UNIX

SHELL / SYSTEME DE FICHIERS

Table des matières

1 Le Shell 4
1.1 Qu'est-ce que le Shell 4
1.2 Début et fin de session 4
1.3 Principe général des commandes 5
1.4 Options et paramètres 6
1.5 Noms de fichier génériques. 7

2 Les fichiers 8
2.1 Fichier ordinaire 9
2.2 Fonctionnalités supplémentaires 9
2.3 Noms de fichiers 9

3 Les répertoires 9
3.1 Le répertoire simple 9
3.2 Représentation logique du système de fichiers 10

4 Les Liens 11
4.1 Les liens physiques (ou hard en anglais, ou encore durs en français) 11
4.2 Les liens symboliques 11

5 Droits d'accès. 12

6 Principales commandes utiles pour les fichiers et répertoires 13
6.1 Consulter le manuel 13
6.2 Identifier les utilisateurs du système 14
6.3 Changer de mot de passe 14
6.4 Afficher une chaine de caractères 15
6.5 Commandes et manipulation des répertoires 15
6.6 Changer de répertoire 15
6.7 Créer et détruire un répertoire 15
6.8 Commandes et manipulation des fichiers 16
6.9 Imprimer un fichier 16
6.10 Chercher une chaine de caractères dans un fichier 16
6.11 Faire une copie d'un fichier 17
6.12 Déplacer ou changer le nom d'un fichier 17

6.13 Identifier le type d'un fichier 17
6.14 Créer un lien sur un fichier 17
6.15 Détruire un fichier 17
6.16 Rechercher un fichier 18
6.17 Permission sur les fichiers 18

1 Le Shell

1.1 Qu'est-ce que le Shell

Le **shell** est l'interprète des commandes d'UNIX. C'est le programme qui dialogue avec vous aussi longtemps que votre poste de travail n'est pas sous le contrôle d'une commande spécifique. Dès que la procédure de connexion (login) est terminée, le **shell** prend le contrôle et vous écoute. Il lit chaque commande que vous tapez, il en développe les éléments génériques (comme les familles de noms de fichiers) et il y remplace les éléments symboliques (comme les variables d'environnement).

De plus, le **shell** comporte un mécanisme d'enchaînement des commandes qui en fait un vrai langage de programmation.

A cause de son rôle d'interface, le **shell** représente pour l'utilisateur la « face visible » d'UNIX. On pourrait croire qu'il s'agit d'un programme très spécial, aux privilèges exorbitants. Il n'en est rien. Mis à part le fait d'être automatiquement lancé par le système au début d'une session, le **shell** est un programme comme les autres, sans aucun droit particulier. Plusieurs interprètes de commandes sont disponibles parfois sur une même machine. De plus, rien ne vous empêche d'écrire votre propre shell.

Les interprètes de commandes les plus répandus sont le « **Bourne shell** » (**sh**), du nom de son auteur Steve Bourne, et le « **C shell** » (**csh**), ainsi appelé parce que, comme langage de programmation, il est assez proche de C. Le C shell est plus riche et plus régulier que le Bourne shell, mais nous étudierons ce dernier car il est plus répandu et plus simple. On le trouve normalement par défaut sur toute distribution UNIX.

Pour chaque utilisateur, le nom du shell qui doit être activé lors de sa connexion est précisé dans le fichier `/etc/passwd` (c'est le dernier champ de la ligne le concernant). C'est le responsable du système qui définit ou modifie ce genre d'informations. Si les machines sont organisées en réseau, il est possible que le fichier `/etc/passwd` de chaque machine ne reflète pas l'ensemble des utilisateurs et des mots de passe.

1.2 Début et fin de session

Si sur un poste donné aucun utilisateur n'est connecté au système, alors l'écran doit afficher la question suivante (si ce n'est pas le cas, appuyez sur la touche retour-chariot, elle apparaîtra) :

```
pcN login:
```

En réponse vous devez taper le « nom d'utilisateur » que vous a attribué le responsable du système.

Attention, vous devez taper ce nom en minuscules, autrement UNIX croira que vous utilisez un mauvais terminal ne disposant que des majuscules et la suite deviendra pénible.

