Chapitre 7 : Droits d'accès aux fichiers

A. Concepts de comptes utilisateur et de groupes 119

1. Hiérarchie des utilisateurs 120
2. Commandes utiles 121

B. Droits Unix 121

1. Droits standards 122
2. SUID, SGID et Sticky Bit 125

C. Gestion des droits 129

1. chgrp 129
2. chmod 129
3. umask 133
4. Gestionnaires de fichiers 134

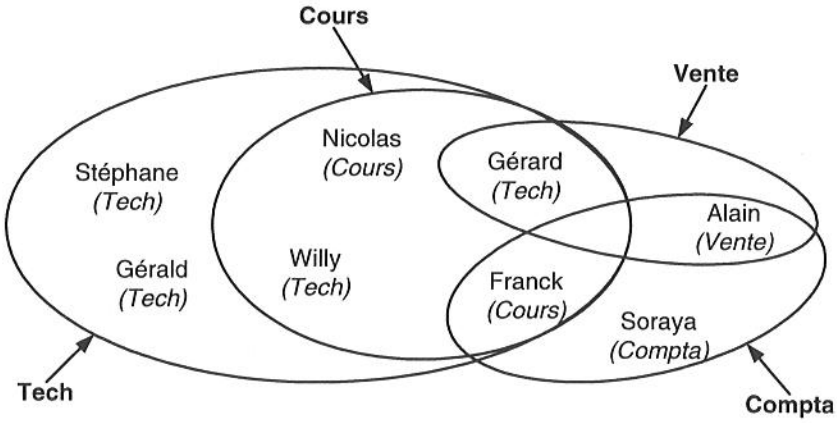
A. Concepts de comptes utilisateur et de groupes

Le système GNU/Linux étant multiutilisateur, les personnes employant celui-ci doivent être identifiées afin d'assurer la confidentialité des informations contenues dans les fichiers. En effet, il ne serait pas acceptable que l'utilisateur "Nicolas" puisse consulter les fichiers personnels de "Stéphane" sans l'accord de ce dernier.

Ces personnes possèdent donc chacune un "compte utilisateur" sur le système ; elles peuvent utiliser ce dernier tout en étant clairement identifiées. Cependant, il est permis de partager des fichiers entre collaborateurs et une notion de "groupe d'utilisateurs" existe sous GNU/Linux.

Un utilisateur doit obligatoirement être membre d'un groupe d'utilisateurs sur un système Unix comme GNU/Linux : c'est son groupe principal qui est utilisé lors de la création des fichiers. Par contre, il peut éventuellement appartenir à plusieurs autres groupes : ses groupes secondaires déterminent ses droits d'accès aux fichiers créés par d'autres membres des groupes.

Par exemple, si l'on représente les différents services d'une société avec leurs personnels, bien que chaque individu ait une fonction première (indiquée entre parenthèses), certains peuvent assumer plusieurs missions :



On voit ici que :

- Stéphane et Gérald appartiennent tous les deux au service technique (Tech).
- Nicolas, qui est avant tout formateur (Cours), fait aussi partie du service technique (Tech).
- Willy, appartenant au service technique (Tech) principalement, travaille aussi dans le service formation (Cours).
- Franck est un formateur (Cours) qui collabore avec les services technique (Tech) et comptabilité (Compta).
- Gérard, du service technique (Tech), offre ses compétences au service commercial (Vente).

- Alain est un commercial (Vente) qui s'acquitte aussi de tâches administratives (Compta).
- Soraya fait uniquement partie du service comptabilité (Compta).

Pour identifier tous ces utilisateurs au niveau du système d'exploitation, un numéro unique leur est attribué : l'UID ("User's ID") ; le propriétaire d'un fichier est déterminé par ce numéro sous Unix. Ces utilisateurs sont aussi dotés d'un nom d'utilisateur unique ("login") et d'un mot de passe ("password") pour qu'ils puissent s'authentifier lors de leur connexion au système.

De la même manière, les groupes d'utilisateurs sont représentés par un nom unique auquel est associé un identifiant numérique : le GID ("Group's ID"). Ce dernier est également utilisé pour déterminer le groupe propriétaire d'un fichier.

