TP n°3: Gestion avancée des fichiers

D. Pellier, D. Janiszek, J. Mauclair

15 août 2018

Sommaire

Objectifs :

- Savoir lire les attributs d'un objet (type, droits, propriétaire, etc.)
- Maîtriser les commandes de gestion des droits d'accès
- Gérer son espace disque
- Compresser/décompresser des fichiers

Éléments techniques abordés :

- *Les commandes*
  - `chmod`, `chown`, `chgrp`
  - `du`, `df`, `quota`
  - `zip`, `gzip`
  - `find`
  - `lpr`, `lpq`, `lpstat`, `lprm`
- *Les répertoires*
  - `/home/`
- *Les fichiers*
  - `/etc/passwd`

1 Gestion des droits d'accès

1.1 Principe

Tout système multi-utilisateur définit une politique de protection des fichiers fondée sur l'identification de l'utilisateur. Cette protection s'applique aux fichiers, aux répertoires mais aussi à l'exécution des logiciels. Chacun dispose d'un espace privé, son répertoire personnel, dont il est propriétaire, et sur lequel il possède le contrôle total. Chaque utilisateur doit protéger cet espace, et adapter les droits en fonction des permissions qu'il souhaite donner aux autres utilisateurs. L'accès au répertoire et aux fichiers des autres utilisateurs dépend des droits que ces derniers leur ont attribués. L'accès à un fichier ou à un répertoire dépend :

— de l'identité de l'utilisateur qui y accède ;
— des droits que le propriétaire leur a attribués.

Ce contrôle est effectif durant toute la session de travail et quel que soit le logiciel applicatif utilisé. Le super-utilisateur, *root*, déroge à cette règle. Il possède tous les droits sur l'ensemble des fichiers et répertoires, sans aucune restriction. Aucun utilisateur ne peut lui interdire l'accès à son espace personnel.

1.2 Notion d'utilisateur et de groupe

Chaque utilisateur est connu du système par un numéro unique, son UID. Ce numéro est attribué par l'administrateur lors de la création du compte. La correspondance entre les noms de login des utilisateurs et leur UID est définie dans le fichier */etc/passwd*. Chaque ligne de ce fichier contient les informations d'un seul utilisateur. La commande suivante affiche son contenu (toutes les lignes ne sont pas présentées - les lignes manquantes sont remplacées par un point de suspension). Sur cet ordinateur, l'utilisateur *dupont* possède l'UID 532.

```
$cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/sh
...
nobody:x:65534:65534:Nobody:/:/bin/sh
dupont:x:532:550:Jacques Dupont,ENSEIGNANT,20050921,:/home/dupont:/bin/bash
martin:x:524:600:Paul Martin,TECHNICIEN,20060119,:/home/martin:/bin/bash
```

Chaque utilisateur appartient à un groupe. Ce groupe principal, ou initial, porte un numéro, le GID. Le fichier */etc/passwd* contient le numéro GID du groupe principal de chaque utilisateur. Dans l'exemple précédent, *dupont* appartient au groupe principal dont le GID est 550. La correspondance entre le GID d'un groupe et son nom se trouve dans le fichier */etc/group*/ . Dans ce fichier sont indiqués, pour chaque groupe, le nom du groupe, le GID et la liste des noms de login des membres du groupe. Un utilisateur appartient à un seul groupe principal, mais il peut être membre d'autres groupes complémentaires. Dans ce cas, son nom de login apparaît dans le fichier */etc/group*, parmi les membres des groupes auxquels il appartient.

A tout moment, l'utilisateur peut connaître son UID, son groupe principal et ses groupes complémentaires en tapant la commande `id`.

N'importe quel utilisateur possède les droits liés à son identité, son UID, et à tous ses groupes d'appartenance.

A noter : Sur tous les systèmes UNIX, l'UID du super-utilisateur root est le numéro 0, et son groupe principal est le GID numéro 0.

1.3 Les attributs d'un fichier

A noter : Les droits sur les fichiers et les répertoires sont identiques. Afin de simplifier la syntaxe, le terme *fichier* sera employé

Chaque fichier possède un ensemble d'attributs, parmi lesquels :

— L'UID de son propriétaire ;
— le GID de son groupe propriétaire
— les droits pour les différentes catégories d'utilisateurs

C'est grâce à ces informations que l'accès au fichier est contrôlé. L'identité et les groupes d'appartenance de l'utilisateur qui demandent l'accès sont comparés au propriétaire, puis au groupe propriétaire du fichier. Les droits d'accès de cet utilisateur sont déterminés en fonction du résultat de cette comparaison. Initialement, le propriétaire est le créateur du fichier, et le groupe propriétaire est le groupe d'appartenance de cet utilisateur au moment de la

création. Le fichier hérite des informations de son créateur. Une fois créé, le fichier est ”autonome” : il conserve les informations indépendamment des actions de son créateur. Il est possible de changer le propriétaire ou le groupe propriétaire, grâce aux commande `chown` et `chgrp`, sans tenir compte de la cohérence de ces informations vis-à-vis du créateur. Ainsi, le groupe propriétaire peut être totalement différent du groupe d'appartenance du propriétaire. La seule contrainte est de posséder les privilèges pour effectuer le changement, ce qui est le cas de root.

Attention Si l'administrateur détruit le compte d'un utilisateur (simplement en le supprimant du fichier */etc/passwd*), les fichiers de ce dernier conservent néanmoins son numéro UID, comme propriétaire. L'administrateur de doit donc jamais réutiliser un UID qui a déja été affecté car il serait propriétaire de ces fichiers.