Si vous avez défini un mot de passe, la question suivante est :

```
Password:
```

Vous devez y répondre en tapant votre mot de passe secret. Exemple de dialogue (les interventions de l'utilisateur sont soulignées) :

```
UNIX(r) System V Release 4.0 (big)
login: zigomar
Password: ~~~~~~
Login incorrect (password erroné : connexion ratée)
login: zigomar
Password: ~~~~~~
Last login: Mon Nov 24 08:52:39 from pc30
You have mail.
big:~$
```

Si la connexion réussit, le système affiche quelques informations d'intérêt général, comme la date de votre dernière connexion, le fait que vous avez reçu du courrier que vous n'avez pas encore lu, ou des messages de la part du responsable du système, puis active un shell qui se met en attente de vos commandes.

Il est fortement conseillé, pour ne pas dire obligatoire, surtout s'il s'agit d'une machine connectée à un réseau informatique, d'installer un mot de passe, pour se mettre à l'abri de la malveillance et, surtout, de l'étourderie des autres utilisateurs. Pour définir ou changer votre mot de passe, composez la commande

passwd

Elle vous demandera de composer (en aveugle, tout cela est secret) l'ancien et le nouveau mot de passe, ce dernier deux fois.

Fin de session.

Pour terminer une session (du Bourne shell) vous devez taper une ligne réduite à l'unique caractère *Ctrl-D*. Attention, si la ligne ne commence pas par Ctrl-D cela ne marchera pas, vous devrez la refaire. Une session du C shell se termine généralement en composant la commande

logout

1.3 Principe général des commandes

Dès la fin de la procédure de connexion, chaque fois qu'une commande particulière ne détient pas le contrôle, le shell est à votre l'écoute. Il manifeste ce fait par l'affichage d'un prompt, généralement (en Bourne shell) le caractère $. Vous devez alors composer une ligne, qui ne sera prise en compte qu'à la frappe d'un retour-chariot. Le plus souvent une ligne contient une commande, mais ce n'est pas obligatoire, car on peut enchaîner plusieurs commandes sur une même ligne, et certaines commandes s'étendent sur plusieurs lignes.

Le premier mot du texte que vous composez pour activer une commande identifie la commande en question, c'est son nom. Il est généralement suivi de plusieurs autres expressions qui indiquent des options et des paramètres de la commande. Il y a trois sortes de commandes, qu'on utilise de la

même manière :

- Les commandes internes,
- Les programmes,
- Les scripts.

Les commandes internes expriment les fonctionnalités du shell lui-même ; elles sont reconnues et exécutées sans l'aide d'un autre fichier. Toute commande qui n'est pas interne est supposée être le nom d'un fichier exécutable, qui peut être soit un fichier du système, c'est à dire qui a été livré avec UNIX, soit un fichier crée par un utilisateur. Dans un cas comme dans l'autre ce fichier peut être un fichier binaire exécutable, par exemple le résultat de la compilation d'un programme écrit en C, ou bien une procédure de commande ou script. Un script est un fichier de texte entièrement composé de commandes.

Comme les commandes externes n'appartiennent presque jamais au répertoire de travail, il serait très pénible de devoir les spécifier par les références complètes des fichiers correspondants. Pour l'éviter, le shell maintient constamment une liste de répertoires où chercher les commandes, c'est la valeur de la variable d'environnement **PATH**. Ainsi, si la valeur de cette variable est la chaîne « :/bin :/usr/bin » et si vous tapez le mot « job » (qui n'est pas le nom d'une commande interne) alors le shell tentera d'exécuter « ./job » puis, en cas d'échec, « /bin/job », puis encore « /usr/bin/job ».

1.4 Options et paramètres

La quasi-totalité des commandes prédéfinies (et vos programmes aussi, s'ils ont l'esprit UNIX) suivent les conventions suivantes :

– Le nom de la commande est souvent très court, donc rarement expressif

– Après le nom de la commande on trouve un nombre quelconque d'options, de la forme

-caractère

dans le cas d'une option qui n'exige pas une valeur, et

-caractère valeur

lorsqu'une valeur est nécessaire.

La nature de l'option, indiquée par le caractère qui lui sert de nom, détermine si une valeur doit ou non être présente.

– Dans beaucoup de commandes, si aucune option ne comporte de valeur ou si une seule option en comporte, on doit regrouper les options. Par exemple on nous fait écrire « -abc » à la place de « -a -b -c » et « -xvf valeur » à la place de « -x -f valeur -v ».

