte utilisateur est op414

Le problème est résolu

Annexe : Représentation des droits

Cet ensemble de 3 droits sur 3 entités se représente généralement de la façon suivante : on écrit côte à côte les droits r, w puis x respectivement pour le propriétaire (u), le groupe (g) et les autres utilisateurs (o). Les codes u, g et o (u comme *user*, g comme *group* et o comme *others*) sont utilisés par les commandes UNIX qui permettent d'attribuer les droits et l'appartenance des fichiers. Lorsqu'un flag est attribué à une entité, on écrit ce flag (r, w ou x), et lorsqu'il n'est pas attribué, on écrit un '-'. Par exemple,

```
rwxr-xr--
\ /\ /\ /
 v  v  v
 |  |  droits des autres utilisateurs (o)
 |  |
 |  droits des utilisateurs appartenant au groupe (g)
 |
droits du propriétaire (u)
```

Une autre manière de représenter ces droits est sous forme binaire grâce à une clef numérique fondée sur la correspondance entre **un nombre décimal et son expression binaire** :

- 0 = 000
- 1 = 001
- 2 = 010
- 3 = 011
- 4 = 100
- 5 = 101
- 6 = 110
- 7 = 111

À l'expression binaire en trois caractères sont associés les 3 types de droits (r w x) ; il suffit donc de déclarer pour chacune des catégories d'utilisateur (user, group, others) un chiffre entre 0 et 7 auquel correspond une séquence de droits d'accès. Par exemple :

- 777 donne 111 111 111 soit r w x r w x r w x
- 605 donne 110 000 101 soit r w - - - - r - x
- 644 donne 110 100 100 soit r w - r - - r - -
- 666 donne 110 110 110 soit r w - r w - r w -

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 :

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

Puis on additionne ces valeurs selon qu'on veuille ou non attribuer le droit en correspondant.

Ainsi, rwx « vaut » 7 (4+2+1), r-x « vaut » 5 (4+1) et r-- « vaut » 4. Les droits complets (rwxr-xr--) sont donc équivalents à 754.

Une manière directe d'attribuer les droits est de les écrire sous cette forme et d'utiliser le code à 3 chiffres résultant avec chmod (voir ci-après).

Exercice sur les droits :

Voici une réponse à la commande : ls –l

```
mateo21@mateo21-desktop:~$ ls -l
total 40
drwxr-xr-x 2 mateo21 mateo21 4096 2007-11-13 21:53 Desktop
drwxr-xr-x 2 mateo21 mateo21 4096 2007-11-13 13:46 Documents
lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -> /usr/share/example-content
drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-25 20:28 images
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Images
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-25 11:11 log
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Modèles
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Musique
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Public
-rw-r--r-- 1 mateo21 mateo21    0 2007-11-15 23:14 rapport.txt
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-19 19:51 tutos
drwxr-xr-x 2 mateo21 mateo21 4096 2007-10-19 01:21 Vidéos
```

Questions :

1- Indiquez le nom de l'utilisateur ainsi que son groupe d'appartenance.
2- Quels sont les types de fichiers pour les trois lignes suivantes :

```
lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -> /usr/share/example-content
drwxr-xr-x 3 mateo21 mateo21 4096 2007-09-25 11:11 log
-rw-r--r-- 1 mateo21 mateo21    0 2007-11-15 23:14 rapport.txt
```

3- Exprimez à travers des phrases les droits associés aux trois fichiers précédents.
4- Indiquez la ligne de commande permettant de changer le propriétaire du fichier *rapport.txt* par le nom de propriétaire *max.*
5- Changer les droits (ou permissions) sur ce fichier pour le propriétaire *max* en utilisant deux écritures différentes. Celui-ci doit pouvoir seulement exécuter le fichier *rapport.txt*
6- Changer les droits du fichier *log* pour le groupe d'utilisateur. Celui-ci doit pouvoir exécuter et lire le fichier *log*

B- Voir les permissions en ligne de commande :

Les droits des fichiers d'un dossier peuvent être affichés par la commande

```
ls -l
```

Les droits d'accès apparaissent alors comme une liste de 10 symboles. :

```
drwxr-xr-x
```