1. Hiérarchie des utilisateurs

Les utilisateurs, et par conséquent les comptes utilisateur, ne sont pas tous égaux sous Unix. On peut distinguer trois types de comptes :

`root`

C'est l'utilisateur le plus important du système du point de vue de l'administration. Il n'est pas concerné pas les droits d'accès aux fichiers et peut faire à peu près tout sur le système, excepté écrire sur un système de fichiers monté en lecture seule (CD-Rom). Son UID égal à 0 lui confère sa spécificité. Ce "super-utilisateur" a donc à sa charge les tâches d'administration du système. Pour éviter toute erreur de manipulation, il est fortement conseillé d'utiliser le compte d'administration uniquement pour les tâches nécessitant les droits du super-utilisateur.

`bin, daemon, sync, apache...`

Il existe sur le système une série de comptes qui ne sont pas affectés à des personnes physiques. Ceux-ci servent à faciliter la gestion des droits d'accès de certaines applications et démons. Les UID compris entre 1 et 499 sont généralement utilisés pour ces comptes.

`franck, nicolas...`

Tous les autres comptes utilisateur sont associés à des personnes réelles ; leur vocation est de permettre à des utilisateurs standards de se connecter et d'utiliser les ressources de la machine. L'UID d'un utilisateur est normalement un nombre supérieur à 499.

On appelle "démons" les programmes s'exécutant en tâche de fond, comme un serveur web ou un serveur d'impression.

À l'instar des comptes utilisateur, il existe différents types de groupes sur un système GNU/Linux permettant de donner des droits communs à un ensemble d'utilisateurs :

root

Son GID est 0 et c'est le groupe principal de l'administrateur.

bin, daemon, sync, apache...

Ces groupes ont le même rôle que les comptes du même nom et permettent de donner les mêmes droits d'accès à un ensemble d'applications. Par convention, les groupes système ont un GID compris entre 1 et 499.

cours, tech...

Ces groupes représentent un ensemble de personnes réelles devant accéder aux mêmes fichiers. Typiquement, ils ont un GID supérieur ou égal à 500.

2. Commandes utiles

les commandes **id** et **groups** permettent d'afficher les informations en rapport avec les groupes. La première donne l'UID de l'utilisateur, le GID de son groupe principal et les GID de tous les groupes auxquels il appartient. La seconde ne fournit que la liste complète des groupes mais accepte plusieurs noms d'utilisateurs en argument :

```
[nicolas]$ whoami
nicolas
[nicolas]$ id
uid=500(nicolas) gid=500(cours) groupes=500(cours),501(tech)
[nicolas]$ id stephane
uid=502(stephane) gid=501(tech) groupes=501(tech)
[nicolas]$ groups
cours tech
[nicolas]$ groups gerard alain willy root
gerard : tech cours vente
alain : vente compta
willy : tech cours
root : root
```

B. Droits Unix

Les permissions d'accès aux fichiers déterminent les actions que peuvent entreprendre les utilisateurs.

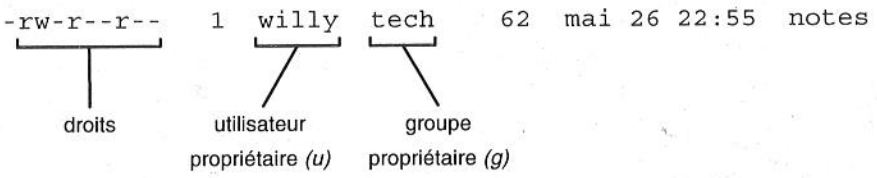
La majorité des problèmes d'installation, de configuration et de fonctionnement des applications sous GNU/Linux est due à des droits d'accès mal positionnés.

En premier lieu, il est nécessaire de savoir que les droits d'accès sous Linux sont définis pour :

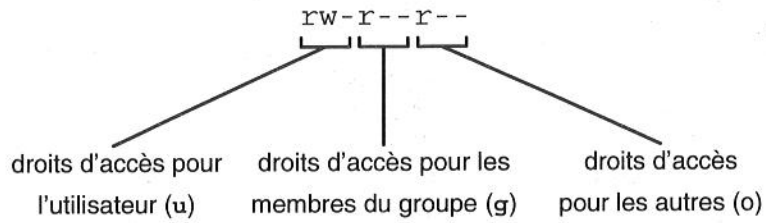
- Un compte utilisateur : propriétaire du fichier, c'est en principe l'utilisateur qui a créé celui-ci.
- Un groupe : ce groupe est généralement le groupe principal du propriétaire du fichier mais peut être modifié par ce dernier et prendre la valeur d'un de ses groupes secondaires.
- Les autres : cette entité représente toute personne autre que le propriétaire et qui n'est pas membre du groupe cité précédemment.