– L'ordre des options entre elles est sans importance. L'ordre des noms de fichier entre eux dépend de la nature de la commande (voir le point suivant)

– Après le nom de la commande on trouve un nombre quelconque de noms de fichiers, qui sont les entrées de la commande. Selon la nature de cette dernière, le travail sera fait soit sur la concaténation des fichiers, soit successivement sur chacun d'eux

– Lorsqu'aucun nom de fichier n'a été indiqué, la commande lit ses données éventuelles sur son entrée standard. Chaque fois que la commande le permet, les données sont sous forme textuelle.

– La commande écrit ses résultats éventuels sur sa sortie standard. Chaque fois que la commande le permet, ces résultats sont sous forme textuelle.

Exemples :

```
cat fic1 fic2 fic3
```

la commande cat (pour catenate) est tout à fait minimale : elle ne fait qu'écrire les caractères qu'elle lit. Mais elle suit les conventions indiquées. Par conséquent, l'expression ci-dessus affiche la concaténation des trois fichiers fic1, fic2 et fic3.

```
cc -g -o monfic.o -c monfic.c
```

cette expression est un appel du compilateur C (cc, pour C compiler) avec les options « -g », « -o monfic.c » et « -c » (les options -g et -c de la commande cc ne requièrent pas de valeur, l'option -o oui), et avec pour entrée le fichier « monfic.c ». Cette commande n'écrit rien sur la sortie standard.

1.5 Noms de fichier génériques.

Beaucoup de commandes du shell ont des noms de fichier pour arguments. Ces noms peuvent comporter les spécifications génériques `*`, `?` et `[...]` avec la signification suivante :

`*` n'importe quelle chaîne de caractères (y compris la chaîne vide),

`?` n'importe quel caractère unique,

`[...]` n'importe quel caractère unique parmi ceux d'un ensemble. On a droit aux spécifications :

c le caractère c,

c1-c2 les caractères compris entre *c1* et *c2*.

Ces expressions sont comprises comme des motifs de recherche dans l'ensemble des noms de fichiers réellement existants dans le système. A l'endroit où elle figure, l'expression générique représente la liste des noms de fichier qui s'unifient avec le motif de recherche.

Par exemple, l'expression

```
*.c
```

représente la liste de tous les fichiers du répertoire de travail dont le nom se termine par « .c ».

L'expression

```
/tmp/*.[ch0-9]
```

représente la liste des fichiers du répertoire /tmp dont le nom se termine par un point suivi de c, h ou un chiffre.

Et l'expression

```
/usr/users/henri/*/core
```

représente la liste de tous les fichiers de nom core qui se trouvent dans un sous-répertoire immédiat (un niveau) de `/usr/users/henri` (comme `/usr/users/henri/p2c/core`).

2 Les fichiers

Un fichier est vu par Unix comme une suite d'octets. La structuration des données n'est pas imposée par le système et est donc de la responsabilité l'utilisateur

Unix *ne distingue pas* les fichiers ASCII, binaires ou autres

Nom de fichier = 1 à 255 caractères *quelconques*

Un système de fichiers (*file system*) est une arborescence de fichiers (structure hiérarchique de **répertoires** et de **fichiers**).

Les répertoires peuvent contenir des fichiers et/ou des sous-répertoires. A chaque fichier sont associés des **droits d'accès**.

Répertoires et sous-répertoires sont traités comme des fichiers.

Le sommet de cette arborescence est le répertoire appelé *racine* (*root*) noté "/".

La racine contient des fichiers système et des répertoires :

/bin, /dev, /lib, /etc/usr, /usr, /sbin, /usr/local, /home ...

Les fichiers standards : il existe 3 fichiers standards (utilisés par défaut):

entrée standard : *clavier*

sortie standard : *écran*

erreur standard : *écran*

Sous Unix un fichier est :

- Toujours désigné par un nom.
- Possède un unique inode (certaines informations concernant le fichier).
- Possède les fonctionnalités suivantes :
  - ouverture.
  - fermeture.
  - lecture (consultation).
  - écriture (modification)

Un fichier peut être :

- ordinaire (on utilise parfois le terme "normal") (-)
- un répertoire (d)

- un lien symbolique (l)
- un pseudo-fichier :
  - accès caractère par caractère (c)
  - dispositif de communication (p)
  - accès par bloc (b)

2.1 Fichier ordinaire

Un fichier ordinaire est caractérisé par :

- son contenu vu comme une suite d'octets (i.e. caractères).
- après l'ouverture l'accès s'effectue au niveau du pointeur du fichier.

