R1.04 - Systèmes - TP 10
Etape 2 : Droits Unix
Le cours

INTERDICTION DE TRAVAILLER SUR UN ORDINATEUR PERSONNEL OU SUR CLÉ USB POUR CE TD !!! [1]

Sécurité

La sécurité des systèmes informatiques est une chose capitale pour les entreprises, et les particuliers aussi. Les intrusions dans les systèmes ont explosé avec l'avènement du tout Internet.

Ces intrusions ne sont pas forcément liées à une mauvaise maîtrise des droits sur le système mais une fois l'intrusion faite, un mauvais paramétrage des droits peut transformer un simple intrus en Power User, une vraie catastrophe pour l'entreprise et souvent aussi pour ses clients !

Un système, informatique ou autre, n'est jamais plus fort que son maillon le plus faible.

Des droits mal positionnés ou mal envisagés sont un maillon faible.

Il est donc primordial de comprendre ET maîtriser les droits Unix.

De façon générale, la gestion des droits ne concerne pas que les fichiers et les dossiers mais, dans ce TD, nous n'allons nous intéresser qu'à ceux en lien avec le Système de Fichiers.

Propriétaires

Sous Linux, un objet du FS a deux propriétaires :

- Un utilisateur propriétaire
- Un groupe propriétaire

[1] Ordinateur personnel autorisé pour prendre des notes mais pas pour y taper des commandes.

Dans un Terminal :

- Créez un dossier de travail **`tp10`**
- Déplacez-vous dans ce dossier.
- Créez un fichier **`test.txt`**
- Créer un dossier **`src`**
- Tapez un **`ls -l`** [2]

Vous devez voir un catalogue détaillé du fichier et du dossier présents dans **`tp10/`**, qui doit ressembler à ceci :

```
drwxr-xr-x 2 begood INFO-1 4096 nov.   24 11:40 src
-rw-r--r-- 2 begood INFO-1    0 nov.   24 11:40 test.txt
```

Vous connaissez déjà certains champs de cet affichage :

- **`src`** et **`test.txt`** sont les noms des objets du FS
- **`nov. 24 11:40`** est leur date+heure de dernière modification[3].
- **`4096`** et **`0`** sont les tailles des objets[4].

Les autres champs avaient été laissés sans explication jusqu'à présent, même si certains se devinent aisément.

La partie qui correspond aux droits est le bloc des 10 symboles de tête (**`drwxr-xr-x`** et **`-rw-r--r--`**). On va expliquer leur signification un peu plus loin car, pour pouvoir analyser les droits sur un objet, on doit avant tout déterminer qui est celui dont on veut analyser les droits.

Pour ce faire, on doit déjà identifier le propriétaire de l'objet. Il y a deux propriétaires possibles.

Utilisateur propriétaire

La partie **`begood`** qui, chez vous, doit être votre *login*, représente donc le nom du propriétaire de l'objet.

Tout objet qui est créé, par exemple, par un **`touch`**, un **`nano`**, un **`mkdir`**, ou encore un **`commande > fichier`**, etc. est automatiquement la propriété de l'utilisateur qui a lancé l'action qui a mené à la création de l'objet en question.

[2] Est-ce nécessaire de vous préciser qu'il s'agit d'un L minuscule ?
[3] L'heure est parfois remplacée par une année quand la date dépasse un an dans le passé. L'heure est toujours accessible mais n'est pas affichée, au profit de l'année.
[4] La taille pour un dossier n'est pas du tout représentative de la somme des tailles des fichiers qu'il contient. On expliquera cette valeur pour les dossiers l'année prochaine.

Sauf pour **root**, un utilisateur ne peut pas céder la propriété d'un objet qu'il possède à un autre utilisateur.

Groupe propriétaire

La partie **INFO-1** représente le groupe propriétaire de l'objet.

Par défaut, il s'agit du groupe principal de l'utilisateur qui a lancé l'action qui a mené à la création de l'objet.

Un utilisateur appartient toujours à un groupe principal et peut appartenir aussi à d'autres groupes, des groupes secondaires.

Sauf pour **root**, un utilisateur ne peut changer la propriété de groupe d'un objet que :

- S'il possède l'objet en tant d'Utilisateur
- S'il appartient aussi au Groupe auquel il va donner la propriété de Groupe.