Les droits d'accès à un fichier sont aussi appelés "mode" sous Unix.

Les droits, l'utilisateur et le groupe propriétaires d'un fichier sont affichés avec la commande `ls -l` :



Dans ce dernier exemple, le fichier appartient à l'utilisateur **willy** et au groupe **tech** ; les neuf caractères **rw-r--r--** définissent les droits d'accès à ce fichier pour l'utilisateur **willy** ("user" ou "u"), les membres du groupe **tech** ("group" ou "g") et les autres ("other" ou "o"). Plus précisément, ces caractères sont répartis comme suit :



Tout utilisateur est donc associé à l'une de ces entités pour déterminer les permissions en vigueur.

Attention, si l'utilisateur est propriétaire du fichier, ce sont les droits du propriétaire qui s'appliquent et non ceux du groupe, même si cet utilisateur est aussi membre de ce groupe.

1. Droits standards

Les droits d'accès fondamentaux sur les fichiers et les répertoires sous Unix/Linux sont les droits de lecture **r** ("Read"), d'écriture **w** ("Write") et d'exécution **x** ("eXecute").

Ces droits – définis pour les entités **u**, **g** et **o** – apparaissent dans l'ordre **r**, suivi de **w**, lui-même suivi de **x** avec la commande **ls -l**. Lorsque l'un de ces caractères est remplacé par un tiret, cela signifie que le droit associé n'est pas autorisé.

Dans l'exemple du paragraphe précédent, l'utilisateur **willy** qui a les droits **rw-** :

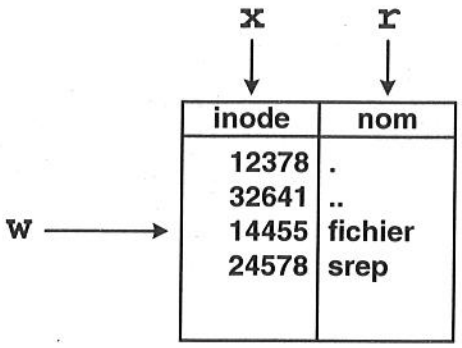
- A le droit de lire le fichier *notes*.
- A le droit de modifier le fichier *notes*.
- N'a pas le droit d'exécuter le fichier *notes*.

De façon plus précise, on distingue les droits Unix standards selon le type de fichier : fichier ordinaire ou répertoire.

droit	fichier	répertoire
r	autorisation de lire le contenu du fichier	autorisation de lister les entrées du répertoire
w	autorisation de modifier le contenu du fichier	autorisation de modifier les entrées du répertoire
x	autorisation d'exécuter le fichier	autorisation d'accéder aux entrées du répertoire

S'il est relativement simple de retenir les autorisations correspondantes lorsque ces droits sont positionnés pour un fichier ordinaire, cela devient moins évident pour un répertoire.

En considérant les répertoires comme des tableaux contenant, dans une colonne les inodes et dans une autre les noms des fichiers présents dans le répertoire, il est plus facile d'appréhender les droits standards :



On voit bien ici que les droits **r** et **x** doivent être positionnés afin d'accéder au fichier *fichier* et au sous-répertoire *srep* : **r** permettant de connaître leur nom et **x** leur numéro d'inode (nécessaire pour les manipuler puisque c'est l'inode qui permet d'accéder aux données d'un fichier et à la liste des fichiers contenus dans un répertoire).

En y ajoutant le droit **w**, il est alors possible de créer/modifier/supprimer une entrée dans le répertoire.

Cette dernière remarque est très importante car cela signifie que ce sont les permissions associées au répertoire qui définissent le droit de vie ou de mort des fichiers à l'intérieur de celui-ci.

Exemple : suppression d'un fichier non accessible en lecture/écriture

L'exemple suivant illustre le cas d'un fichier non accessible en lecture/écriture mais pouvant être supprimé malgré tout, à cause des droits associés au répertoire.