0	1	2	3	4	5	6	7	8	9	...
P	R	O	G	R	A	M	_	T	R	I

2.2 Fonctionnalités supplémentaires

- Positionnement du pointeur de fichier.
- Ecriture en fin de fichier ayant pour effet l'extension du fichier.

2.3 Noms de fichiers

Unix fait la différence entre les majuscules et les minuscules (la casse). En théorie tous les caractères du clavier sont autorisés. En pratique, il vaut mieux s'abstenir pour certains (comme * ou ?). Le caractère - est déconseillé au début d'un nom de fichier (certaines commandes pourraient l'interpréter comme une option).

3 Les répertoires

3.1 Le répertoire simple

Comme tout fichier, un répertoire possède son inode.

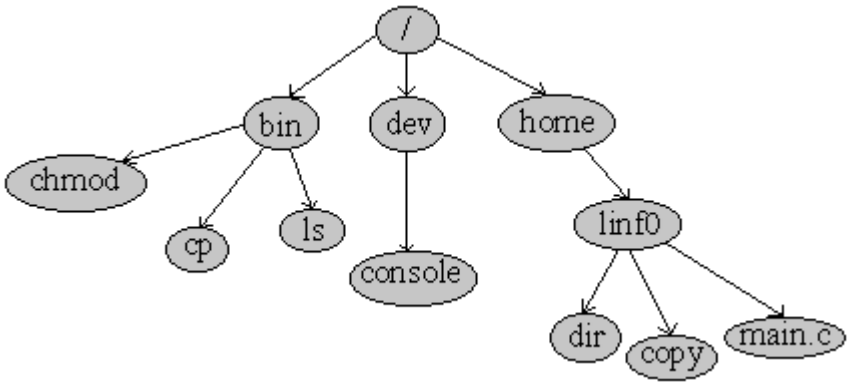
Le contenu du répertoire représente la correspondance entre les noms de fichiers et les inodes. Les noms " **.** " et " **..** "figurent dans tout répertoire. " **.** " correspond au répertoire courant et " **..** " au répertoire supérieur (ou répertoire parent).

Nom	Numéro d'inode

.	2
..	2
bin	3
dev	5
home	9
...	

3.2 Représentation logique du système de fichiers

C'est une arborescence :



- La désignation d'un chemin peut être :
  - absolu : Par exemple : `/home/linf0/dir`.
  - relatif :par exemple : si le répertoire courant est /home il suffit de taper `linf0` pour accéder à /home/linf0.

Ci-dessous la liste des dossiers les plus courants que l'on retrouve à la racine de Linux.

- bin : contient des programmes (exécutables) susceptibles d'être utilisés par tous les utilisateurs de la machine.
- boot : fichiers permettant le démarrage de Linux.
- dev : fichiers contenant les périphériques. En fait – on en reparlera plus tard – ce dossier contient des sous-dossiers qui « représentent » chacun un périphérique. On y retrouve ainsi par exemple le fichier qui représente le lecteur CD.
- etc : fichiers de configuration.
- home : répertoires personnels des utilisateurs.
  Chaque utilisateur de l'ordinateur possède son dossier personnel. Par exemple, pour l'utilisateur Thomas, le dossier personnel se trouve dans/home/thomas/.
- lib : dossier contenant les bibliothèques partagées (généralement des fichiers.so) utilisées par les programmes. C'est en fait là qu'on trouve l'équivalent des.dllde Windows.

- media : lorsqu'un périphérique amovible (comme une carte mémoire SD ou une clé USB) est inséré dans votre ordinateur, Linux vous permet d'y accéder à partir d'un sous-dossier demedia. On parle de montage.
- mnt : c'est un peu pareil quemedia, mais pour un usage plus temporaire.
- opt : répertoire utilisé pour les add-ons de programmes.
- proc : contient des informations système.
- root : c'est le dossier personnel de l'utilisateur « root ». Normalement, les dossiers personnels sont placés danshome, mais celui de « root » fait exception. En effet, comme je vous l'ai dit dans le chapitre précédent, « root » est le superutilisateur, le « chef » de la machine en quelque sorte. Il a droit à un espace spécial.
- sbin : contient des programmes système importants.
- tmp : dossier temporaire utilisé par les programmes pour stocker des fichiers.
- usr : c'est un des plus gros dossiers, dans lequel vont s'installer la plupart des programmes demandés par l'utilisateur.
- var : ce dossier contient des données « variables », souvent des logs (traces écrites de ce qui s'est passé récemment sur l'ordinateur).

