Module administration système et réseau GNU/Linux

Gestion des systèmes de fichiers

REFERENTIEL EMPLOI ACTIVITES COMPETENCES DU TITRE PROFESSIONNEL TSGERI	
Activité type	Maintenir, exploiter et sécuriser une infrastructure centralisée
Compétences professionnelles	Maintenir et exploiter un serveur Linux

Sommaire

A. FHS en détail

Organisation standard du système de fichiers

Répertoire	Observation	Abréviation
/	Contient les répertoires spéciaux	
/boot	Fichiers relatifs au démarrage du système	
/sbin	Commandes indispensables au démarrage système	system binaries
/bin	Exécutables des commandes de base du système	binaries
/usr/bin	Commandes d'administration système	
/lib	Librairies partagées et modules du noyau	libraries
/usr	Tout ce qui n'est pas nécessaire au fonctionnement minimal du système	UNIX System Resources
/mnt	Pour le montage de SF temporaires	mount
/media	Pour le montage de médias amovibles	
/root	Répertoire de connexion de l'administrateur	
/home	Données utilisateurs	
/tmp	Fichiers temporaires	temporary
/dev	Fichiers spéciaux des périphériques	device
/etc	Fichiers de configuration et de scripts	editable text configuration
/opt	Spécifiques aux applications installées	optional
/proc	Système de fichiers virtuel représentant les différents processus	processes
/var	Fichiers variables divers	variables

B. Les différents types de fichiers

Sous Linux, il existe 7 types de fichiers, regroupés en 3 catégories : les fichiers ordinaires, les répertoires et les fichiers spéciaux.

- **Les fichiers ordinaires** (ou fichiers réguliers) sont destinés à contenir du texte, le code exécutable d'un programme, ou tout type d'information (données binaires, c'est à dire non texte). Linux ne fait pas de supposition sur leur contenu et ne tient pas compte de leur extension. Cela ne veut pas dire que l'extension d'un fichier n'a pas d'intérêt. Elle permet aux utilisateurs de rapidement connaître le contenu d'un fichier. Elle est aussi utilisée par les gestionnaires de fichiers graphiques de type Nautilus lorsqu'on double-clique sur un fichier.
- **Les répertoires** (ou dossiers) permettent d'organiser les fichiers du système. En effet, ce sont des fichiers dont le rôle est de contenir d'autres fichiers (spéciaux, ordinaires ou répertoires). Cela permet notamment de regrouper dans un même répertoire les fichiers ayant des caractéristiques communes (même propriétaire, les programmes, les documentations, . . .), et de hiérarchiser les fichiers du système.
- les **fichiers spéciaux** regroupent les 5 autres types de fichiers. Sans rentrer dans les détails, ce sont :
  - les **fichiers liens** (ou raccourcis) : Ces fichiers donnent la possibilité de donner plusieurs noms logiques à un même fichier physique. Un nouveau point d'accès au fichier est par conséquent créé..
  - 2 types de fichiers spéciaux représentent les périphériques (auxquels sont associés des pilotes ou drivers) :
    - les **fichiers de périphériques en mode caractère** représentent notamment les claviers, souris, terminaux, modems, imprimantes, etc ;
    - les **fichiers de périphériques en mode bloc** représentent notamment les disques durs (et leurs partitions), clés USB, CD/DVD, etc ;
  - Les fichiers de communication entre les processus du système :
    - les fichiers **socket** (ou socket unix) permettent la communication bidirectionnelle inter-processus (sur système local ou distant) ;
    - les **tubes** (appelés FIFO) passent les informations entre processus par FIFO (First In First Out). Un processus écrit de informations transitoires dans un fichier *pipe* et un autre les lit. Après lecture, les informations ne sont plus accessibles..

A. Taux d’occupation

1. Par système de fichiers

La commande **df** affiche les systèmes de fichiers montés mais elle affiche également la quantité d'espace occupé des systèmes de fichiers
Par défaut, l'espace est donné en blocs de 1k.