Par exemple, en tant qu'utilisateur **alain** :

```
[alain]$ ls -ld rep
drwxrwxrwx  2 alain vente 4096 jun  3 01:51 rep
[alain]$ ls -l rep
total 4
-rw-------  1 alain vente 47 jun  3 01:51 secret
```

Maintenant, en tant que **nicolas** :

```
[nicolas]$ ls -l rep
total 4
-rw-------  1 alain vente 47 jun  3 01:51 secret
[nicolas]$ cat rep/secret
cat: rep/secret: Permission denied
[nicolas]$ rm rep/secret
rm: détruire un fichier protégé en écriture fichier régulier
'rep/secret'? o
[nicolas]$ ls -l rep
total 0
```

Exemple : droits d'accès sur les chemins

Le droit d'accéder à un fichier n'est pas uniquement défini par les permissions données à ce fichier et au répertoire qui le contient ; il faut avoir le droit de traverser tous les répertoires spécifiés dans le chemin d'accès au fichier.

Ainsi pour accéder en lecture au fichier */rep1/rep2/rep3/fic*, il faut posséder le droit **r** sur le fichier `fic` mais aussi les droits **r** et **x** sur les trois répertoires *rep1*, *rep2* et *rep3*.

Pour le chemin suivant :

```
[root]# ls -ld /rep1
drwxr-xr-x  3 root root 4096 jun  3 01:55 /rep1/
[root]# ls -ld /rep1/rep2
drwxr-xr--  3 root root 4096 jun  3 01:55 /rep1/rep2/
[root]# ls -ld /rep1/rep2/rep3
drwxr-xr-x  2 root root 4096 jun  3 01:55 /rep1/rep2/rep3/
```

```
[root]# ls -l /rep1/rep2/rep3
total 4
-rw-r--r--  1 root root 23 jun  3 01:55 fic
```

L'utilisateur **alain**, auquel les droits de l'entité "others" s'appliquent, ne peut accéder au fichier car il ne possède pas les droits suffisants sur le répertoire */rep1/rep2* :

```
[alain]$ cat /rep1/rep2/rep3/fic
cat: /rep1/rep2/rep3/fic: Permission denied
[alain]$ cd /rep1
[alain]$ ls -l
total 4
drwxr-xr--  3 root root 4096 jun  3 01:55 rep2
[alain]$ cd rep2
-bash: cd: rep2: Permission denied
```

Exemple : fichier journal

Certains fichiers peuvent posséder des droits d'accès atypiques à première vue. Par exemple, un fichier journal contenant les messages de plusieurs applications doit être accessible en modification pour différents processus mais pas nécessairement accessible en lecture pour des raisons de sécurité.

On peut comparer ce type de fichier à une urne de vote où chacun peut inscrire des données mais où seul le propriétaire du fichier – de l'urne – possède l'autorisation de lire l'ensemble du contenu.

```
[alain]$ ls -l urne
-rw------w-  1 root root 0 jun  3 02:01 urne
```

2. SUID, SGID et Sticky Bit

Les trois droits supplémentaires SUID, SGID et Sticky Bit sont moins connus mais tout aussi indispensables dans la gestion des autorisations sous Linux.

Comme pour les droits standards, on peut différencier la signification de ces droits étendus selon le type de fichier considéré : fichier ou répertoire.

droit	fichier	répertoire
SUID	Exécute le fichier sous l'identité du propriétaire du fichier.	
SGID	Exécute le fichier sous l'identité du groupe du fichier.	Les fichiers créés dans le répertoire héritent du groupe du répertoire à la place du groupe principal de l'utilisateur.

droit	fichier	répertoire
Sticky	L'image de l'exécutable reste en mémoire, son rechargement est alors plus rapide (cette utilisation n'a plus raison d'être avec la gestion actuelle de la mémoire).	Seul le propriétaire du fichier ou du répertoire peut supprimer les fichiers.

Pour des raisons de sécurité, les droits d'endossement SUID et SGID fonctionnent uniquement pour des fichiers binaires (code compilé) et non pour les scripts (à l'exception des scripts Perl).