1.4 Lire les droits

Chaque fichier a plusieurs propriétés associées : le propriétaire, le groupe propriétaire, la date de dernière modification, et les droits d'accès. On peut examiner ces propriétés grâce à l'option `-l` de `ls`. Dans cet exemple, nous voyons les permissions standard d'un répertoire et de deux fichiers :

```
$ls -l
total 205
drwxr-xr-x    2 toto phy03       512 Jan 16 10:02 fiches/
-rw-r--r--    1 toto phy03     72008 Oct  2  2003 article.dvi
-rw-r--r--    1 toto phy03    145905 Oct  2  2003 article.pdf
```

Permissions

Les permissions sont indiquées dans la colonne de gauche, suivant un format bien particulier :

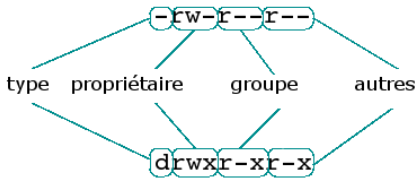

FIGURE 1 – Format des droits d'accès

Comme nous le voyons sur l'image ci-dessus, le bloc de permissions se divise en quatre éléments.

1.Type : Le premier caractère du bloc de permissions indique le type du fichier : - pour un fichier normal, d pour un répertoire. On trouve également parfois l pour les liens symboliques, et d'autres choses plus exotiques.

2.Droits du propriétaire :

— r ou - : droit de lire (r pour read) le fichier (r pour oui, - pour non)

— w ou - : droit d'écrire (w pour write) dans le fichier

— x ou - : droit d'exécuter (x pour execute) le fichier.

Pour un répertoire, les choses sont un peu différentes. Le couple rx donne le droit d'examiner le répertoire et son contenu. Le bit w donne le droit d'ajouter ou de supprimer des fichiers dans le répertoire.

3.Droits du groupe : Comme les droits du propriétaire, mais s'applique aux gens qui sont dans le groupe propriétaire (voir ci-dessous sur les groupes).

4.Droits des autres : Comme les droits du propriétaire, mais s'applique aux gens qui sont ni le propriétaire, ni dans le groupe propriétaire.

Liens

Nombre de liens du fichier ; un répertoire en a au moins deux (. et ..). Un répertoire qui contient 5 sous-répertoires en a 7, etc.

Propriétaire

Le nom de login de la personne à qui appartient ce fichier. Seul le propriétaire peut changer les droits ou le groupe d'un fichier.

Groupe propriétaire

Les groupes sont des ensembles d'utilisateurs qui sont fixés par l'administrateur du système. Ils sont un moyen pour gérer un peu finement les droits d'accès. C'est la commande id qui vous révèle le(s) groupe(s) auquel vous appartenez :

```
$ id
uid=4242(toto) gid=276(phy03) groups=276(phy03)
```

La commande vous indique d'abord votre identifiant d'utilisateur (UID) et votre login, puis votre identifiant de groupe principal (GID) et enfin tous les groupes dans lesquels vous vous trouvez.

Taille

Elle est indiquée en octets.

A retenir

Il y a donc 3 catégories d'utilisateurs pour définir les droits sur un fichier :
— le propriétaire, ou *user* ;
— les membres du groupe propriétaire, ou *group* ;
— les autres, qui ne sont ni propriétaires ni membres du groupe, autrement dit la catégorie *other*

1.5 Les droits d'accès en détail

Modèle concentrique des droits d'accès

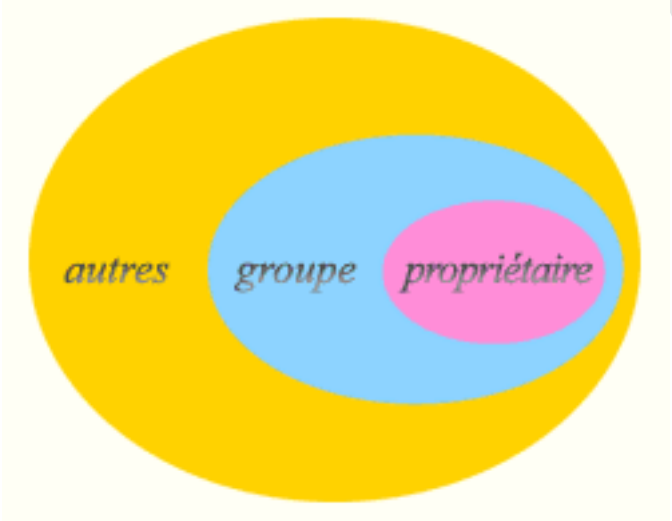

FIGURE 2 – Modèle concentrique des droits d'accès

Le schéma ci-dessus montre qu'un ensemble de propriétaires forme un groupe, qu'un ensemble de groupes forme la catégorie "autres" (qui sont tous ceux qui prétendent à accéder aux données). L'accès à un sous ensemble concentrique suppose a priori d'obtenir des droits supplémentaires.

Les droits Pour chaque catégorie, il est possible d'attribuer les droits suivants :
— lecture ou *read* représenté par la lettre r
— écriture ou *write* (w) représenté par la lettre w
— exécution ou *execute* (x) représenté par la lettre x

L'absence de droit est désigné par le caractère -. Il remplace r,w ou x.

A chaque catégorie d'utilisateur on associe un triplet de droits : lecture, écriture et exécution. Au total 9 droits (3*3) sont affectés à chaque fichier. Lorsqu'un droit est alloué, on voit la lettre correspondante (r, w ou x). Si

le droit est refusé, on voit un tiret (-). Dans l'exemple ci-dessous, le propriétaire dispose des droits de lecture et d'écriture. Tandit que le groupe ainsi que les autres ne disposent que du droit de lecture.

