


## Introduction aux systèmes d'exploitation (IS1)

### TP n° 4 : droits des fichiers et des répertoires

Le but de ce TP est d'apprendre à modifier les droits des fichiers et des dossiers, et de comprendre le rôle de ces droits. Comme d'habitude, il vous est demandé de déposer sur Moodle un fichier appelé `reponses_TP4.txt` contenant les commandes utilisées pour répondre aux questions marquées par le symbole . N'oubliez pas d'y insérer vos nom(s) et prénom(s), surtout si vous travaillez en binôme.


### Modifier les droits

« `chmod` » (**change mode**) avec deux arguments *modifs* et *fic*, permet d'appliquer au fichier *fic* les modifications de permissions décrites dans *modifs* ; *modifs* est une liste de la forme *modif<sub>1</sub>, ..., modif<sub>n</sub>* où chaque *modif<sub>i</sub>* est formée :

- de la liste des utilisateurs concernés, par concaténation des symboles *u* pour le propriétaire (*user*), *g* pour le groupe (*group*), *o* pour les autres (*other*), ou *a* pour tous (*all*) ;
- d'un symbole pour indiquer le type de modification : `=` pour affecter précisément des droits, `+` pour en ajouter, `-` pour en enlever ;
- de la liste des droits concernés, choisis parmi *r*, *w* ou *x*.

Par exemple, la ligne de commande `chmod a+r,u+w,go-wx` permet d'ajouter à tous le droit en lecture, au propriétaire le droit en écriture, et d'enlever aux membres du groupe et aux autres les droits en écriture et exécution.

### Exercice 1 – « `chmod` »

1. Télécharger depuis Moodle le script « `tp4.sh` » dans votre répertoire `~/Cours/IS1`. Modifier ses droits pour pouvoir l'exécuter, puis l'utiliser pour créer une arborescence de racine TP4.
2.  Faire en sorte que le fichier `TP4/A/titi` ait successivement les droits suivants (et vérifier !), en proposant une ligne de commande la plus compacte possible :
  - a. `-rwxr-xr-x 1 mimi staff 56 Oct 2 17:57 TP4/A/titi`
  - b. `-r-xr--r-- 1 mimi staff 56 Oct 2 17:57 TP4/A/titi`
  - c. `-r--r--r-- 1 mimi staff 56 Oct 2 17:57 TP4/A/titi`
  - d. `----r----- 1 mimi staff 56 Oct 2 17:57 TP4/A/titi`
  - e. `--wx--xr-- 1 mimi staff 56 Oct 2 17:57 TP4/A/titi`
  - f. `--w---xr-x 1 mimi staff 56 Oct 2 17:57 TP4/A/titi`

**Utiliser la notation octale** Outre la notation symbolique, la commande « `chmod` » peut également être utilisée avec la syntaxe suivante :

`chmod abc fic`

où `abc` est l'écriture octale (i.e. en base 8) du nombre dont la représentation binaire  $r_u w_u x_u r_g w_g x_g r_o w_o x_o$  respecte la règle suivante : le bit  $r_u$  est à 1 si le propriétaire a le droit de lecture et à 0 sinon, le bit  $w_u$  est à 1 si le propriétaire a le droit d'écriture et à 0 sinon et  $x_u$  est à 1 si le propriétaire a le droit d'exécution et à 0 sinon ; de même les bits  $r_g, w_g, x_g$  correspondent aux droits du groupe propriétaire, et les bits  $r_o, w_o, x_o$  correspondent aux droits des autres utilisateurs.

Par exemple, les droits `rw-r--r--` (dans la notation de « `ls -l` ») s'écrivent donc en octal 754.

### Exercice 2 – « `chmod` » et droits en octal

✎ En utilisant la notation octale, donner aux fichiers ordinaires de la sous-arborescence TP4/B les droits suivants :

```
$ ls -l TP4/B/*/*
-r--r--r-- 1 mimi staff 0 Oct 2 17:57 TP4/B/A/toto
-rwx-wxr-x 1 mimi staff 0 Oct 2 17:57 TP4/B/A/tutu
-r-xrw---x 1 mimi staff 0 Oct 2 17:57 TP4/B/B/toto
--w---xr-x 1 mimi staff 0 Oct 2 17:57 TP4/B/B/tutu
-rw-r--r-- 1 mimi staff 0 Oct 2 17:57 TP4/B/C/toto
--wx--x--- 1 mimi staff 0 Oct 2 17:57 TP4/B/C/tutu
```

### Exercice 3 – choisir la notation adaptée

✎ Effectuer chacune des modifications suivantes *en une seule commande*, en utilisant la notation (octale ou symbolique) qui vous semble la plus adaptée :

1. Donner à TP4/B/A/toto les droits `r-xr--r--`.
2. Donner à TP4/B/A/tutu les droits `rw--w-r--`.
3. Donner à TP4/B/B/toto les droits `-w---xrw-`.
4. Donner à TP4/B/B/tutu les droits `rw-r-xr-x`.
5. Donner les droits `-w-r-xr--` aux deux fichiers du répertoire TP4/B/C.
6. Ajouter les droits en exécution à tous les utilisateurs sur les trois fichiers dont le nom de base est tutu (dans la sous-arborescence TP4/B).
7. Donner les droits `rw-r-----` sur les trois fichiers dont le nom de base est toto.