La représentation de ces droits dans l'affichage de la commande `ls -l` est la suivante :

SUID

Le caractère `x` des droits du propriétaire est remplacé par un `s` lorsque ce droit est positionné ; un **S** (majuscule) signifie que le droit en exécution n'est pas positionné en même temps.

SGID

Idem mais au niveau des droits du groupe.

Sticky Bit

Un `t` est indiqué à la place du `x` au niveau des droits de l'entité "others". Comme précédemment, si ce droit est en majuscule (`T`), cela signifie que le droit `x` qui est masqué n'est pas activé.

Par exemple :

```
[root]# ls -l /usr/bin/passwd
-r-s--x--x    1 root     root        15368 mai 28  2002 /usr/bin/passwd
[root]# ls -l /usr/bin/write
-rwxr-sr-x    1 root     tty         18605 août 30 22:00 /usr/bin/write
[root]# ls -ld /tmp/
drwxrwxrwt    5 root     root         1024 déc 12 15:12 /tmp/
```

Exemple : gestion des mots de passe

En temps normal, les programmes endossent l'identité des utilisateurs qui les exécutent. Ceci implique qu'il n'est pas possible pour un utilisateur d'outrepasser ses droits d'accès à un fichier en employant une commande particulière puisque c'est la commande qui a les mêmes autorisations.

Cependant, le fichier */etc/shadow* qui contient les mots de passe des comptes utilisateur sur le système possède les droits suivants :

```
[alain]$ ls -l /etc/shadow
-r--------  1 root root 1087 mai 27 00:09 /etc/shadow
```

Seul l'administrateur a le droit de lecture sur ce fichier. Pourtant, les utilisateurs peuvent employer la commande **passwd** pour modifier leur mot de passe.

Ceci est dû au bit SUID qui donne au programme les droits du propriétaire du fichier (et donc de root) et non ceux de l'utilisateur qui l'a lancé :

```
[alain]$ ls -l /usr/bin/passwd
-r-s--x--x  1 root root 15760 jui 21  2003 /usr/bin/passwd
```

Exemple : travail collaboratif

Dans l'exemple de groupes d'utilisateurs présenté en début de chapitre, les utilisateurs **nicolas** et **willy** sont tous deux membres des groupes **cours** et **tech**. Par contre, le groupe principal du premier est **cours** et celui du second **tech**.

```
[root]# id nicolas
uid=500(nicolas) gid=500(cours) groupes=500(cours),501(tech)
[root]# id willy
uid=504(willy) gid=501(tech) groupes=501(tech),500(cours)
```

Pour qu'ils puissent collaborer ensemble aussi bien sur des projets techniques que des formations, il faut que les fichiers qu'ils créent appartiennent au groupe adéquat. Or, par défaut les fichiers créés par Nicolas appartiennent au groupe **cours** et ceux créés par Willy au groupe **tech**.

Ce problème est résolu en créant un répertoire de travail par groupe d'utilisateurs et en positionnant le droit SGID dessus, avec le groupe adéquat comme groupe propriétaire :

```
[root]# ls -l
total 8
drwxrwsr-x  2 root cours 4096 jun  3 02:23 formation/
drwxrwsr-x  2 root tech  4096 jun  3 02:23 technique/
```

Lorsque l'utilisateur **nicolas** crée des fichiers :

```
[nicolas]$ touch fic1 formation/fic2 technique/fic3
[nicolas]$ ls -lR
.:
total 8
-rw-r--r--  1 nicolas cours    0 jun  3 02:27 fic1
drwxrwsr-x  2 root    cours 4096 jun  3 02:27 formation/
drwxrwsr-x  2 root    tech  4096 jun  3 02:27 technique/
```

```
./formation:
total 0
-rw-r--r--  1 nicolas cours 0 jun  3 02:27 fic2

./technique:
total 0
-rw-r--r--  1 nicolas tech 0 jun  3 02:27 fic3
```

Lorsque l'utilisateur **willy** crée des fichiers :

```
[willy]$ touch ficA formation/ficB technique/ficC
[willy]$ ls -lR
.:
total 8
-rw-r--r--  1 nicolas cours    0 jun  3 02:27 fic1
-rw-r--r--  1 willy   tech     0 jun  3 02:27 ficA
drwxrwsr-x  2 root    cours 4096 jun  3 02:27 formation/
drwxrwsr-x  2 root    tech  4096 jun  3 02:27 technique/

./formation:
total 0
-rw-r--r--  1 nicolas cours 0 jun  3 02:27 fic2
-rw-r--r--  1 willy   cours 0 jun  3 02:27 ficB

./technique:
total 0
-rw-r--r--  1 nicolas tech 0 jun  3 02:27 fic3
-rw-r--r--  1 willy   tech 0 jun  3 02:27 ficC
```