Cette liste de dossiers est en fait présente sur tous les OS de type Unix, et pas seulement sous Linux.

4 Les Liens

4.1 Les liens physiques (ou hard en anglais, ou encore durs en français)

Sous Unix, un même fichier peut avoir plusieurs noms : c'est ce qu'on appelle un "lien dur". On les crée avec la commande ln. Par exemple, si j'ai un fichier toto, et si je tape la commande **`ln toto titi`**, j'obtiens titi qui est un nouveau lien dur sur le fichier toto. toto et titi représentent le même fichier: si je modifie titi, les changements apparaîtront aussi dans toto. En particulier, il existe un seul exemplaire du fichier sur le disque : la création d'un lien dur ne prend pas de place.

A partir de ce moment, toto et titi sont deux liens vers le même fichier, aucun des deux ne représentant le fichier plus que l'autre. Si maintenant j'efface toto, le fichier existe encore sous le nom de titi.

Un fichier est définitivement supprimé quand son dernier lien est effacé.

4.2 Les liens symboliques

Les liens durs ont pas mal de défauts : notamment, tous les liens d'un fichier doivent être sur le même disque physique, et les liens sur des répertoires posent des problèmes de boucles avec les programmes qui examinent toute une arborescence. A part dans quelques cas précis, on considère qu'il est préférable d'employer des liens symboliques. Comme un lien dur, un lien symbolique est un alias d'un fichier. Mais contrairement au lien dur, le lien symbolique n'est qu'un alias : on distingue bien le fichier lui-même et les liens, qui ne veulent plus rien dire si le fichier est effacé.

Par exemple, j'ai toujours mon fichier toto et je veux créer un lien symbolique titi. Pour cela, je tape **`ln -s toto titi`**. Là encore, si je modifie toto, titi sera également affecté, et vice-versa. Par contre, si j'efface toto et que je veux ensuite modifier titi, le système va donner une erreur: titi est un lien symbolique invalide, puisque le fichier qu'il désignait a disparu.

Commande pour ajouter un nouveau lien :

`ln nom nom_supplémentaire`

Exemple :

Répertoire courant /home/linf13

```
ln /bin/chmod droits
```

Lien symbolique : ln -s ancien nouveau

5 Droits d'accès.

La commande **`ls -l`** permet d'avoir une liste détaillée indiquant, entre autres, les droits d'accès.

Spécial			user			group			other		
Set UID	Set GID	Sticky	r	w	x	r	w	x	r	w	x

	Fichier	Répertoire
r	Lecture autorisée	Lecture de la totalité du répertoire possible mais en absence de ce droit, on peut accéder à une entrée individuelle. Ainsi on peut lire un fichier dans un répertoire privé du droit de lecture.
w	Ecriture autorisée	On peut créer ou supprimer les fichiers du répertoire.
x	Exécution autorisée	En absence de ce droit, aucun accès au répertoire et a la sous arborescence issue du répertoire n'est possible.

	Fichier	Répertoire
Sticky Bit	Un programme exécutable sera maintenu en zone de swap après la fin de son exécution.	Un fichier du répertoire ne peut être supprimé que par son propriétaire.
Set GID	Set GID si le fichier est exécutable sinon verrouillage obligatoire.	Les fichiers (y compris les répertoires) créés dans le répertoire héritent du GID du répertoire.

Set UID	Set UID si le fichier est exécutable.	Les fichiers (y compris les répertoires) créés dans le répertoire héritent du UID du répertoire.

Normalement l'UID d'un processus est l'UID du créateur.

Si Set UID=1 alors l'UID du processus engendré à partir d'un fichier est l'UID du fichier.

Si Set UID=0 alors c'est l'UID du root (administrateur système).