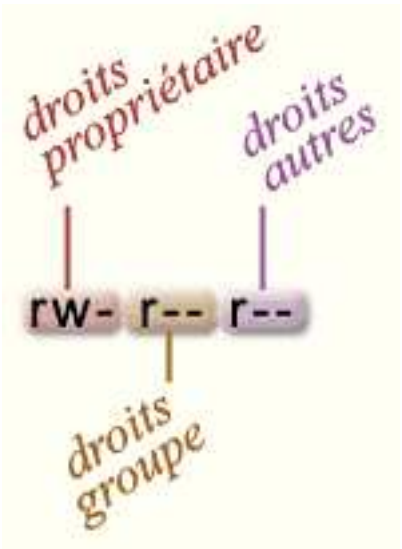

FIGURE 3 – Les droits

Les droits sont données dans l'ordre rwx pour chaque catégorie (user,group,other). Ainsi,les droits d'un fichier pour lequel le propriétaire possède les permissions en lecture, écriture et exécution, le groupe propriétaire les permissions en lecture et exécution, les autres aucune permission, seront notés : `rwxr-x`.

Combinaisons des droits

A chacune des 3 catégories d'utilisateur, on associe d'une des 8 combinaisons différentes possibles pour l'allocation des droits que le tableau ci-dessous récapitule.

Triplet	Droits correspondants
- - -	aucun
- - x	exécution
- w -	écriture
- w x	écriture et exécution
r - -	lecture
r - x	lecture et exécution
r w -	lecture et écriture
r w x	lecture, écriture et exécution

Or les droits globaux d'un fichier sont identifiés par l'association de 3 triplets de droits. Ce qui nous fait 8^3=512 combinaisons différentes. Le tableau suivant regroupe quelques unes de ces combinaisons possibles.

Droits globaux	Description
rwxr - x r - x	propriétaire : tous les droits, groupe et autres : pas accès en écriture.
rwxr - - r - -	propriétaire : tous les droits, groupe et autres : accès qu'en lecture.
rwxr - x - - -	propriétaire : tous les droits, groupe : lecture et exécution, autres : aucun droit.
rwx - - - - - -	propriétaire : tous les droits, groupe et autres : aucun droit.
rw-r - - r - -	propriétaire : droits de lecture et écriture, groupe et autres : droit en lecture.
rw-rw - - - -	propriétaire et groupe : lecture et écriture, autres : aucun droit.

Commande d'affichage des droits La commande d'affichage des droits pour un fichier est `ls -l`. Sans plus de détails, cette commande liste les informations sur les fichiers et répertoires se trouvant à l'endroit du système du fichier où se situe l'utilisateur. Appliquée à un répertoire, cette commande va lister les informations de tous les fichiers et sous-répertoires. La commande `ls -ld` appliquée à un répertoire permet de connaître les informations de ce répertoire.

1.6 La modification : la commande chmod

La commande qui modifie les droits est chmod (change file modes). Sa syntaxe générale est :

```
$chmod -options modes fichier
```

où :

- `-options` représente les options de la commande, dont la principale est -R. Elle permet d'appliquer récursivement les modifications au répertoire et à son contenu.
- `modes` représente les modifications apportées aux droits. Elles sont indiquées soit en notation symbolique, soit en notation octale. Dans ces 2 représentations, l'utilisateur précise quels sont les droits modifiés et pour qui s'appliquent les changement.
- `fichier` est le nom du fichier ou du répertoire dont les droits sont à modifier.

Voici quelques exemples de syntaxe : La commande suivante enlève le droit `r` au groupe, pour tous les fichiers commencant par essai :

```
$chmod g-r essai*
```

La commande suivante enlève le droit `x` au groupe, pour le répertoire Exercice et tout son contenu. Cette commande est exécutée récursivement sur les sous-répertoires

```
$chmod -R g-x Exercice
```

La commande suivante ajoute le droit `w` au propriétaire, pour le fichier essai.txt

```
$chmod u+w essai.txt
```

Attention Les droits affectés aux répertoires pères prévalent sur ceux qui sont appliqués à l'intérieur. Vous pouvez donner tous les droits à vos fichiers, mais si le répertoire parent de permet pas d'y entrer, aucune action ne pourra être réalisée

La notation symbolique

En notation symbolique, l'argument `modes` peut être par exemple `u+w`. Il est composé de trois parties :

1. La `catégorie` d'utilisateur à laquelle s'appliquent les modifications. Elle est spécifiée par une lettre ou groupe de lettres : `u` quand la modification s'applique au propriétaire du fichier (user), `g` quand elle s'applique au groupe propriétaire du fichier (group), `o` quand elle s'applique aux autres (other) et `a` quand elle s'applique à tous (all). De même, `ug` signifie que les droits sont changés pour user et group, `uo` pour user et other, `go` pour group et other . `ugo` est équivalent à `a`. Le tableau ci-dessus récapitule les différentes catégories :

Catégorie	Description
u	propriétaire
g	groupe
o	autres
a	tous

2. Le `type` de modification est représenté par : `+` quand il faut ajouter les droits, `-` quand il faut les retirer, et `=` quand il faut installer exactement et seulement ces droits. Les opérateurs `+` et `-` modifient les droits précisés dans modes et laissent les autres inchangés. `=` retire tous les droits pour ne laisser que ceux indiqués. Le tableau ci-dessous récapitule les différents types de modification :

Opération	Description
+	ajouter
-	retirer
=	définir

3. Les `droits` à modifier. Ils sont définis par la lettre ou groupe de lettre : `r` (read) qui porte sur le droit en lecture, `w` (write) qui porte sur le droit en écriture, `x` (execute) qui porte sur le droit d'exécution. De même, `rw` porte sur `read` et `write`, `rx` sur `read` et `execute`, `wx` sur `write` et `execute` et `rwx` pour `read write` et `execute`.