Exemple :

```
prompt$ df -h
Sys. de fichiers   Taille  Uti.                  Disp.  Uti% Monté sur
/dev/sda6                  14G    5,7G          7,5G  44%  /
tmpfs                      1,5G  0              1,5G   0%  /lib/init/rw
udev                       1,5G  268K           1,5G   1%     /dev
tmpfs                      1,5G  12K            1,5G   1%  /dev/shm
/dev/sda7                  406G 117G           268G  31% /home
```

2. Par arborescence

La commande **du** affiche les statistiques sur l'utilisation du disque, mais cette fois en se basant sur les répertoires, et non plus les périphériques.

La commande suivante affiche la taille utilisée pour /usr et /bin en utilisant une unité compréhensible par les humains (-h ou -human-readable) et n'affiche que le total (-s ou -summarise).

```
prompt$ du -sh /usr
4,8G  /usr
prompt$  du -sh /bin
4,9M  /bin
```

B. Droit d’accès et attribut des fichiers

1. Droits de base

Le rôle d’un système d’exploitation est aussi d’assurer la sécurité et l’accès aux données, ce qui est possible grâce au mécanisme des droits. Chaque fichier ou répertoire se voit attribuer des droits qui lui sont propres, des autorisations d’accès individuelles. Lors d’un accès le système vérifie si celui-ci est permis.

À sa création par l’administrateur, un utilisateur se voit affecter un **UID** (User Identification) unique.

Les utilisateurs sont définis dans le fichier **/etc/passwd**. De même chaque utilisateur est rattaché à un groupe au moins (groupe principal), chaque groupe possédant un identifiant unique, le **GID** (Group Identification). Les groupes sont définis dans **/etc/group**.

La commande **id** permet d'obtenir ces informations. En interne, le système travaille uniquement avec les UID et GID, et pas avec les noms eux-mêmes.

```
prompt$ id
uid=1000(olivier)gid=1000(olivier),groupes=1000(olivier),4(adm),
20(dialout),124(wireshark),999(docker)
```

À chaque fichier (**inode**) sont associés un **UID** et un **GID** définissant son propriétaire et son groupe d'appartenance.

Vous affectez des droits pour le propriétaire, pour le groupe d'appartenance et pour le reste du monde (les autres). On distingue trois cas de figure :

- UID de l'utilisateur identique à l'UID défini pour le fichier. Cet utilisateur est propriétaire du fichier.
- Les UID sont différents : le système vérifie si le GID de l'utilisateur est identique au GID du fichier. Si oui l'utilisateur appartient au groupe associé au fichier.
- Dans les autres cas (aucune correspondance) : il s'agit du reste du monde (others), ni le propriétaire, ni un membre du groupe.

Exemple :

```
prompt$ ls -lh fic01
-rw-r--r-- 1 olivier adm 2,9K oct.  16 21:34 fic01
```

Sur cette ligne du tableau, le fichier fic01 appartient à l'utilisateur « olivier » et au groupe « adm », et possède les droits -rw-r--r-- (exprimé en mode symbolique).

Le tableau ci-dessous résume les permissions de fichiers en mode symbolique.

Droit	Signification
Général	
r	Readable (lecture).
w	Writable (écriture).
x	Executable (exécutable comme programme).
Fichier normal	
r	Le contenu du fichier peut être lu, chargé en mémoire, visualisé, recopié.
w	Le contenu du fichier peut être modifié, on peut écrire dedans. La suppression n'est pas forcément liée à ce droit (voir droits sur répertoire).
x	Le fichier peut être exécuté depuis la ligne de commande, s'il s'agit soit d'un programme binaire (compilé), soit d'un script (shell, perl...).
Répertoire	
r	Les éléments du répertoire (catalogue) sont accessibles en lecture. Sans cette autorisation, ls et les critères de filtre sur le répertoire et son contenu ne sont pas possibles. L'accès individuel à un fichier reste possible si vous connaissez son chemin.
w	Les éléments du répertoire (catalogue) sont modifiables et il est possible de créer, renommer et supprimer des fichiers dans ce répertoire. **C'est ce droit qui contrôle l'autorisation de suppression d'un fichier.**
x	Le catalogue peut être accédé par CD et listé. Sans cette autorisation il est impossible d'accéder au répertoire et d'agir sur son contenu qui devient verrouillé.

Ainsi pour un fichier :

rwx	r-x	r--
Droits de l'utilisateur, en lecture, écriture et exécution.	Droits pour les membres du groupe en lecture et exécution.	Droits pour le reste du monde en lecture uniquement.