Le premier symbole est soit « - », « d », soit « l », nous indiquant la nature du fichier :

- **-** : fichier
- **d** : dossier
- **l** : lien

Exemple :

Reprenons l'exemple théorique précédent :

```
drwxr-xr-x
```

Il se traduit de la manière suivante :

- **d** : c'est un dossier.
- **rwx** pour le 1er groupe de 3 symboles : son propriétaire peut lire, écrire et exécuter.
- **r-x** pour le 2nd groupe de 3 symboles : le groupe peut uniquement lire et exécuter le fichier, sans pouvoir le modifier.
- **r-x** pour le 3ème groupe de 3 symboles : le reste du monde peut uniquement lire et exécuter le fichier, sans pouvoir le modifier.

En pratique, en exécutant la commande suivante,

```
ls -l
```

on obtient la liste du contenu du répertoire courant, par exemple :

```
drwxr-xr-x   6 cyrille cyrille     4096 2008-10-29 23:09 Bureau
drwxr-x---   2 cyrille cyrille     4096 2008-10-22 22:46 Documents
lrwxrwxrwx   1 cyrille cyrille       26 2008-09-22 22:30 Examples -> /usr/share/example-
content
-rw-r--r--   1 cyrille cyrille  1544881 2008-10-18 15:37 forum.xcf
drwxr-xr-x   7 cyrille cyrille     4096 2008-09-23 18:16 Images
drwxr-xr-x   2 cyrille cyrille     4096 2008-09-22 22:45 Modèles
drwxr-xr-x 267 cyrille cyrille    20480 2008-10-27 22:17 Musique
drwxr-xr-x   2 cyrille cyrille     4096 2008-09-22 22:45 Public
drwxr-xr-x   2 cyrille cyrille     4096 2008-10-26 13:14 Vidéos
```

On retrouve dans la première colonne le groupe de 10 caractères permettant de connaître les droits pour chaque fichier.

Ainsi, pour le fichier `forum.xcf`, on a :

```
-rw-r--r--
```

- Le 1er caractère est - ⇒ c'est un fichier.
- Le premier groupe de 3 caractères est **rw-** ⇒ le propriétaire a le droit de lecture et écriture (mais pas d'exécution) sur le fichier.
- Les 2 groupes suivants sont **r--** ⇒ Les utilisateurs du groupe et les autres n'ont que le droit de lecture (pas d'écriture, ni d'exécution)

C- Modifier les permissions en ligne de commande

Un fichier a un **propriétaire** et un **groupe**. Nous pouvons les changer.

chown

La commande `chown` (***ch**ange **own**er*, changer le propriétaire) permet de changer le propriétaire du fichier. Seuls le super-utilisateur ou le propriétaire actuel d'un fichier peut utiliser `chown`. La commande s'utilise de la façon suivante :

```
sudo chown toto fichier1
```

Le fichier `fichier1` appartient maintenant à l'utilisateur `toto`.

chown permet aussi de changer en une seule commande le propriétaire et le groupe du fichier :

```
sudo chown toto:lesPotes fichier1
```

Le fichier `fichier1` appartient alors à l'utilisateur `toto` et au groupe `lesPotes`.

chgrp

La commande **chgrp** (pour **ch**ange **gr**ou**p**) permet de changer le groupe auquel appartient le fichier. Tous les membres de ce groupe seront concernés par les permissions du groupe de la 2ème série de **rwx**. Encore une fois, seuls le super-utilisateur ou *le propriétaire actuel* d'un fichier peut utiliser `chgrp` (un membre du groupe ne peut pas changer le groupe propriétaire). La commande s'utilise de la façon suivante :

```
sudo chgrp mesPotes fichier2
```

Le fichier `fichier2` appartient maintenant au groupe `mesPotes`. Tous les membres du groupe `mesPotes` auront accès à ce fichier selon les permissions du groupe.*Quand l'utilisateur actuel n'est pas le propriétaire actuel du fichier, il sera nécessaire de faire précéder la commande par `sudo`, puisqu'elle devra être effectuée avec les droits d'administration.*

chown, pour changer simultanément le propriétaire et le groupe propriétaire

Pour changer à la fois le propriétaire *et* le groupe propriétaire, une syntaxe particulière de la commande `chown` peut être utilisée. Encore une fois, seuls le super-utilisateur ou *le propriétaire actuel* d'un fichier peut utiliser `chown` (un membre du groupe ne peut pas effectuer de changement de propriété). La commande s'utilise de la façon suivante :

```
chown nouveau_propriétaire:nouveau_groupe_propriétaire nom_du.fichier
```