Droit	Description
r	lecture
w	écriture
x	exécution

Exemples Voici quelques exemples d'utilisation de la commande `chmod` en notation symbolique :

Exemple de l'utilisation de chmod	Description
chmod g=rwx temps.txt	Alloue au groupe tous les droits.
chmod g-w temps.txt	Retire au groupe le droit d'écriture.
chmod o-x temps.txt	Retire aux autres le droit en exécution
chmod a-rwx temps.txt	Retire à tous tous les droits.
chmod u=rw temps.txt	Alloue au propriétaire les droits en lecture et en écriture.
chmod a+r temps.txt	Rajoute à tous le droit en lecture.

La notation octale

La commande `chmod` permet aussi, à l'aide d'un format numérique, de redéfinir les droits d'accès. Mais avant d'aborder cette notation, vous devez maîtriser les systèmes de numération. Vous connaissez déjà le système décimal, ou base 10, que vous utilisez couramment. Dans ce système, chaque chiffre, de droite à gauche, a pour valeur ce chiffre multiplié par une puissance de 10, selon sa position dans le nombre. La figure ci-dessous dissèque ainsi le nombre 5783 en base 10.

Dans un système de numération, la valeur d'un nombre est la somme de la valeur de chaque chiffre qui le compose multiplié par la base, élevée à la puissance n, où n est la position du chiffre par rapport à la droite. Ainsi, dans 5783, le 7 se trouve à deux positions par rapport au chiffre le plus à droite. Sa valeur est donc égale à l base (ici, 10) élevée à la puissance 2 (sa position), c'est à dire 100, multipliée par 7. Sans surprise, le chiffre 7 dans le nombre 5783 vaut donc 700.

La base octale est la base 8. Elles contient les chiffres de 0 à 7. Elle permet le codage numérique des droits d'accès. Chaque chiffre de cette base peut s'écrire en base binaire (la base 2), comme un nombre de trois bits. Le tableau suivant donne pour chaque chiffre de la base octale, sa signification en binaire :

valeur octale	valeur binaire
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Quel est le rapport avec la commande `chmod` ? Les droits d'accès sous Unix représentant un ensemble d'interrupteurs oui/non. Le groupe a t'il l'accès en écriture ? 1 signifie oui, 0 signifie non. Dans le système binaire, chaque chiffre ne peut prendre que deux valeurs : oui ou non, 1 ou 0, autorisé ou bloqué. Il est donc possible de décrire facilement et sans ambigüité un bloc de permissions comme une suite de zéros et de uns, c'est à dire, comme un chiffre binaire.

La figure suivante illustre ce propos :

Dans cette notation, chaque chiffre binaire d'un nombre de trois bits représente donc respectivement les droits r, w, x. La valeur 0 indique l'absence de droits, la valeur 1 sa présence.

Le tableau suivant présente la correspondance entre un chiffre octal et sa valeur en binaire, puis sa signification en termes de droit. Ainsi, le chiffre 7 correspond à `rwx` et le chiffre 4 à `r`.

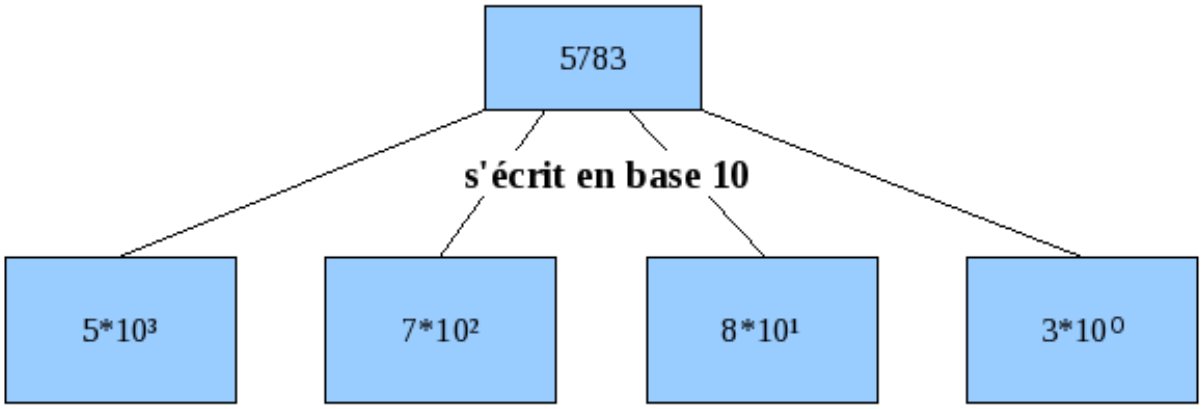

FIGURE 4 – Interprétation de 5793 en base 10

valeur octale	valeur binaire	droits
0	0 0 0	- - -
1	0 0 1	- - x
2	0 1 0	- w -
3	0 1 1	- w x
4	1 0 0	r - -
5	1 0 1	r - x
6	1 1 0	r w -
7	1 1 1	r w x

Nous savons donc passer d'un nombre octal à un nombre binaire. Il existe également un moyen de passer d'un nombre binaire à un nombre octal. La figure suivante illustre comment :

On peut donc exprimer n'importe quelle combinaison de droits d'accès pour un fichier ou un répertoire grâce à trois chiffres, un pour le propriétaire, un pour le groupe et un pour les autres. La notation octale utilise donc un nombre de trois chiffres octaux, un par catégorie d'utilisateurs. Les droits **r w - r - x r - -** correspondent ainsi à "`654`" en base octale.