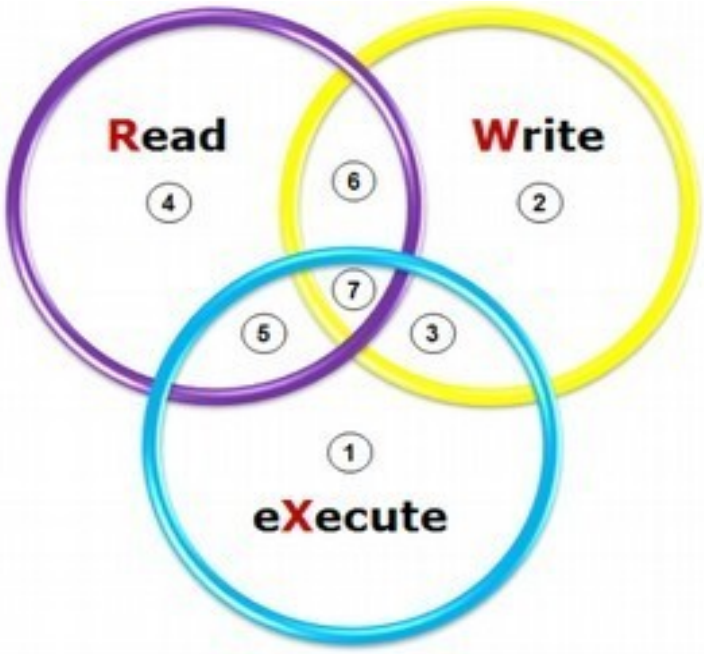
2. Modification des droits

Lors de sa création, un fichier ou un répertoire dispose de droits par défaut. On utilise alors la commande **chmod** (change mode) si l'on souhaite modifier les droits.

On peut le faire selon deux méthodes :
- symbolique
- octale

Correspondances de représentation des droits		
Droit	**Valeur alphanumérique**	**Valeur octale**
aucun droit	---	0
exécution seulement	--x	1
écriture seulement	-w-	2
ecriture et exécution	-wx	3
lecture seulement	r--	4
lecture et exécution	r-x	5
lecture et écriture	rw-	6
tous les droits (lecture, écriture et exécution)	rwx	7

Un autre moyen de se repérer quand on débute avec les droits linux :



Si l'on veut qu'un dossier soit accessible en lecture seule, il faut positionner dessus les droits d'exécution (pour permettre l'accès) et de lecture pour pouvoir le parcourir.

La méthode symbolique :

```
[root]# chmod u+rwx,g+wx,o-r /tmp/fichier1
[root]# chmod g=x,o-r /tmp/fichier2
[root]# chmod o=r /tmp/fichier3

[root]# ls -l /tmp/fic*
-rwxrwx--- 1 root root ... /tmp/fichier1
-rwx--x--- 1 root root ... /tmp/fichier2
-rwx--xr-- 1 root root ... /tmp/fichier3
```

Quel utilisateur ?		
u	user/owner	utilisateur / propriétaire
g	group	groupe
o	other	les autres
a	all	tous les utilisateurs
Que faire ?		
+	add this permission	ajouter cette permission
-	remove this permission	enlever cette permission
=	set exactly this permission	ne positionner que cette permission
Quelles permissions?		
r	read	lire
w	write	écrire
x	execute	exécuter

La méthode octale :

```
[root]# chmod 741 /tmp/fichier1
[root]# chmod -R 744 /tmp/fichier2

[root]# ls -l /tmp/fic*
-rwxr----x 1 root root ... /tmp/fichier1
-rwxr--r-- 1 root root ... /tmp/fichier2
```

Le tableau suivant vous aide :

Propriétaire			Groupe			Reste du monde		
r	**w**	**x**	**r**	**w**	**x**	**r**	**w**	**x**
400	200	100	40	20	10	4	2	1

Pour obtenir le droit final il suffit d'additionner les valeurs.