Pour modifier des droits on utilise la commande chmod :

- Options :
  - `-r` descente récursive d'une sous-arborescence.
  - `-f` fonctionnement silencieux.
- Mode :
  - numérique : codage en octal de bits de permission.
  - symbolique : | u | g | o | a | + | - | = | r | w | x | S | l | t |

Exemple :

```
chmod 644 fichier
```

Autorise la lecture et l'écriture sur le fichier pour le propriétaire, mais n'autorise que la lecture pour les autres.

```
chmod 1777 répertoire
```

Autorise la lecture, l'écriture et l'exécution (ici l'accès) du répertoire pour tout le monde. Le premier chiffre correspond à l'octet du Set UID, Set GID, Sticky. Ici on met le Sticky Bit a 1.

```
chmod go-rx fichier
chmod +t répertoire # identique à u+t, g+t, o+t
chmod ug=xs programme
chmod +l fichier # identique à u+l, g+l, o+l
chmod ug=S programme # illicite
chmod g+xl programme # illicite
chmod ug+x, o-r fichier
```

6 Principales commandes utiles pour les fichiers et répertoires

6.1 Consulter le manuel

```
man [n] commande
```

Visualisation à l'écran des informations concernant la commande spécifiée. L'affichage est réalisé par un more.

Le manuel est divisé en huit sections :

- 1 : Les commandes utilisateurs
- 2 : Les appels sytèmes
- 3 : La librairie des sous-routines
- 4 : Les formats de fichiers
- 5 : Les fichiers spéciaux
- 7 : Les possibilités diverses
- 8 ou 1m : Les commandes d'administrations système
- 9 : glossaire

On peut spécifier la section dans laquelle on veut effectuer la recherche (grâce au paramètre n).

6.2 Identifier les utilisateurs du système

`who`

Fournit de informations sur l'ensemble des utilisateurs qui sont actuellement connectés sur la station.

`who am i`

Renvoie uniquement les informations relatives à l'utilisateur courant.

`whoami`

Renvoie l'identificateur de l'utilisateur courant.

`id`

Renvoie l'UID (user identifier), le GID (Groupe identifier) de l'utilisateur courant.

Il ne faut pas confondre who am i (cas particulier de la commande who) et whoami. Le premier donne des informations sur l'utilisateur connecté et le second l'identificateur de l'utilisateur courant.

6.3 Changer de mot de passe

Pour se connecter, il faut :

- un login (identificateur de l'utilisateur) assigné par votre administrateur système
- un password (mot de passe) propre à chaque utilisateur

`passwd`

Permet de définir et de contrôler son mot de passe.

A l'appel de cette commande, vous devez saisir l'ancien mot de passe, puis vous devez saisir deux fois votre nouveau mot de passe.

6.4 Afficher une chaine de caractères

echo chaine

Affiche la chaine passée en paramètre .(vous pouvez aussi afficher des variables: echo $PATH, pour visualiser la variable PATH)

banner chaine

Affiche la chaine passée en paramètre avec des grosses lettres

6.5 Commandes et manipulation des répertoires

Visualiser le contenu d'un répertoire

ls [-FaRl]

- -F : Positionne à la fin des noms un / pour les répertoires et un * pour les fichiers exécutables
- -a : Affiche tous les fichiers, y compris les fichiers cachés (ceux qui commencent par .)
- -R : Affichage récursif
- -l : Description complète du contenu d'un répertoire (une ligne par fichier). Le premier caractère de la ligne indique le type du fichier :
  - - : standard
  - d : répertoire
- -d : Evite de lister le contenu d'un répertoire : si rep est un repertoire, ls -l rep listera le contenu du répertoire rep, alors que ls -ld rep listera la description du répertoire

6.6 Changer de répertoire

cd chemin

Change le répertoire courant pour celui spécifié par le chemin.

cd

Change le répertoire courant pour le home directory.

cd ..

Change le répertoire courant pour le répertoire parent.

cd -

Change le répertoire courant pour le répertoire précédent.

pwd

(print working directory) affiche le chemin du répertoire courant.

6.7 Créer et détruire un répertoire

mkdir répertoire

Création d'un répertoire contenant les deux fichiers . et ..

rmdir répertoire

Supprime un répertoire vide (pour supprimer un répertoire non vide, il faut utiliser la commande **rm -r** (attention, danger !)

6.8 Commandes et manipulation des fichiers

Visualiser le contenu d'un fichier

cat fich1 fich2

Concatène et affiche (sur la sortie standard) le contenu des fichiers.

more fich

Visualise le contenu du ou des fichiers par page.