Exemples Voici quelques exemples d'utilisation de la commande `chmod` en notation octale :

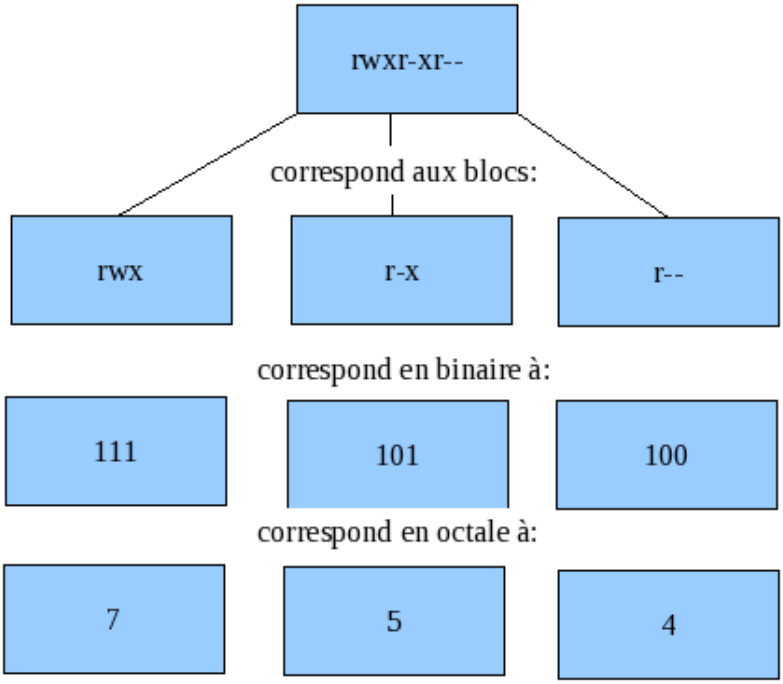

FIGURE 5 – Représentation des droits d'accès en base octale

Exemple en notation octale	Droits globaux associés
chmod 640 temps.txt	r w - r - - - - -
chmod 700 temps.txt	r w x - - - - - -
chmod 664 temps.txt	r w - r w - r - -
chmod 764 temps.txt	r w x r w - r - -
chmod 610 temps.txt	r w - - - x - - -

1.7 Le changement de propriétaire : la commande chown

La commande *chown* (CHange OWNer) permet (entre autres) de modifier le propriétaire d'un fichier : elle donne un fichier à quelqu'un d'autre. Toutefois, seul l'utilisateur `root` peut l'utiliser, les simples utilisateurs n'ont aucun autre moyen de se débarrasser d'un fichier que de l'effacer. Sa syntaxe est :

```
$ chown -options nouveau_proprietaire:nouveau_groupe fichier
```

où

- `options` représente les options de la commande, dont la principale est -R. Elle applique récursivement les modifications au répertoire et à son contenu.
- `nouveau_propriétaire` est le nom ou l'UID du nouveau propriétaire
- `nouveau_groupe` est le nom ou GID du ouveau groupe propriétaire. Il est optionnel et s'il est présent, il est séparé du nouveau propriétaire par le signe " :".
- `fichier` est le nom du fichier ou du répertoire à modifier.

```
$ chown martin essai
```

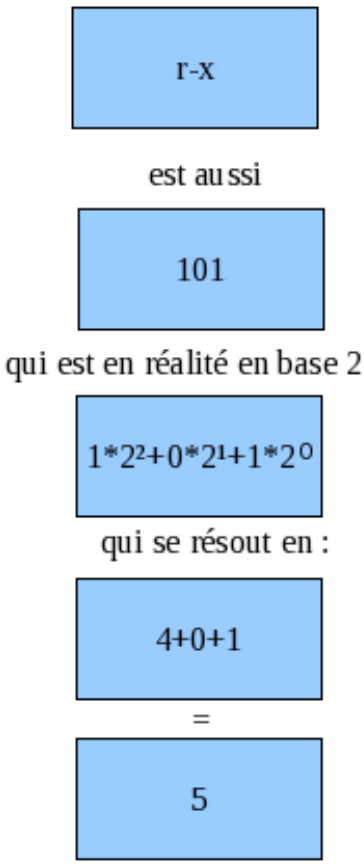

FIGURE 6 – Passage des droits d'accès à la base octale

change le propriétaire du fichier essai par le propriétaire martin

1.8 Le changement de groupe : la commande chgrp

La commande chgrp (change group) change le groupe propriétaire d'un fichier. L'utilisateur root peut affecter n'importe quel groupe. Le propriétaire actuel peut aussi changer de groupe, mais uniquement parmi ceux dont il est membre. La syntaxe est :

```
$ chgrp -options nouveau_groupe fichier
```

où

- `options` représente les options de la commande, dont la principale est -R. Elle applique récursivement les modifications au répertoire et à son contenu.
- `nouveau_groupe` est le nom ou GID du nouveau groupe propriétaire.
- `fichier` est le nom du fichier ou du répertoire à modifier.