Qui suis-je ?

Dans la suite de ce TP, on va s'intéresser aux droits que **vous** avez (ou n'avez pas) sur des objets.

Ceci nécessite donc de savoir qui on est. Ca peut paraître une question bizarre mais, même si on sait qui on est en tant qu'utilisateur, on ne connaît pas toujours tous les groupes auxquels on appartient.

Pour connaître votre appartenance (groupe principal et éventuels groupes secondaires), utilisez la commande suivante :

```
id
```

Vous obtenez quelque chose du genre :

```
uid=2000(begood) gid=1115(INFO-1) groups=1115(INFO-1),...
```

où :

- **uid** est le *User ID*, l'identifiant unique de l'utilisateur. Entre parenthèses se trouve un rappel, pour information, du nom qui correspond à cet **ID**, car l'OS ne connaît pas des noms d'utilisateurs mais seulement des **ID**s.
- **gid** est le *Group ID*, l'identifiant unique du groupe principal de l'utilisateur. Entre parenthèses se trouve un rappel, pour information, du nom de groupe qui correspond à cet **ID**, car l'OS ne connaît pas des noms de groupes mais seulement des **ID**s.

- **groups**[5] est la liste des *Group IDs* auxquels appartient l'utilisateur. Il y a toujours au moins un **ID** dans cette liste car le groupe principal y apparaît aussi, en 1ère position. Il peut y en avoir d'autres, qui sont donc les groupes secondaires (représentés ici par les **...**)

A propos des IDs

Pour vous montrer que l'OS ne connaît que des **ID**s et que les noms (utilisateurs et groupes) ne sont que des convenances pour rendre la vie plus agréable aux utilisateurs, lancez cette commande depuis votre Terminal qui doit toujours être ouvert sur le dossier **tp10/** :

```
ls -ln
```

Le **-n** permet d'afficher les **ID**s des propriétaires au lieu de leurs noms. Vérifiez que ces **ID**s correspondent bien à vous et votre groupe principal dont vous connaissez maintenant les **ID**s grâce à la commande **id** utilisée précédemment.

La correspondance **ID**s ⇔ Noms se trouve dans le fichier **/etc/passwd**, et la correspondance **ID**s ⇔ Groupes se trouve dans le fichier **/etc/groups**. A l'IUT c'est un peu différent car on utilise un annuaire réseau qui s'appelle **LDAP** et qui s'occupe de cette correspondance. Mais chez vous (ordinateur personnel), vous devriez retrouver cette correspondance.

Bloc des droits

Maintenant que l'on se connaît bien (on a identifié qui on est), on va s'intéresser aux droits qui sont placés sur les fichiers.

Ces droits sont décrits par 3 triplets de lettres situés à partir du 2ème caractère car le 1er caractère à gauche représente la nature du fichier (un **-** pour un fichier standard et un **d** pour un dossier). Ces 3 triplets s'appellent **UGO** dans le jargon Unix/Linux :

- **U** pour **U**ser
- **G** pour **G**roup
- **O** pour **O**thers

Chaque triplet (**U**, **G** et **O**) est constitué de 3 lettres :

- **r** pour le droit **r**ead (lecture)
- **w** pour le droit **w**rite (écriture)
- **x** pour le droit e**x**ecute (exécution)

[5] Il est possible que ce soit affiché en français : **groupes**

Ces lettres sont à des positions fixes : **r** suivi de **w** suivi de **x**. Si un certain droit est refusé, c'est un **-** (tiret) qui prend la place de la lettre pour indiquer l'absence du droit en question. Exemples :

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

Le 1er (**rw-**) indique des droits de lecture (**r**) et d'écriture (**w**) mais pas d'exécution (**-**)

Le 2ème (**--x**) indique l'interdiction de lecture (**-**) et d'écriture (**-**) mais donne un droit d'exécution (**x**)

Le 3ème (**r-x**) indique le droit de lecture (**r**) mais pas d'écriture (**-**) ainsi que le droit d'exécution (**x**)

Fichiers

Pour les fichiers, les droits doivent se comprendre ainsi :

- **r** : on peut lire le contenu du fichier
- **w** : on peut modifier le contenu du fichier
- **x** : on peut exécuter le fichier, c'est-à-dire faire un **./fichier** par exemple.