Par exemple si vous voulez **rwxr-x-r-x**

→ alors obtenez : 400+200+100+40+10+4+1=**755**

et pour rw-r--r--

→ vous obtenez : 400+200+40+4=**644**

On constate que pour gérer les droits la méthode symbolique est la plus fine. En effet elle permet de modifier un seul droit tandis qu'avec la méthode octale c'est l'intégralité des droits qui est modifiée en une fois.

3. Masque des droits

Lors de la création d'un fichier ou d'un répertoire, des droits leur sont automatiquement assignés. **Généralement**, c'est rw-r--r-- (644) pour un fichier et rwxr-xr-x (755) pour un répertoire. Ces valeurs sont contrôlées par un masque, lui-même modifiable par la commande **umask**. La commande **umask** prend comme paramètre une valeur octale dont chaque droit individuel sera supprimé des droits d'accès maximum du fichier ou du répertoire.

- Par défaut, tous les fichiers sont créés avec les droits 666 (rw-rw-rw-).
- Par défaut tous les répertoires sont créés avec les droits 777 (rwxrwxrwx).
- Puis le masque est automatiquement appliqué.
- Le masque est le même pour l'ensemble des fichiers.
- Un masque ne modifie pas les droits des fichiers existants, mais seulement ceux des nouveaux fichiers.

Les droits par défaut (maximum) des fichiers et des répertoires ne sont pas identiques. C'est logique : comme le droit x permet de rentrer dans un répertoire, il est normal que celui-ci en dispose par défaut.

Ce même droit (x) est inutile par défaut sur les fichiers : seule une très petite minorité des fichiers sont des scripts et des programmes binaires.

Le paramètre umask est géré dans le fichier /etc/profile.

Le masque par défaut est **généralement** 022, soit ----w--w-. Cela signifie que pour tout dossier ou fichier créé, le système enlève automatiquement le droit d'écriture pour le groupe de l'utilisateur et pour les autres.

Pour obtenir cette valeur, tapez **umask** sans paramètre.

```
prompt$ umask
0022
```

Calcul de masque

Pour un fichier

```
Défaut  rw-rw-rw- (666)
Retirer ----w--w- (022)
Reste   rw-r--r-- (644)
```

Pour un répertoire

```
Défaut  rwxrwxrwx (777)
Retirer ----w--w- (022)
Reste   rwxr-xr-x (755)
```

Notez qu'appliquer un masque n'est pas soustraire, mais supprimer des droits de ceux par défaut, droit par droit. Par exemple :

```
Défaut  rw-rw-rw- (666)
Retirer ----wxrwx (037)
Reste   rw-r----- (640)
```

Et non 629, ce qui est impossible en octal…

Une fois le masque défini, il faut éditer le fichier /etc/profile et relancez le système pour qu'il soit pris en compte.

4. Changer de propriétaire et de groupe

Modification de l'utilisateur et du groupe avec chown et chgrp :

```
prompt# ls -lh fic01
-rw-r--r-- 1 olivier adm 2,9K oct.  16 21:34 fic01
prompt# chown root fic01
prompt# ls -lh fic01
-rw-r--r-- 1 root adm 2,9K oct.  16 21:34 fic01
prompt# chgrp groupe01 fic01
prompt# ls -lh fic01
-rw-r--r-- 1 root groupe01 2,9K oct.  16 21:34 fic01
```

Remarque : Les trois outils chmod, chown et chgrp ont l'option très pratique -R (R est bien en majuscule) qui applique les modifications récursivement sur les répertoires et fichiers indiqués.

C. Annexes

1. Les droits d'accès étendus : SUID, SGID et Sticky-bit

a. SUID, SGID

Ces droits permettent d'exécuter une commande suivant les droits positionnés sur la commande et non plus suivant les droits de l'utilisateur.

La commande s'exécute avec l'identité du propriétaire (**suid**) ou du groupe (**sgid**) de la commande.

Il s'agit d'une possibilité supplémentaire de droits d'accès attribués à un utilisateur lorsqu'il est nécessaire qu'il dispose des mêmes droits que ceux du propriétaire d'un fichier ou ceux du groupe concerné.

En effet, un utilisateur peut avoir à exécuter un programme (en général un utilitaire système) mais ne pas avoir les droits d'accès nécessaires. En positionnant les droits adéquats ("**s**" au niveau du propriétaire et/ou au niveau du groupe), l'utilisateur du programme possède, pour le temps d'exécution de celui-ci, l'identité du propriétaire (ou celle du groupe) du programme.