1.9 Exercices 3.1 (Les droits)

1. Donnez la représentation binaire du chiffre 7.
2. Donnez la représentation binaire du chiffre 4.
3. Que représente 101 en octal ?
4. Que représente 010 en octal ?
5. Donnez la représentation des droits rwx en octal et en binaire (sur 3 bits).
6. Donnez la représentation binaire du nombre 754 écrit en octal.
7. Que représente 754 en termes de droits d'accès ?
8. Donnez la représentation binaire du nombre 640 écrit en octal.
9. Que représente 640 en termes de droits d'accès ?
10. Donnez la représentation des droits r - x r w - r - - en octal.
11. Donnez la représentation des droits r - x r w - r - - en binaire.
12. Que signifient les droits r w - r - x r - - pour le propriétaire du répertoire ou du fichier concerné ?
13. Que signifient les droits r w x r - - - - - pour le groupe propriétaire du répertoire ou du fichier concerné ?
14. Affichez le nom du propriétaire de votre répertoire de travail.
15. Utilisez la commande `ls -l` afin d'obtenir le contenu de quelques répertoires. Regroupez les informations relatives aux droits en trois groupes distincts, et calculez-en l'équivalent numérique.
16. Vous est-il possible de faire une copie du fichier /etc/passwd ? Vous est-il possible de supprimer ou de modifier ce fichier ? Expliquer la situation à l'aide de la commande *ls -l*
17. Changez les droits de votre répertoire personnel pour être le seul à posséder tous les droits.
18. Dans votre répertoire personnel, créez un fichier vide essai en tapant : `cp /dev/null essai`.
19. Faîtes en sorte que votre groupe puisse lire dans ce fichier.
20. Faîtes en sorte que votre groupe puisse écrire dans ce fichier.
21. Retirez tous les droits sur ce fichier à votre groupe
22. Autorisez tout le monde et votre groupe à l'accès sur ce fichier en lecture.
23. Créez un fichier vide essai2 dans un répertoire Rep2. En notation octale, attribuez-vous les droits de lecture, écriture, affectez ceux de lecture au groupe et n'attribuez aucun droit aux autres.
24. Modifiez les droits du répertoire Rep2 et de son contenu afin que vous disposiez des droits de lecture, d'écriture et d'exécution, que le groupe bénéficie de la lecture et de l'exécution, et les autres de l'exécution.
25. Vérifiez la modification
26. Dans votre répertoire courant, créez un répertoire courant essai_droit, par défaut ce répertoire est à 755 (rwxr-xr-x), quelles sont les commandes (en notation symbolique et en base 8) pour lui donner les droits suivants (on suppose qu'après chaque commande on remet le répertoire à 755 :

Commande	**propriétaire**			**groupe**			**les autres**		
	lecture	écriture	accès	lecture	écriture	accès	lecture	écriture	accès
1	oui	oui	oui	oui	non	oui	non	non	oui
2	oui	non	oui	non	oui	non	non	non	oui
3	non	oui	non	non	non	oui	oui	non	non
4	non	non	oui	oui	non	oui	non	non	non

2 Gestion de l'espace disque

Le répertoire dans lequel vous vous trouvez après connexion s'appelle le répertoire d'accueil ($HOME). Ce répertoire est dans un *file system* dont la capacité en octets est ficée. La création de fichiers va au fur et à mesure remplir le *file system* jusqu'à ce que le message **file system full** apparaisse. Il est alors impossible de continuer à travailler, et il faudra faire de la place. Il est donc nécessaire de gérer l'espace disque, en veillant à éliminer les fichiers devenus inutiles (*core*, fichiers temporaires,...) et en archivant les fichiers qui ne sont plus utilisés afin de pouvoir les supprimer, ou tout au moins de les conserver sous la forme d'une archive compressée.

2.1 Contrôler la consommation disque avec la commande du

La commande `du` affiche la quantité d'espace disque occupé par chacun de ses arguments. Elle affiche par défaut la taille en blocs de 512 octets. avec l'option -k, la taille est affichée en kilo-octets et l'option -s, affiche la somme pour chaque argument. Si dans un répertoire donné, on saisit la commande `du -sk *` comme dans l'exemple suivant : Exemple :

```
$du -sk *
0       fic_ref
12      prog
4       prog.c
```

on obtient l'affichage en kilo-octets de la taille totale de tous les fichiers et répertoires situés dans ce répertoire.

Dans l'exemple

```
$du -sk .
20      .
```

on obtient l'affichage en kilo-octets de la taille du répertoire courant, et ce en incluant tous les répertoires et fichiers de la sous-arborescence.

2.2 Vérifier l'espace disponible avec la commande df

Déterminer combien d'espace disque reste disponible est possible grâce à la commande `df`. Cependant, elle ne donne pas de résumé et pour connaître la taille disponible de notre espace, il faut souvent additionner les résultats obtenus.

1. Voici un exemple de résultat du système lorque nous appelons `df` sans argument :

Sys. de fich.	1K–blocs	Occupe	Disponible	Capacite	Monte sur
/dev/zd0a	17259	14514	1019	93%	/
/dev/zd8d	185379	143995	22846	86%	/userf
/dev/zd7f	185379	12984	153857	8%	/tmp
/dev/zd3f	385689	307148	39971	88%	/users
/dev/zd5c	371507	314532	19824	94%	/usr
/dev/zd0g	254987	36844	192644	16%	/var

On obtient beaucoup d'informations, mais il n'est pas très facile d'additionner rapidement ces valeurs pour obtenir l'espace total disponible. Ces données conservent toutefois leur intérêt. La première colonne concerne le nom du périphérique, la 2ème la taille totale du périphérique en Kilo-octets, la 3ème, les kilo-octets utilisés, la 4ème, les kilo-octets disponibles, la 5ème le pourcentage utilisé et enfin, la 6ème colonne indique le nom du disque.

2. Puisque vous savez que votre répertoire de connexion se trouve sur le disque `/users`, il vous suffit de chercher ce répertoire dans la colonne de droite pour apprendre que vous utilisez dans cet exemple le disque `/dev/zd3f`. Vous apprenez aussi qu'il s'agit d'un disque de 385 689 kilo-octets, occupé à 88%, ce qui signifie que 307 148 Ko sont occupés et qu'il reste 39 971 Ko (environ 38 Mo).

3. Vous pouvez additionner les colonnes afin de constater que le système dispose de 1260Mo d'espace disque dont 830Mo sont occupés. Il reste dont 430 Mo de libre, soit environ 34% de l'espace total.