  Le fichier doit évidemment contenir du code exécutable (binaire ou script).

  Pour être réellement exécutable, il faudra aussi le droit de lecture (**r**) du fichier car il est évidemment impossible d'exécuter un code informatique issu d'un fichier qu'il est interdit de lire !

  Donc **x** va naturellement avec **r** pour que ce soit utile, même si on peut placer les droits (inutilement) différemment.

Pouvoir modifier un fichier, c'est-à-dire pouvoir écrire dans le fichier, ne nécessite pas forcément d'avoir le droit de le lire. Par exemple, on peut faire ceci dans un fichier sur lequel on a un droit **w** mais pas de droit **r** :

```
echo OK >> toto
```

On ajoute **OK** à la fin de **toto**. On ne fait qu'écrire dans le fichier sans le lire. On n'a besoin que du droit **w**, le droit **r** n'est pas nécessaire.

Par contre, éditer le fichier **toto** avec **nano** ou **VSC** nécessitera les deux droits (**r** et **w**) car ce sont des éditeurs de texte qui affichent le contenu du fichier pour pouvoir le modifier. Ça nécessite donc de pouvoir lire le fichier en amont.

Dossiers

Pour les dossiers, les droits sont proches de ceux des fichiers mais avec une petite subtilité.

> Pour bien comprendre les droits sur les dossiers, **il faut voir un dossier "comme" un fichier texte standard qui contiendrait la liste du contenu du dossier**.

Ainsi :

- **`r`** : on peut lire le contenu du dossier, c'est-à-dire lire la liste de son contenu. On peut donc faire un **`ls -l`** du dossier, par exemple.
- **`w`** : on peut modifier la liste de son contenu. C'est subtil, on en reparle plus bas.
- **`x`** : on peut traverser le dossier, c'est-à-dire que le nom du dossier a le droit d'apparaître dans un chemin, relatif ou absolu, en tant que dossier à traverser pour se rendre à l'objet de destination.

Reparlons du cas spécial du **`x`** sur un dossier :

- ce **`x`** est nécessaire pour faire une action sur un fichier dans le dossier en question car il faut traverser le dossier pour atteindre un fichier qu'il contient, même pour le supprimer, même pour le renommer. Par exemple, on écrit : **`rm dossier/fic`**
- ce **`x`** n'est pas nécessaire pour faire juste un **`ls`** du dossier : **`ls dossier`**
  on ne le traverse pas, il n'y a pas de **`/quelque_chose`** derrière.

Reparlons maintenant du cas spécial du **`w`** sur un dossier : que signifie "on peut modifier la liste de son contenu" ?

Pour bien comprendre, il faut se remettre en mémoire cette analogie qui a été faite juste avant : **il faut voir un dossier "comme" un fichier texte standard qui contiendrait la liste du contenu du dossier**.

Modifier un fichier texte c'est ajouter, supprimer ou altérer des lignes de texte dans le fichier.

Si on considère un dossier comme une liste de son contenu, c'est le considérer comme un fichier texte dont les lignes seraient les noms des fichiers et des dossiers qu'il contient.

Dans ce cas, modifier le dossier c'est modifier cette liste, et ce serait ajouter, supprimer ou altérer des lignes de la liste, donc des noms de fichiers ou de dossiers contenus.
ATTENTION, ce n'est pas toucher au contenu de ces fichiers mais juste à leur nom.

Ainsi, les actions concernées par le droit **`w`** sur un dossier sont d'autoriser ou non :

- la création d'un fichier ou d'un dossier à l'intérieur du dossier.
- la suppression d'un fichier ou d'un dossier (vide) à l'intérieur du dossier.

- le renommage d'un fichier ou d'un dossier

et ceci **PEU IMPORTENT LES DROITS** sur les objets en question !

C'est souvent contre intuitif mais pour comprendre il suffit d'imaginer un coffre dans un armoire : on peut mettre le coffre à la poubelle si on a la clé de l'armoire, même si on ne connaît pas le code du coffre !