Exemple :

Le fichier **/usr/bin/passwd** est un fichier exécutable (une commande) qui porte un **SUID**.

Lorsque l'utilisateur bob va le lancer, ce dernier devra accéder au fichier **/etc/shadow**, or les droits sur ce fichier ne permettent pas à bob d'y accéder.

Ayant un **SUID** cette commande sera exécutée avec l'UID de root et le GID de root. Ce dernier étant le propriétaire du fichier **/etc/shadow**, il aura les droits en lecture.

La mise en place du SUID et SGID peut s'effectuer comme ci-dessous.

Méthode octale :

```
[root]# chmod 4777 fic1
[root]# chmod 2777 fic2
```

Méthode symbolique :

```
[root]# chmod u+s fic1
[root]# chmod g+s fic2
```

Résultat :

```
[root]# ls -l
-rwsrwxrwx ... fic1
-rwxrwsrwx ... fic2
```

SGID et les répertoires

Ce droit a une tout autre utilisation s'il est appliqué à un répertoire. Normalement, lorsqu'un fichier est créé par un utilisateur, il en est propriétaire, et un groupe par défaut lui est appliqué (celui du propriétaire).
Cependant, lorsqu'un fichier est créé dans un répertoire portant le droit SGID, alors ce fichier se verra attribuer par défaut **le groupe du répertoire** (et non celui du propriétaire). De plus, si c'est un autre répertoire qui est créé dans le répertoire portant le droit SGID, ce sous-répertoire portera également ce droit.

b. Sticky-bit

La permission sticky bit avec la valeur 1000 a l'effet suivant :

- appliqué à un répertoire, il empêche les utilisateurs de supprimer les fichiers dont ils ne sont pas propriétaires (idéal pour un répertoire partagé par un groupe) le sticky bit est typiquement utilisé pour /tmp.

- appliqué à un fichier, il était utilisé pour charger le fichier en mémoire pour en accélérer les futurs accès ou exécutions.

La valeur symbolique pour un exécutable est **t** ou **T** pour un fichier non exécutable.

Méthode octale :

```
[root]# chmod 1777 repertoire
```

Méthode symbolique :

```
[root]# ls -ld repertoire
drwxrwxrwx ... repertoire
[root]# chmod o+t repertoire
[root]# ls -l
drwxrwxrw**t** ... repertoire
```

2.Les attributs sur les fichiers

Au côté des permissions standard, il existe un autre système pour modifier la façon dont on peut accéder à un fichier. Les attributs de fichiers ne s'affichent pas avec la commande **ls** mais avec la commande **lsattr**. On utilise la commande **chattr** pour paramétrer ou supprimer ces attributs.

Les attributs suivants existent. Veuillez noter les cas d'utilisation :