4. De nombreux systèmes Unix disposent désormais d'une option `-h` permettant d'obtenir un résultat plus lisible. En voici un exemple sur une autre machine :

```
$ df -h
Sys. de fich.          Tail.   Occ.   Disp.   %Occ.   Monte sur
/dev/sd0a              15G     2.1G    13G      14%   /
/dev/sd1a              15G     157M    15G       1%   /web
```

2.3 Les quotas de disque

La gestion des quotas n'est pas toujours disponible sur tous les systèmes UNIX. Les quotas facilitent la gestion de l'espace disque en définissant des limites pour un utilisateur ou un groupe d'utilisateurs. Le système de quota propose deux limites :

- la limite physique ou "hard" : l'utilisateur ne pourra pas la dépasser. Chaque tentative d'écriture sur disque au-delà de cette limite provoquera un échec de la commande.
- la limite logique ou "soft" : l'utilisateur peut la dépasser, mais il dispose d'un délai de grâce (généralement 7 jours) pour détruire des fichiers et redescendre en dessous de ce seuil. Passé ce délai, la seule action autorisée sera la destruction de fichiers pour libérer l'espace disque.

Pour un utilisateur qui n'a aucune restriction la saisie de la commande `quota` provoquera :

```
$quota
Disk quotas for user dupont (uid 532): aucun
```

S'il existe des restrictions, elle provoquera :

```
$quota
Disk quotas for user martin (uid 554):
File system      blocks      quota      limit      grace      files      quota      limit
        grace
/home/etud01       1324      19500      20000                    40          0          0
```

où

- `blocks` est la taille en kilo-octets de l'espace occupé. Cet utilisateur occupe 1324 Ko d'espace disque.
- `quota` est la limite logique ("soft") que l'utilisateur peut dépasser temporairement. Cet utilisateur dispose de 19500Ko.
- `limit` est la limite physique ("hard") que l'utilisateur ne peut pas dépasser. Cette limite est ici de 20000Ko ou 20 Mo.
- `grace` est le nombre de jours autorisés pour redescendre en dessous de la limite logique.
- `files` est le nombre de fichiers appartenant à cet utilisateur.
- `quota, limit, grace` sont les limites concernant le nombre de fichiers.

```
$quota -s
Quotas disque pour user if07545 (uid 8844) :
Systeme de fichiers           space   quota   limite   sursisfichiers   quota   limite   sursis
/users/licence/if07545         28K     196M     293M                        8         0       0
```

Avec le paramètre -s les quota sont exprimés en précisant si les valeurs sont exprimées en kilo-octets K, en méga-octets M ou en giga-octets G.

2.4 Réduire les gros fichiers avec la compression

Maintenant que vous savez déterminer l'espace que vous consommez, vous êtes prêt à apprendre à économiser de la place sans effacer de fichiers. Unix offre un programme standard : `zip` qui permet de compresser des fichiers sans perte d'information.

1. Dans cet exemple simple, `zip` reçoit une liste de fichiers et compresse chacun d'entre eux, et les stocke dans une archive .zip.

```
$ ls -l LISTS
-rw-------   1 martin    106020 Oct 10 13:47 LISTS
$ zip archive.zip LISTS
$ ls -l
-rw-------   1 martin     44103 Oct 10 13:47 LISTS.Z
-rw-------   1 martin     16150 Oct 10 13:47 archive.zip
```

La compression du fichier l'a fait passer de 44Ko à quelques 16Ko (un gain d'environ 60% de l'espace disque). Si vous avez de gros fichiers que vous n'utilisez pas souvent, `zip` permet d'économiser beaucoup d'espace. Le gain de place lié à la compression d'un fichier dépend du type d'informations contenues dans le fichier ; le gain sera important dans le cas d'un fichier texte tandis qu'il sera quasi-nul dans le cas d'un fichier musique au format mp3.

2. `unzip` effectue l'opération inverse. Vous devez indiquer le nom exact du fichier à décompresser (avec l'extension .zip).

```
$ unzip archive.zip
$ ls -l
-rw-------   1 martin   106020 Oct 10 13:47 LISTS
-rw-------   1 martin    16150 Oct 10 13:47 archive.zip
```

Le programme `gzip` permet également de compresser/décompresser un fichier/répertoire. Plus performant que le précédent, il compresse un ou plusieurs fichier(s) en lui rajoutant l'extension par défaut .gz. Sa syntaxe est :

```
$ gzip fichier
```

Le programme `gunzip` ou l'option -d de `gzip` permettent de décompresser des fichiers en .gz.

2.5 Exercices 3.2 (Espace disque)

1. Affichez la taille de tous les fichiers et répertoires se trouvant dans votre répertoire de travail
2. Affichez la taille de votre répertoire public_html/
3. Dans ce répertoire, affichez la taille de tous les fichiers
4. Dans ce répertoire, affichez la taille de tous les fichiers se terminant par `.php`
5. Dans ce répertoire, affichez la taille de tous les fichiers se terminant par `.html`
6. Testez la commande quota pour vérifier vos limites autorisées
7. Exécutez `compress` sur différents fichiers de votre répertoire en veillant à ne pas compresser ceux qui seraient nécessaires à l'exécution de vos programmes (notamment les préférences ou les fichiers cachés)
8. Dans le manuel de la commande `compress` trouvez l'option qui permet d'afficher des informations relatives aux taux de compression obtenus par `compress`

2.6 La commande : find

La commande `find` permet de rechercher des objets (fichiers , répertoires, liens, etc.) dans l'arborescence en fonction de critères définis par l'utilisateur. Le point de départ de la recherche est un répertoire donné en argument. Il peut être indiqué par un chemin relatif ou absolu. Autrement dit, la commande `find` effectue une recherche parmi tous les fichiers contenus dans le répertoire indiqué ainsi que dans tous ses sous-répertoires. Si le répertoire indiqué est la racine du système de fichiers (/) alors la recherche est effectuée parmi tous les fichiers de l'ordinateur.

Pour lancer une recherche, on utilise la syntaxe suivante :

```
$ find repertoire -option1 -option2 -option3 (...)
```

La commande `find` est un outil de base des mécanismes usuels d'administration, comme le nettoyage de l'espace disque ou la sauvegarde. Elle est utilisée soit de manière interactive, en mode ligne de commande, soit de manière différée dans des scripts shell. Elle possède de nombreuses options qui sont autant de critères de recherche possible.