Notez cependant que, pour les dossiers, le droit **`w`** doit aussi s'accompagner du droit **`x`**, sinon ça ne peut pas fonctionner : le droit **`x`** permet de traverser le dossier, autrement dit de pouvoir faire figurer le dossier dans un chemin. Ainsi, pour pouvoir spécifier un objet du dossier, il faudra bien le traverser, c'est-à-dire faire figurer le nom du dossier dans le chemin qui mène à l'objet ! Donc **`x`** est nécessaire pour que **`w`** ait un sens et une utilité sur un dossier.

Identifier les droits

Maintenant qu'on a vu la signification des triplets **`r`**, **`w`** et **`x`**, intéressons-nous à **`UGO`**.

Il n'y a pas un seul mais trois triplets. Chacun correspond à une des 3 lettres **`U`**, **`G`** et **`O`**. Les droits sur un objet vont dépendre du processus qui tente de faire une action sur l'objet. Un processus s'exécute toujours sous l'identité d'un utilisateur et d'un groupe. Si vous lancez un **`mkdir doss`**, la commande **`mkdir`** s'exécute sous votre identité. De cette identité découlent les droits qui s'appliquent.

Pour ce faire, il s'agit simplement d'appliquer un raisonnement en 2 questions, après avoir noté qui est l'utilisateur concerné. Dans ce TP ce sera généralement vous-même, et dans ce cas on utilise la commande **`id`** pour connaître le détail (utilisateur et groupes).

Les 2 questions (et donc 3 cas) :

① L'utilisateur est-il le propriétaire ? Oui, alors les droits sont ceux de **`U`** (**U**ser, 1er triplet)

② Sinon, l'utilisateur est-il membre du groupe propriétaire ? Oui, alors les droits sont ceux de **`G`** (**G**roup, 2ème triplet)

③ Sinon les droits sont ceux de **`O`** (**O**thers, 3ème triplet)

ATTENTION, en imaginant que l'utilisateur soit aussi membre du groupe propriétaire et que les droits de **`G`** lui soient plus favorables, ce sera quand même les droits **`U`** qui seront pris en compte. Il en découle que les "autres" (**O**thers) peuvent avoir des droits supérieurs à l'**U**tilisateur ou au **G**roupe. Même si c'est très peu probable, c'est tout à fait possible.

Changer les droits

On pourrait penser que changer les droits sur un objet nécessite, comme pour renommer un fichier, de posséder le droit d'écrire dans le dossier, "dans la liste des objets contenus dans le dossier"…

Il n'en est rien.

Les droits sur un objet du FS sont stockés dans les métadonnées associées au fichier[6] et pas dans le catalogue du dossier conteneur. Si on y réfléchit un instant, c'est normal, sinon le propriétaire du dossier conteneur pourrait se donner les droits de voir le contenu d'un fichier ! Repensez à l'histoire du coffre dans l'armoire.

Seul le propriétaire[7] de l'objet peut modifier les droits sur un cet objet. Même les membres du groupe propriétaire n'ont pas ce droit, peu importent les droits qu'ils possèdent sur l'objet.

La commande qui permet de changer les droits est **`chmod`**[8].

Il y a deux syntaxes.

Syntaxe symbolique

La syntaxe symbolique utilise les lettres **`u`**, **`g`** et **`o`** (minuscules) associées aux lettres **`r`**, **`w`** et **`x`** pour établir les droits.

Dans les exemples suivants, testez chaque **`chmod`** suivi d'un **`ls -l`** pour observer le résultat. Notez que les exemples suivants utilisent un seul fichier cible, mais qu'on peut évidemment préciser plusieurs fichiers cibles et aussi utiliser des jokers.

Avant de commencer, créez un fichier de travail :

```
touch fichier
umask 0000
```

La commande **`umask`** permet de modifier automatiquement certains droits par défaut, et un **`umask 0000`** annule cet automatisme.

Sans entrer dans les détails, on fait cela juste pour les besoins des exercices suivants. Vous n'aurez plus besoin de l'utiliser après ce TP, ou disons plutôt que le jour où vous en

[6] On appelle cela l'**`inode`** de l'objet, on le verra en BUT2.
[7] Ou **`root`** évidemment.
[8] **Ch**ange access **mod**e.

aurez besoin, vous aurez une connaissance en système suffisante pour savoir pourquoi et comment l'utiliser.