- **« A »** : Quand on accède à un fichier avec l'attribut A, sa valeur atime (temps d'accès) n'est pas modifiée. Ceci évite un certain nombre d'accès disques, ce qui est plutôt pratique pour des fichiers temporaires. Attention cependant, certains outils comme tmpwatch se basent sur la valeur du atime pour déterminer l'utilisation récente d'un fichier, donc si le atime n'est pas mis à jour l'état du fichier sera mal interprété.
- **« a »** : L'accès en écriture aux fichiers avec l'attribut « a » est limité en ajout (append). Seul le superutilisateur ou un processus disposant de la capacité CAP_LINUX_IMMUTABLE peut définir ou supprimer cet attribut. L'utilisation la plus évidente est pour les fichiers journaux système, pour empêcher un intrus de retirer les traces de leur passage. Cependant, vous devriez garder à l'esprit qu'un intrus a besoin des droits root pour éditer ces fichiers journaux. Par conséquent, s'il les a, il peut retirer l'attribut « a », modifier les fichiers puis rétablir l'attribut « a ».
- **« c »** : un fichier avec l'attribut « c » est automatiquement compressé par le noyau. Vous accédez à vos données décompressées en lecture, et l'écriture sur le fichier compresse les données avant de les stocker sur le disque. Note : Même si l'attribut est défini sur un fichier et qu'il est affiché avec la commande lsattr, il n'est pas traité par les pilotes du noyau pour Ext2 ou Ext3.
- **« D »** : les modifications sur un répertoire avec l'attribut « D » sont écris de façon synchrone sur le disque. C'est l'équivalent de l'option dirsync de mount mais ici appliquée à un sous-ensemble de fichiers. Avec cet attribut, les opérations suivantes sont synchrones dans le répertoire : create, link, unlink, symlink, mkdir, rmdir, mknod et rename.
- **« d »** : un fichier avec l'attribut « d » ne sera pas sauvegardé quand la commande dump sera lancée.
- **« i »** : un fichier avec l'attribut « i » (immutable = immuable) ne peut pas être modifié. Il est impossible de le supprimer ou de le renommer, ni de créer un lien vers ce fichier, ni d'écrire dans le fichier. Seul le superutilisateur ou un processus ayant la capacité CAP_LINUX_IMMUTABLE peut définir ou retirer cet attribut.
- **« j »** : les données d'un fichier ayant l'attribut « j » sont inscrites dans le journal ext3 avant d'être inscrites dans le fichier si le système de fichiers est monté avec l'option « data=ordered » ou « data=writeback ». Lorsque le système de fichiers est monté avec l'option « data », l'attribut est sans effet puisque les données de tous les fichiers sont écrites dans le journal. Seul le super utilisateur ou un processus ayant la capacité CAP_SYS_RESSOURCE peut définir ou retirer cet attribut.

- **« s »** : lorsqu'un fichier avec l'attribut « s » est supprimé, ses blocs sont remis à zéro et réinscrits sur le disque. Note : comme pour l'attribut « c », cet attribut n'est pas traité par les pilotes de système de fichiers ext2 et ext3.
- **« S »** : les données d'un fichier ayant l'attribut « S » sont inscrites sur le disque de façon synchrones. C'est l'équivalent de l'option de « sync » de mount appliquée à un sous-ensemble de fichiers. Cet attribut est généralement utilisé pour les fichiers dits « cooked files », les fichiers dans lesquels les serveurs de bases de données stockent leurs informations. Ainsi, on contourne l'ensemble des deux systèmes de cache des pilotes du noyau et le cache de la base de données, optimisé, écrit directement sur le disque.
- **« T »** : l'attribut « T » n'est disponible qu'à partir des noyaux 2.6.x. Son but est d'indiquer le sommet de la hiérarchie, fonctionnalité utilisée par l'algorithme « Orlov block allocator ». Les nouvelles règles d'allocation de fichiers sur les systèmes de fichiers ext2 et ext3 rapprochent les sous-répertoires de façon à accélérer les accès sur une arborescence, si l'arborescence a été créée avec un noyau 2.6.
- **« t »** : le dernier bloc d'un fichier avec l'attribut « t » ne pourra pas être partagé avec d'autres fichiers (pour les systèmes de fichiers qui gèrent le « tail-merging »). C'est nécessaire pour des applications comme LILO qui accèdent directement au système de fichier et qui ne sont pas capables d'interpréter les blocs partagés. Note : au moment de l'écriture de la documentation, ext2 et ext3 ne font pas, en dehors de rustines très expérimentales, de « tail-merging ».
- **« u »** : lorsqu'on supprime un fichier ayant l'attribut “ u ”, son contenu est sauvegardé. Ainsi, l'utilisateur peut demander sa récupération. Encore un attribut qui est géré par tous sauf le noyau.

Exemple :

```
prompt$ lsattr fichier1
------------------- fichier1
prompt$ chattr +i fichier1
chattr: Opération non permise lors de l'initialisation des
drapeaux sur fichier1
prompt$ lsattr fichier1
------------------- fichier1
```

Seul le root peut modifier les attributs

```
prompt# chattr +i fichier1
prompt# lsattr fichier1
----i--------------- fichier1
prompt# rm fichier1
rm: impossible de supprimer « fichier1 »: Opération non permise
prompt# rm -f fichier1
rm: impossible de supprimer « fichier1 »: Opération non permise
prompt# chattr -i fichier1
prompt# rm fichier1
prompt# ls fichier1
ls: impossible d'accéder à fichier1: Aucun fichier ou dossier de
ce type
```