— Les options de la commande `find`

Option	Rôle
`-exec`	Exécuter une commande sur chaque élément trouvé
`-mtime -atime -ctime`	Effectuer une recherche à partir de la date ; respectivement : de modification, du dernier accès ou de création du fichier
`-name`	Effectuer une recherche à partir du nom de l'objet
`-newer`	Effectuer une recherche sur les objets plus récents que l'objet de référence
`-nouser -nogroup`	Effectuer une recherche d'objets n'ayant plus, respectivement : de propriétaire ou de groupe
`-perm`	Effectuer une recherche à partir des permissions de l'objet
`-print`	**Afficher le résultat de la recherche**
`-prune`	Empêcher que la recherche se poursuive dans les sous-répertoires
`-size`	Effectuer une recherche en fonction de la taille de l'objet
`-uid -gid`	Effectuer une recherche à partir du numéro, respectivement : de propriétaire ou de groupe
`-user -group`	Effectuer une recherche à partir du nom, respectivement : de propriétaire ou de groupe

— Les opérateurs logique de la commande `find`

Option	Opérateur	Rôle
-a	ET	L'ensemble des critères doit être vérifié
-o	OU	Un des critères doit être vérifié
!	NON	Inverse le résultat critère à vérifier
{\}(*expression*{\})	(*expression*)	Forcer l'ordre d'évaluation de l'expression

> Attention ! si vous omettez l'option `-print` la commande sera exécutée mais elle n'affichera aucun résultat.

Exemples

Pour *afficher* l'ensemble des fichiers dont la *taille est supérieure à 10 kilo-octets* **et** qui appartiennent à l'*utilisateur dupont* dans l'arborescence */tmp* (c'est à dire : tous les fichiers contenus dans le répertoire tmp ainsi que dans tous ses sous-répertoires) :

```
$ find /tmp -size +10k -a -user dupont -print
```

Pour *afficher* l'ensemble des fichiers appartenant à l'*utilisateur dupont* **ou** à l'*utilisateur martin* dans *toute l'arborescence* (c'est à dire : tous les fichiers de l'ordinateur) :

```
$ find / -user dupont -o -user martin -print
```

Pour *afficher* l'ensemble des fichiers **ne** commençant **pas** par *to* dans l'*arborescence /tmp* :

```
$ find /tmp ! -name "to*" -print
```

Pour *afficher* l'ensemble des fichiers qui sont de *taille nulle* **et** qui ont été *modifiés aujourd'hui* dans *toute l'arborescence* (les parenthèses forcent l'ordre d'évaluation) :

```
$ find / \( -size 0 -a -mtime 0 \) -print
```

L'exécution d'une commande externe

La commande `find` permet d'effectuer une recherche. Elle nécessite aussi que l'on précise ce que l'on souhaite faire du résultat de la recherche. L'action la plus élémentaire consiste à afficher le résultat de la recherche en utilisant l'option `-print`. Toutefois, on peut appliquer d'autres traitements à chaque élément obtenu au cours de la recherche en utilisant l'option `-exec` suivi d'un commande

Pour *effacer* l'ensemble des fichiers qui sont de *taille nulle* **et** qui n'ont pas été *modifiés au cours de la dernière semaine* dans *toute l'arborescence* :

```
$ find / \( -size 0 -a -mtime +7 \) -exec rm {} \;
```

Explication : pour chaque objet trouvé prend place au niveau des accolades et la commande `rm` est lancée.

Une fois que l'on a trouvé des fichiers satisfaisant à un certain nombre de critères, il peut être utile de déterminer s'il existe une différence entre eux. La commande `diff` permet de faire la différence entre deux fichiers. En retour elle indique les lignes où se trouvent les éventuelles différences.

```
$ diff fichier1 fichier2
```

On peut aussi souhaiter trier le résultat ; pour cela il faut utiliser la commande `sort`

> Une autre méthode pour chercher des fichiers consiste à utiliser les commandes **locate** et **updatedb**

2.7 Exercice 3.3

1. Cherchez dans l'arborescence de votre répertoire personnel tous les fichiers dont le nom se termine par .ini, afin d'améliorer la lisibilité du résultat, redirigez les erreurs vers /dev/null
2. Cherchez dans toute l'arborescence de la machine tous les fichiers dont le nom commencent par X ou x, afin d'améliorer la lisibilité du résultat, redirigez les erreurs vers /dev/null
3. Cherchez dans l'arborescence /etc de la machine tous les fichiers et répertoires dont le nom commencent par X ou x, afin d'améliorer la lisibilité du résultat, redirigez les erreurs vers /dev/null
4. Cherchez dans /usr les fichiers dont la taille dépasse 1Mo et dont les droits sont fixés à 755 (-rwxr-xr-x)
5. Cherchez le nombre de fichiers dans votre répertoire personnel vous appartenant et ayant les droits fixés à 666 (-rw-rw-rw-)

3 Impression de fichier

Il existe plusieurs systèmes d'impression, et chacun possède ses propres commandes. Le plus répandu est celui du système BSD, qui utilise le démon (processus particulier toujours actif) `lpd`. Dans ce système, la variable PRINTER indique l'imprimante par défaut, si elle est positionnée. Les commandes de gestion de l'impression sont les suivantes :

— **lpr** Envoie la requête d'impression. Sa syntaxe générale est :

```
$lpr -P imprimante fic
```

où `imprimante` est le nom de l'imprimante et `fic` est le fichier à imprimer.

— **lpq** ou **lpstat** Ces commandes affichent l'état général des impressions

— **lprm** Détruit une impression

```
$lprm -P imprimante JobID
```

ou

```
$lprm JobID
```

où `JobID` est le numéro d'impression dans la file d'attente