## Effets des droits, suite

### Exercice 4 – autour des fichiers exécutable

1. Essayer d'exécuter les fichiers TP4/A/toto et TP4/A/tutu. Recommencer après leur avoir supprimé le droit en lecture.  
➤ Que constatez-vous ? Comment cela s'explique-t-il ?
2. ➤ Dans votre répertoire TP4, créer un fichier youpi puis l'éditer avec un éditeur de texte (tel « emacs ») pour qu'il contienne la ligne suivante :  
echo "Moi aussi, j'y arrive !!!"  
puis changer les droits de youpi afin de pouvoir l'utiliser comme une commande.

### Exercice 5 – effets des droits sur les répertoires

1. Quel(s) droit(s) avez-vous sur chacun des sous-répertoires de TP4 ? Pouvez-vous lister leur contenu ? ➤ Que pouvez-vous en conclure sur le rôle du droit en lecture pour les répertoires ?
2. Pouvez-vous renommer le fichier TP4/B/titi ? Le supprimer ? Créer un nouveau fichier dans TP4/B (avec la commande « touch ») ? Pouvez-vous remplacer le contenu de TP4/B/titi par « *Damned, je suis fait !* » ?  
➤ Que pouvez-vous en conclure sur le rôle du droit en écriture pour les répertoires ?
3. Dans le répertoire TP4/ créez un répertoire Gringotts/, dans lequel vous créerez un fichier Gringotts/coffre et un sous-répertoire Gringotts/Caveaux. Vérifiez que vous avez les droits rwx sur ces deux répertoires et les droits rw sur le fichier coffre. Placez-vous dans le répertoire TP4/ pour les questions suivantes.
  - ➤ A l'aide de la commande echo écrivez le mot "Lingots" dans le fichier coffre en utilisant sa référence relative.
  - ➤ A l'aide de la commande touch créez deux fichiers cav1 et cav2 dans le répertoire Caveaux en utilisant les références relatives.
  - Essayez la commande cat sur coffre. Essayez la commande ls sur Gringotts. Essayez la commande ls sur Caveaux.
  - Exécutez maintenant la commande chmod u-x Gringotts/ et ensuite ressayer les trois commandes précédentes. ➤ Lesquelles de ces commandes fonctionnent toujours ? Pourquoi ?
  - Essayez d'ajouter un nouveau cav3 dans Caveaux ou de modifier le contenu de coffre avec echo.
  - Essayez d'effacer l'arborescence dont Gringotts est racine.
  - Exécutez maintenant la commande chmod u+x Gringotts/ et ensuite effacez l'arborescence.

4. Pouvez-vous lister le contenu de toute l'arborescence de racine TP4/C? Pouvez-vous consulter les droits du fichier TP4/C/toutou? Supprimer TP4/C/toutou? Créer TP4/C/turlu? Vous déplacer dans TP4/C? ➤ Que pouvez-vous en conclure sur le rôle du droit en exécution pour les répertoires?
5. Le répertoire TP4/D contient un fichier de nom tata. Pouvez-vous consulter ses droits? Son contenu? Qu'en est-il pour le fichier TP4/D/D/tatou? ➤ Conclure quant aux rôles respectifs du droit en lecture et du droit en exécution pour un répertoire.
6. ➤ Donner au répertoire TP4/C les droits minimaux pour pouvoir lire le fichier TP4/C/toutou.
7. ➤ Faire le nécessaire pour déplacer TP4/C/toutou dans TP4/B.
8. ➤ Faire le nécessaire pour consulter le contenu du répertoire TP4/D. Pouvez-vous lister tout le contenu de l'arborescence de racine TP4/D? Faire les modifications nécessaires (*et rien de plus*).
9. ➤ Faire ce qu'il faut pour pouvoir supprimer TP4/D.

## Droits par défaut des nouveaux fichiers

Lorsque de nouveaux fichiers ou répertoires sont créés, des droits par défaut leur sont attribués. Ces droits sont calculés à partir d'un ensemble de droits par défaut en utilisant un *masque des droits par défaut des fichiers utilisateurs* (appelé *umask*), qui peut être défini à l'aide de la commande « *umask* » (en utilisant la (les) même(s) syntaxe(s) que pour « *chmod* »).

Par exemple, la commande « *umask o-r* » retire aux utilisateurs ne faisant pas partie de son propre groupe l'accès en lecture aux *nouveaux* fichiers. Attention, la portée de cette modification est limitée à la session courante.