Pour un fichier contenant plus d'une page :

- q ou Q : pour terminer la visualisation
- RETURN : pour visualiser une ligne supplémentaire
- ESPACE : pour visualiser la page suivante
- h : pour obtenir de l'aide

6.9 Imprimer un fichier

lp [-dimp] fichiers

Imprime le ou les fichiers spécifiés sur l'imprimante par défaut ou sur celle spécifiée par l'option -d (attention, pas de blanc entre l'option et le nom de l'imprimante).

lpq

Visualise la file d'impression courante

lpstat [-t]

Renvoie des informations sur l'état de l'imprimante par défaut et de sa queue d'impression (l'option -t permet de visualiser toutes les imprimantes).

cancel num_impression

Détruit l'impression désignée par num_impression (vous récupérez ce numéro par la commande lpq ou lpstat).

6.10 Chercher une chaine de caractères dans un fichier

grep [-iv] expression fichiers

- sans option : recherche dans les fichiers les lignes contenant l'expression
- -i : pour ne pas tenir compte des majuscules/minuscules
- -v : pour afficher les lignes ne contenant pas l'expression spécifiée.

6.11 Faire une copie d'un fichier

`cp source destination`

Copie le fichier source dans le fichier destination.

Si le fichier destination n'existe pas, il est créé . Sinon son contenu est écrasé sans avertissement.

Si la destination est un répertoire, alors la source peut être une liste de fichiers.

6.12 Déplacer ou changer le nom d'un fichier

`mv source destination`

Renomme ou déplace le fichier source en destination. Si la destination est un répertoire, alors la source peut être une liste de fichiers.

6.13 Identifier le type d'un fichier

`file fichiers`

Détermine le type du ou des fichiers spécifiés.

6.14 Créer un lien sur un fichier

`ln [-s] source destination`

Création un lien sur un fichier ou un répertoire. Un lien est un moyen d'accéder à un même fichier ou un répertoire sous plusieurs noms ou à partir de plusieurs répertoires. Attention un lien n'est pas une copie : si vous modifiez le fichier alors tous les liens sur ce fichier seront modifiés.

Il existe deux sortes de liens: le lien physique et le lien symbolique (avec l'option -s). Le lien physique ne peut adresser que des fichiers, alors que le lien symbolique peut aussi lier des répertoires.

description différents liens

Dans le cas de lien physique, pour effacer le fichier, vous devez effacer tous les liens qui pointent sur ce fichier. Par contre pour des liens symboliques, vous pouvez effacer le fichier sans effacer les liens, mais alors ceux-ci seront invalides.

6.15 Détruire un fichier

`rm [-irf] fichiers`

Efface les fichiers(attention, on ne peut pas récupérer un fichier qui a été effacé)

- -i : interactif, demande une confirmation sur chaque fichier
- -f : force la suppression du fichier

- -r : récursivité : permet d'effacer un répertoire et son contenu

6.16 Rechercher un fichier

find rep -name nom -print

Recherche le(s) fichier(s) caractérisé par name (vous pouvez utiliser une expression régulière), à partir du répertoire rep et affiche le résultat.

Vous pouvez décrire le fichier à rechercher par une expression régulière, ou indiquer le type de fichiers à chercher ou encore le propriétaire…

Vous pouvez aussi exécuter d'autres actions, comme effacer le fichiers... (pour plus de détails, voir le man)

6.17 Permission sur les fichiers

Pour chaque fichier, il y a trois classes d'utilisateurs :

- utilisateur : le propriétaire du fichier
- groupe : le groupe auquel appartient le fichier
- autre : tous les autres

Les permissions accordées à ces trois classes sont :

- r : lecture
- w : écriture
- x : exécution (pour un fichier, peut être exécuté, pour un répertoire, peut devenir répertoire courant)

chmod mode fichiers

Change les permissions du ou des fichiers/répertoires.

Exemple mode désiré : rwxr-xr-- :

user	group	other	
rwx	r-x	r--	
111	101	100	en binaire
7	5	4	

d'où la commande chmod 754 fichier.

chown proprietaire fichiers

Change le propriétaire du fichier. Le nouveau propriétaire doit être connu du système.

chgrp groupe fichiers

Change le groupe du fichier. Le nouveau groupe doit être connu du système.