Exemple : espace de stockage partagé /tmp

Les droits standards Unix font que tout utilisateur possédant les droits **rwx** sur un répertoire a la possibilité de créer des fichiers dans celui-ci, mais aussi de supprimer tous ceux qui s'y trouvent.

Le répertoire temporaire */tmp* autorise tout utilisateur à y stocker ses propres fichiers mais pour éviter qu'un utilisateur ne supprime les fichiers d'un autre, le droit Sticky doit être positionné :

```
[willy]$ ls -ld /tmp
drwxrwxrwt  5 root root 4096 jun  3 02:32 /tmp/
[willy]$ touch /tmp/fichierDeWilly
[willy]$ ls -l /tmp
total 0
-rw-r--r--  1 nicolas cours 0 jun  3 02:32 fichierDeNicolas
-rw-r--r--  1 willy   tech  0 jun  3 02:32 fichierDeWilly
```

```
[willy]$ rm /tmp/fichierDeNicolas
rm: détruire un fichier protégé en écriture fichier régulier vide '/tmp
/fichierDeNicolas'? o
rm: ne peut enlever '/tmp/fichierDeNicolas': Operation not permitted
```

C. Gestion des droits

1. chgrp

Le groupe par défaut attribué aux nouveaux fichiers est le groupe principal de l'utilisateur qui l'a créé, à moins que le droit SGID soit positionné sur le répertoire d'accueil.

La commande **chgrp** ("change group") permet de modifier ce groupe ; l'utilisateur peut céder ce fichier à n'importe quel groupe auquel il appartient. La syntaxe de la commande est :

chgrp [-R] <groupe> <fichier ...>

L'option principale **-R** indique à la commande d'appliquer la modification de façon récursive (c'est-à-dire à tous les fichiers et sous-répertoires du répertoire passé en argument).

- À l'inverse de l'administrateur, l'utilisateur ordinaire ne peut modifier que les fichiers dont il est le propriétaire.

Par exemple :

```
[alain]$ id
uid=506(alain) gid=502(vente) groupes=502(vente),503(compta)
[alain]$ touch fichier
[alain]$ ls -l fichier
-rw-r--r--  1 alain vente 0 jun  3 03:58 fichier
[alain]$ chgrp compta fichier
[alain]$ ls -l fichier
-rw-r--r--  1 alain compta 0 jun  3 03:58 fichier
```

- La commande similaire **chown** ("change owner") permet également de modifier le propriétaire d'un fichier mais seul l'administrateur possède les autorisations requises.

2. chmod

La commande **chmod** permet de modifier les droits (ou "mode") des fichiers. Seuls le propriétaire du fichier et l'administrateur système (**root**) peuvent utiliser cette commande.

La syntaxe générale de la commande est la suivante :

```
chmod [-R] <droits> <fichier ...>
```

Comme pour la commande **chgrp**, l'option **-R** indique à la commande d'appliquer la modification de façon récursive.

Il y a deux manières d'indiquer les droits d'accès d'un fichier sous Unix : en notation symbolique et en notation octale.

Notation symbolique

La notation symbolique s'appuie sur les caractères **r** (lecture) **w** (écriture) **x** (exécution) pour désigner les droits et **u** (utilisateur) **g** (groupe) **o** (autres) pour symboliser les entités concernées.

- Dans la notation symbolique, le caractère **a** est équivalent à **ugo** et représente les trois entités.

Un **+** dans cette syntaxe indique qu'il faut positionner le droit, un **-** indique que ce droit doit être supprimé et un **=** permet de définir les trois droits en une seule étape pour l'entité spécifiée.