La syntaxe octale peut également être utilisée comme suit :

```
umask abc
```

où abc est l'écriture en base 8 du nombre dont la représentation binaire suit la même logique que pour « *chmod* », à ceci près qu'un bit est à 1 si le droit correspondant est **prohibé** par le masque et est à 0 sinon. Par exemple, « *umask 777* » crée un masque tel que les nouveaux fichiers n'offriront par défaut aucun droit à aucun utilisateur.

### Exercice 6 – « *umask* »

1. Comparer les réponses obtenues avec « *umask* » et « *umask -S* ». ➤ Expliquer.
2. Dans votre répertoire Test (à créer s'il n'existe pas), créer un nouveau fichier toto et un nouveau répertoire Rep, et comparer leurs droits. ➤ Que constatez-vous?

3. Saisir la ligne de commande « `umask u=rwx,g=rx,o=` », puis créer un nouveau fichier `titi` et un répertoire `Rep1` (depuis le même terminal). ➤ Expliquer les différences de droits avec `toto` et `Rep`.
4. Dans un autre terminal, créer un répertoire `Rep2`. Que constatez-vous ?
5. Dans ce deuxième terminal, saisir la ligne `umask 027`, puis créer un répertoire `Rep3`. Expliquer les différences de droits avec `Rep2`.
6. ➤ Définir un *umask* très restrictif qui interdit à quiconque à part vous l'accès en lecture ou en écriture, ainsi que la traversée de vos répertoires. Tester sur un nouveau fichier et un nouveau répertoire.
7. ➤ Définir un *umask* très permissif qui autorise tout le monde à lire vos fichiers et traverser vos répertoires, mais n'autorise que vous à écrire. Tester sur un nouveau fichier et un nouveau répertoire.
8. ➤ Définir un *umask* équilibré qui vous autorise un accès complet et autorise un accès en lecture et exécution aux membres de votre groupe, et un accès en exécution seule aux autres utilisateurs. Tester sur un nouveau fichier et un nouveau répertoire.
9. ➤ Si le masque par défaut ne vous satisfait pas, comment pouvez-vous le changer de manière pérenne ?

## Partager ses fichiers et répertoires

Parfois, par exemple dans le cadre d'un projet à plusieurs ou de la fabrication d'une page web, il peut être nécessaire de donner accès à certains de ses fichiers ou répertoires à d'autres utilisateurs. Les exercices suivants montrent deux cas d'utilisation possibles.

### Exercice 7 – donner accès à ses fichiers

1. Donner à votre répertoire personnel le droit d'exécution (donc de parcours) pour tout le monde (mais aucun autre droit). Faire de même pour le sous-répertoire `Test`.
2. Dans le répertoire `Test`, créer un fichier `journal_intime` et lui attribuer les droits suffisants pour qu'une autre personne de votre groupe UNIX puisse y accéder en lecture (mais pas en écriture). Donner à un(e) voisin(e) toutes les informations nécessaires pour lire le contenu de `journal_intime`.

### Exercice 8 – ouvrir un répertoire d'accueil

1. Dans votre répertoire `Test`, créer un sous-répertoire `Public`, et y copier votre `journal_intime`. Donner les droits `rxwxrwx---` à `Public`, mais protéger `journal_intime` de façon à ce que vous soyez l'unique utilisateur à pouvoir le lire.

2. Demander à un(e) voisin(e) de modifier votre répertoire Public en copiant `journal_intime`, puis en le renommant, et enfin en le supprimant. ➤ Conclusion ?
3. ➤ Faire les manipulations nécessaires afin que votre voisin(e) crée dans votre arborescence personnelle un fichier `voisin` tel que :
  - lui ou elle seul(e) puisse écrire dans ce fichier,
  - tout autre membre du groupe puisse lire ce fichier,
  - vous seul(e) puissiez effacer ce fichier.

## Un droit très spécial...

On a parfois besoin d'un répertoire accessible à tous, ouvert en particulier en écriture. Comme on l'a vu, cela pose un problème de taille : tout utilisateur peut alors supprimer n'importe quel fichier, même s'il ne lui appartient pas. Le droit *sticky bit* est utilisé pour gérer ce type d'accès particulier. Il est symbolisé par un `t`, qui remplace le `x` des autres utilisateurs dans l'affichage de « `ls -l` ». Sa valeur octale est 1000.

### Exercice 9 – (optionnel) `/tmp` et le *sticky bit*

Le répertoire `/tmp` peut être utilisé par tous les utilisateurs pour y stocker temporairement des fichiers. Il est en général vidé à chaque redémarrage de la machine.

1. Consulter les droits de `/tmp`.
2. Vérifier que vous pouvez créer et supprimer des fichiers dans `/tmp`.
3. Pouvez-vous supprimer les fichiers des autres utilisateurs ? Les renommer ?