Par contre, si vous refermez ou changez de Terminal durant le TP, refaites cette commande le temps des exercices suivants.

Ajouter/supprimer des droits

Pour ajouter des droits on utilise **+** (plus), pour en supprimer on utilise **-** (moins). On peut les combiner en une seule expression et en mettre plusieurs séparées par des **,** (virgules)

Exemple 1 : global

```
ls -l
chmod +x fichier
ls -l
chmod -x fichier
ls -l
chmod -w fichier
ls -l
chmod +wx,-r fichier
ls -l
```

Les droits sont ajoutés ou supprimés pour chaque cible (**U**, **G** et **O**). Si les droits sont déjà présents ou absents, un **+** ou un **-** ne change rien.

Exemple 2 : par cible

```
chmod u+r fichier
ls -l
chmod go-w fichier
ls -l
chmod o-x,ug+rwx fichier
ls -l
```

En préfixant une expression des droits par une ou plusieurs des lettres **u**, **g** ou **o**, les droits sont ajoutés ou supprimés pour la cible ou la combinaison de cibles donnée (**U**, **G** et **O**). Les cibles non précisées ne sont pas impactées.

Affectation des droits

Pour affecter des droits de façon absolue (remplacement des droits par d'autres droits), on utilise **=** (égal). Le reste fonctionne comme pour **+** et **-**

Exemples :

```
ls -l
chmod u=xw fichier
ls -l
chmod go=rx fichier
ls -l
chmod u=x,o=,g=r fichier
ls -l
```

Les droits sont fixés exactement comme indiqués, pour la cible ou la combinaison de cibles donnée (**U**, **G** et **O**). Les cibles non précisées ne sont pas impactées. Un **=** (égal) tout seul supprime simplement tous les droits pour la cible.

chmod +x ?

Maintenant que vous avez compris que la commande **chmod** permet de modifier des droits sur un objet, vous devriez comprendre le fameux **chmod +x un_script** qu'on vous a fait faire plusieurs fois dans de précédents TP. Il permet donc d'ajouter un droit d'exécution à un fichier, un script généralement, pour pouvoir ensuite le lancer :

```
chmod +x un_script
./un_script
```

La raison est que, sans lui, le script n'est pas exécutable. Et vous comprenez aussi pourquoi il n'est pas nécessaire de répéter ce **chmod +x** à chaque modification ultérieure du fichier **un_script** : une fois le droit d'exécution placé, le fichier conserve ce droit, évidemment.

Syntaxe numérique

Pour utiliser la syntaxe numérique, il faut connaître la valeur de chaque lettre du triplet rwx :

- **r** = **4**
- **w** = **2**

- **x** = **1**

Les plus affûtés auront reconnu des puissances de 2. En fait, la syntaxe numérique s'écrit en octal (base 8). Il suffit d'ajouter les valeurs pour obtenir la correspondance symbolique ⇔ numérique :

- **rx** = **4+1** = **5**
- **rw** = **4+2** = **6**
- **rwx** = **4+2+1** = **7**
- **rien** = **0**

Chaque triplet est alors représenté par sa valeur numérique. Ainsi :

- **755** représente **rwxr-xr-x**
- **644** représente **rw-r--r--**
- **400** représente **r--------**

Avec la syntaxe numérique, on définit les droits *in extenso*, on ne peut pas juste donner les droits d'une partie **U**, **G** ou **O**. On donne **UGO** en intégralité. Exemple :

```
ls -l
chmod 642 fichier
ls -l
chmod 700 fichier
ls -l
chmod 000 fichier
ls -l
```

Changer les propriétaires

Seul **root** peut changer d'utilisateur propriétaire. Pour changer le propriétaire, on utilise **chown**[9], avec cette syntaxe :

```
chown nv_utilisateur objet
```

Pour changer de groupe ne peut se faire que par **root** ou par l'utilisateur propriétaire, à l'unique condition de faire partie du nouveau groupe propriétaire. On utilise **chgrp**[10], avec cette syntaxe :

```
chgrp nv_groupe objet
```

[9] **Ch**ange **Own**er
[10] **Ch**ange **Gr**ou**p**