La syntaxe générale ressemble alors à :

```
<entité(s)>[+-=]<droit(s)>
```

On a par exemple :

u+x	Ajoute (**+**) le droit d'exécution (**x**) pour le propriétaire (**u**).
g-w	Supprime (**-**) le droit de modification (**w**) pour le groupe (**g**).
o-rw	Supprime (**-**) les droits de lecture (**r**) et de modification (**w**) pour les autres (**o**).
ug-x	Supprime (**-**) le droit d'exécution (**x**) pour le propriétaire (**u**) et le groupe (**g**).
a+r	Ajoute (**+**) le droit de lecture (**r**) pour toutes les entités (**a**).
u=rw	Fixe (**=**) les droits de lecture (**r**) et de modification (**w**) et supprime le droit d'exécution (**x**) pour le propriétaire (**u**).

- Ajouter un droit déjà permis – ou en supprimer un non autorisé – ne produit pas d'erreur ; c'est une opération nulle.

Les autres droits sont positionnés avec la lettre **s** et l'entité **u** pour le SUID, la lettre **s** et l'entité **g** pour le SGID, et la lettre **t** et l'entité **o** pour le bit Sticky ; soit :

- **u+s** : ajoute (+) le bit SUID (**s** et **u**).
- **g+s** : ajoute (+) le bit SGID (**s** et **g**).
- **o+t** : ajoute (+) le bit SUID (**t** et **o**).

*Exemples de la commande **chmod** en utilisant la notation symbolique :*

```
[alain]$ ls -l fichier
-rw-r--r--  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod u+x fichier
[alain]$ ls -l fichier
-rwxr--r--  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod go+w fichier
[alain]$ ls -l fichier
-rwxrw-rw-  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod a-r fichier
[alain]$ ls -l fichier
--wx-w--w-  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod o=rx fichier
[alain]$ ls -l fichier
--wx-w-r-x  1 alain compta 0 jun  3 03:58 fichier
```

Notation octale

Cette seconde notation est le codage sur 3 bits des droits de chaque entité.

En positionnant le bit à 1 lorsque le droit correspondant est autorisé, et à 0 lorsqu'il n'est pas permis, cela donne par exemple 101 pour les droits **r-x**.

Un nombre de 3 bits pouvant être codé sur un seul chiffre en base octale, la table de conversion suivante permet de faire le lien entre les notations :

Droits (symbolique)	notation binaire	notation octale
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

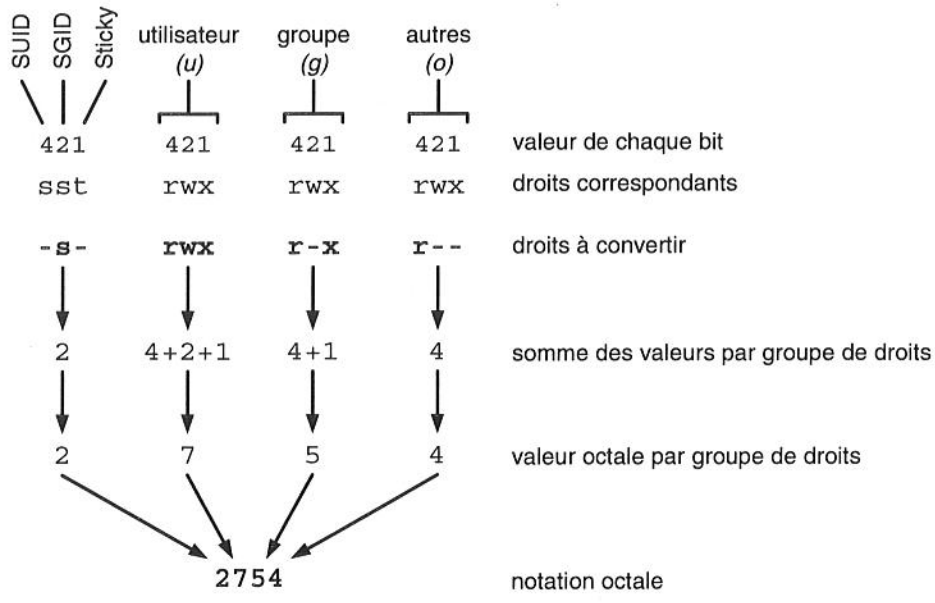
Ceci répété pour les trois entités, les droits **rwxr-xr--** peuvent être traduits en notation octale avec 754.