Quand l'utilisateur actuel n'est pas le propriétaire actuel du fichier, il sera nécessaire de faire précéder la commande par `sudo`, puisqu'elle devra être effectuée avec les droits d'administration.

Imaginons le même fichier `foo.txt` possédé par *utilisateur1* et appartenant au groupe propriétaire *groupe1*. Le propriétaire doit devenir *utilisateur2* et la propriété de groupe de ce fichier doit passer au groupe *groupe2*. En étant connecté au compte *utilisateur1*, l'exécution de cette commande effectuera l'opération demandée :

```
chown utilisateur2:groupe2 foo.txt
```

chmod

L'outil `chmod` (***ch**ange **mod**e*, changer les permissions) permet de modifier les permissions sur un fichier. Il peut s'employer de deux façons : soit en précisant les permissions de manière octale, à l'aide de chiffres[2)] ; soit en ajoutant ou en retirant des permissions à une ou plusieurs catégories d'utilisateurs à l'aide des symboles `r w et x`, que nous avons présenté plus haut. Nous préférerons présenter cette seconde façon ("ajout ou retrait de permissions à l'aide des symboles"), car elle est probablement plus intuitive pour les néophytes. Sachez seulement que les deux méthodes sont équivalentes, c'est-à-dire qu'elles affectent toutes deux les permissions de la même manière.

En gérant chaque droit séparément

De cette façon, on va choisir :

1. À qui s'applique le changement
    - **u** (**u**ser, utilisateur) représente la catégorie "propriétaire" ;
    - **g** (**g**roup, groupe) représente la catégorie "groupe propriétaire" ;
    - **o** (**o**thers, autres) représente la catégorie "reste du monde" ;
    - **a** (**a**ll, tous) représente l'ensemble des trois catégories.
2. La modification que l'on veut faire
    - **+** : ajouter
    - **-** : supprimer
    - **=** : ne rien changer
3. Le droit que l'on veut modifier
    - **r** : **r**ead ⇒ lecture
    - **w** : **w**rite ⇒ écriture
    - **x** : e**x**ecute ⇒ exécution

Par exemple :

```
chmod o-w fichier3
```

enlèvera le droit d'écriture pour les autres.

```
chmod a+x
```

ajoutera le droit d'exécution à tout le monde.

On peut aussi combiner plusieurs actions en même temps :

- On ajoute la permission de lecture, d'écriture et d'exécution sur le fichier `fichier3` pour le **propriétaire** ;
- On ajoute la permission de lecture et d'exécution au **groupe propriétaire**, on retire la permission d'écriture ;
- On ajoute la permission de lecture aux **autres**, on retire la permission d'écriture et d'exécution.

```
chmod u+rwx,g+rx-w,o+r-wx fichier3
```

etc.

En octal

En octal, chaque « groupement » de droits (pour user, group et other) sera représenté par un chiffre et à chaque droit correspond une valeur :

- r = 4
- w = 2
- x = 1
- - = 0

Par exemple,

- Pour **rwx**, on aura : 4+2+1 = 7
- Pour **rw-**, on aura : 4+2+0 = 6
- Pour **r--**, on aura : 4+0+0 = 4

Récursivement

Pour chacune de ces commandes, on peut les lancer récursivement sur un répertoire. **C'est à dire que l'action sera effectuée sur le répertoire désigné et sur tous les fichiers ou répertoires qu'il contient.** Ceci ce fait en ajoutant l'option **-R**

Attention! Un `chmod -R` mal employé peut rendre votre système définitivement inutilisable.

Par exemple :

```
chmod -R 750 monDossier/
```

donnera tous les droits au propriétaire, les droits de lecture et exécution au groupe et aucuns droits aux autres…