

## 7.3

### Gérer des droits

NSI 1ÈRE - JB DUTHOIT

- Un utilisateur fait partie de groupe(s), dont un par défaut. En fait, le système d'exploitation l'identifie par un numéro (UID, identifiant d'utilisateur ou user ID) ainsi que ses groupes (par leur GID).

Utiliser la commande **id**, afin de déterminer les groupes auxquels appartient **alice** en précisant lequel est le principal (indiqué juste après **gid**). Vérifie qu'on retrouve également ces informations dans les fichiers (à afficher avec **cat**) **/etc/passwd** et **/etc/group**.

- Quand il crée un fichier ou un répertoire, ce dernier "appartient" à cet utilisateur, ainsi qu'à son groupe par défaut (des droits sont alors choisis, configurables avec la commande **umask**). Pour définir ces droits associés au fichier, on divise alors le monde en trois catégories :

- L'utilisateur propriétaire, désigné par **u**
- Les membres du groupe propriétaire (**g**)
- Tous les autres utilisateurs (**o** pour others)

Et pour chacune de ces catégories, on attribue ou non chacun des droits suivants :

- Lecture (**r** pour read) qui autorise donc la copie, pour un fichier ordinaire. Pour un répertoire, il permet d'obtenir la liste de ses fichiers.
- Écriture (**w** pour write) qui permet notamment la modification (pour un répertoire, l'ajout, la suppression, le renommage des fichiers qu'il contient).
- Exécution (**x** pour execute, qui indique pour un fichier ordinaire qu'il peut-être considéré comme une commande ; pour un répertoire, cela autorise à se positionner dedans, par exemple avec **cd**).

Avec **ls -l**, on a déjà visualisé ces permissions : après le premier caractère de la ligne, trois blocs de trois caractères **rwx** dont certains peuvent être remplacés par **-** indiquent (pour **u**, puis **g**, puis **o**) si l'on a accordé le droit (lettre présente) ou non (- présent). Détaillez les droits des dossiers de l'espace personnel d'**alice**, puis ceux des fichiers ordinaires.

- En fait, le propriétaire d'un fichier peut en modifier les droits avec **chmod**. Seul lui peut le faire, ainsi que le "super-utilisateur" (ou administrateur système) **root** qui a les pleins pouvoirs sur la machine (ce qui est donc dangereux : on ne l'utilise que quand cela est strictement nécessaire).

La syntaxe est **chmod modifications fichier**, où modifications est composé :

- du **public** (une ou plusieurs lettres parmi **u**, **g** et **o** définis ci-dessus, voire **a** pour « tous », soit all), puis
- d'un **opérateur** (= pour attribuer des droits et seulement ceux-là, + pour ajouter des droits à ceux déjà donnés et - pour en ôter à ceux qui existent) et enfin
- du ou des **droits** désignés par l'une ou plusieurs des lettres parmi **r**, **w** et **x**.

Exemples :

- `chmod o-wx mon_fichier` enlève les droits en écriture et en exécution aux « autres utilisateurs ».
- `chmod a+x mon_fichier` donne les droits en écriture et en exécution à tous les utilisateurs,
- On peut même donner plusieurs séries d'attributs, séparées par des virgules.  
`chmod ug=rwx,o=r mon_fichier` donne tous les droits à l'utilisateur et au groupe, mais seulement le droit en lecture aux autres utilisateurs.

Donner à ton tour la commande qui permet d'ôter les droits en écriture et en exécution au groupe et aux autres utilisateurs, pour `fichier1.txt`. Vérifie le résultat après exécution.

4. On peut utiliser comme précédemment \* et les autres motifs. Il est également possible de changer les droits récursivement c'est-à-dire en remontant aussi loin que nécessaire dans les sous-dossiers, en ajoutant l'option `-R`

Donne tous les droits à l'utilisateur et ne garde que ceux en lecture pour les autres catégories, pour tous les sous-dossiers du répertoire personnel d'`alice` dont le nom commence par un S. Puis seulement le droit en lecture pour tout utilisateur, pour tous les fichiers et sous-dossiers à un niveau quelconque du dossier `imdb`.

Vérifier le résultat.

5. Il existe **une autre manière de décrire les droits** à appliquer : on indique un nombre en écriture octale (base 8), dont chacun des trois chiffres (il y en a parfois 4, on n'étudie pas ce cas ici), associés respectivement à **u**, **g** et **o**, est calculé ainsi : on additionne les droits à attribuer, en comptant **4 pour r**, **2 pour w** et **1 pour x**. Par exemple, `rwx` vaut 7, `rx` vaut 5. On peut choisir directement des droits, mais pas en ajouter ou en ôter, avec cette notation.

Indiquer quels droits sont attribués par `chmod 754 fichier1.txt` puis vérifier-le.

6. La commande `umask` (user mask) sans argument renvoie un « masque » qui indique les droits attribués par défaut à la création d'un fichier. En donnant comme argument une valeur octale, on peut choisir une nouvelle valeur pour ce masque.

Pour déterminer la valeur, on soustrait *chiffre par chiffre* les droits calculés comme précédemment à 666 pour les fichiers ordinaires et à 777 pour les répertoires. Par exemple, pour obtenir les droits par défaut 644, soit `rw-r--r--`, sur les fichiers ordinaires, le masque vaut  $666 - 644 = 022$  : on exécute `umask 022`. Ainsi, sur les nouveaux répertoires, on aura les droits  $777 - 022 = 755$ , soit `rwxr-xr-x` (droit en exécution en plus de ceux des fichiers, autrement dit celui de se positionner dedans avec cd).

7. Les permissions dépendent des propriétaires (utilisateur et groupe) et on peut modifier ces derniers (là encore, seul `root` peut tout faire et un utilisateur peut changer le groupe d'un fichier qui lui appartient). Les commandes utilisées sont `chown` (change owner) et `chogrp` (change group) : `chown bob fichier1.txt` donnerait la propriété de ce fichier à l'utilisateur `bob`, si on a suffisamment de droits pour exécuter cette commande.

Il existe également des commandes pour ajouter un utilisateur à un groupe, que nous n'étudierons pas ici.

Donner la commande qui donne le groupe `developer` à `fichier2.txt`, puis vérifier le résultat.