Pour les droits SUID, SGID et Sticky, on ajoute à ce nombre en base octale un chiffre supplémentaire à gauche, suivant la relation :

Droits (symbolique)	notation binaire	notation octale
---	000	0
--t	001	1
-s-	010	2
-st	011	3
s--	100	4
s-t	101	5
ss-	110	6
sst	111	7

Cela donne donc par exemple 2754 pour **rwxr-sr--**.

Un autre moyen rapide de convertir les droits Unix en notation octale consiste à additionner les valeurs 4, 2 et 1 pour chaque ensemble de trois droits lorsqu'ils sont autorisés :



*Exemples de la commande **chmod** en utilisant la notation octale :*

```
[alain]$ ls -l fichier
--wx-w-r-x  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod 644 fichier
[alain]$ ls -l fichier
```

```
-rw-r--r--  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod 0744 fichier
[alain]$ ls -l fichier
-rwxr--r--  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod 766 fichier
[alain]$ ls -l fichier
-rwxrw-rw-  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod 0322 fichier
[alain]$ ls -l fichier
--wx-w--w-  1 alain compta 0 jun  3 03:58 fichier
[alain]$ chmod 4755 fichier
[alain]$ ls -l fichier
-rwsr-xr-x  1 alain compta 0 jun  3 03:58 fichier
```

Les chiffres manquants de la notation octale (sur 4 chiffres) sont remplacés automatiquement par des zéros à gauche ; ainsi 755 équivaut à 0755.

3. umask

Les droits d'accès par défaut des nouveaux fichiers sont définis par la valeur d'un masque que l'on applique à l'ensemble des droits standards.

La valeur habituelle de ce masque pour les utilisateurs ordinaires est 002 ; cela signifie que l'on soustrait bit à bit les droits du masque `------w-` (en notation symbolique) à l'ensemble des droits fondamentaux **`rwxrwxrwx`** (ou 777 en octal).

Dans le cas de fichiers ordinaires, les droits d'exécution **`x`** ne sont jamais positionnés par défaut et sont donc, eux aussi, soustraits :

	fichier	**répertoire**
ensemble des droits fondamentaux	`rwxrwxrwx`	`rwxrwxrwx`
masque 002	`-------w-`	`-------w-`
droits par défaut des fichiers	`rw-rw-r--` (suppression des droits x)	`rwxrwxr-x`

Le masque de l'administrateur est généralement 022 ; **`root`** fait donc moins confiance à son groupe qu'un utilisateur ordinaire.

La valeur du masque est affichée et définie à l'aide de la commande **`umask`** :

```
[franck]$ umask
0002
[franck]$ umask -S
u=rwx,g=rwx,o=rx
```

```
[franck]$ touch fichier1
[franck]$ mkdir rep1
[franck]$ ls -l
total 4
-rw-rw-r--  1 franck cours    0 jun  3 04:55 fichier1
drwxrwxr-x  2 franck cours 4096 jun  3 04:55 rep1/
[franck]$ umask 0022
[franck]$ touch fichier2
[franck]$ mkdir rep2
[franck]$ ls -l
total 8
-rw-rw-r--  1 franck cours    0 jun  3 04:55 fichier1
-rw-r--r--  1 franck cours    0 jun  3 04:55 fichier2
drwxrwxr-x  2 franck cours 4096 jun  3 04:55 rep1/
drwxr-xr-x  2 franck cours 4096 jun  3 04:55 rep2/
[franck]$ umask 1234
-bash: umask: 1234: octal number out of range
[franck]$ umask 6
[franck]$ umask
0006
```

Pour des raisons de sécurité évidentes, il n'est pas possible de positionner par défaut les droits d'accès étendus (SUID, SGID et sticky bit) lors de la création de fichiers.

- Cette commande est inscrite dans le fichier `/etc/profile` ou `/etc/bashrc` de façon à être exécutée à chaque nouveau shell. L'utilisateur a la possibilité de surcharger la valeur définie par l'administrateur dans son propre fichier de configuration `$HOME/.bash_profile`.

4. Gestionnaires de fichiers

Tout gestionnaire de fichiers sous Linux se doit de présenter les droits d'accès associés aux fichiers et de les modifier.

Une bonne appréhension de l'ensemble de ce chapitre permet de comprendre les options proposées par les interfaces de ces gestionnaires de fichiers.

Avec Konqueror par exemple :