Fichiers I : arborescence des répertoires

Fichiers : plan

On va observer un modèle en 4 couches

- arborescence

  `ls -ilah, cd, cp -lar, rm -rf, mv, dd, tar -cxzvf, pwd -P,`
  `mkdir -p, rmdir, ln -s, readlink, file, du, find, cat, touch`

- systèmes de fichiers

  `mkfs -t, fsck, mount, umount, findmnt, stat, df`

- périphériques bloc (disque, partition, boucle, chiffrement, lvm, raid)

  `(g)parted, lsblk, cryptsetup, lvm, mdadm, losetup, dmsetup,`

- matériel

Fichiers : arborescence

Arborescence de répertoires

Les fichiers sont organisés dans une arborescence de répertoires.

Filesystem Hierarchy Standard (FHS)

Elle définit le nom et le contenu des principaux répertoires.

```
$ man 7 hier
$ 7 file-hierarchy
```

Notations et raccourcis

Notations:

- `/` désigne le répertoire racine
- `.` désigne le répertoire courant
- `..` désigne le répertoire parent du répertoire courant (`/.. = /`)

Notations dépendant du shell courant :

- `~` désigne le répertoire maison de l'utilisateur courant. Souvent `/home/<username>`. Il est contrôlé par la variable d'environnement `HOME` et défini dans `/etc/passwd`.
- `-` désigne le répertoire précédemment visité (variable `OLDPWD`, uniquement pour la commande `cd`)

Chemin absolu, chemin relatif

Fichiers cachés

La commande `ls` et les gestionnaires de fichiers graphiques ne montrent pas les fichiers dont le nom commence par un point. L'objectif initial était de ne pas montrer les répertoires `.` et `..` qui sont toujours présents.

Mais cette utilisation a été détournée pour rendre plus lisible le répertoire utilisateur, en cachant l'historique des commandes bash (`.bash_history`), des fichiers de cache (`.cache`) ou de configuration (`.bashrc`, `.vimrc`, `.gitconfig`), etc.

L'option `-a` permet à `ls` de ne pas ignorer ces fichiers.

```
$ man ls
```

Presque tout est fichier

Les fichiers sont une interface qui permet de modéliser de façon uniforme le fait de lire, écrire, dire qui a le droit d'accéder, ... à une ressource. Cette ressource n'est pas nécéssairement une suite d'octets inscrits sur une mémoire de masse (disque dur, clef USB, etc).

Démo.

Types de fichiers

- `-` : fichiers réguliers
- `d` : répertoires (en: directories)
- `l` : liens symboliques (en: symlinks)
- `p` : tubes nommés (en: named pipes)
- `s` : socket
- `b` : périphériques en mode bloc (en: block device)
- `c` : périphériques en mode caractère (en: character device)

Fichiers réguliers

Les fichiers réguliers ne sont pas typés, le filesystem n'a nulle part où indiquer l'utilisation prévue d'un fichier ou son *format* : vidéo (formats `avi`, `mpeg`, `webm`, ...), image (formats `jpeg`, `png`, `gif`, `svg`, ...) exécutable, document (formats `odt`, `pdf`, ...), archive compressée (formats `tar.gz`, `zip`, ...), fichier texte (encodage `ASCII`, `UTF-8`, `latin-1`, ...).

Certaines applications utilisent l'*extension* du nom de fichier pour s'y retrouver rapidement, d'autres explorent le début du fichier pour deviner son utilisation possible ("magic number"):

```
$ man file
```

Les fichiers texte ne spécifient pas leur encodage, donc c'est au programme qui l'utilise de le déterminer comme il peut. Pour changer l'encodage d'un fichier texte:

```
$ man iconv
```

Liens symboliques

(en: symlink)

C'est l'analogue des raccourcis sous Windows.

C'est un fichier particulier qui contient une référence à un chemin, absolu ou relatif. Lorsqu'on y accède, l'OS accède au fichier ou répertoire représenté par le chemin ("déréférencement"), si celui-ci existe (lorsqu'il n'existe pas, on parle de lien cassé (en: broken link)).

L'option `--no-dereference` de certaines commandes permet d'appliquer la commande au lien symbolique lui-même et non pas à sa référence.

Utilisé comme un « interrupteur » pour activer ou desactiver des configurations.

Démo.

Tubes nommés et sockets

- Ils permettent aux processus de communiquer entre eux (dans les deux sens pour les sockets, dans un seul sens pour les tubes).
- On les trouve en général dans `/run`, mais comme chaque programme fait ce qu'il veut (contrairement au périphériques qui sont gérés par `udev` dans un système de fichiers spécial monté dans `/dev`), on peut parfois en trouver ailleurs (`/tmp` voire dans le `HOME` de l'user qui a lancé le programme).
- Voir les cours d'architecture et de système et réseaux.

Démo.

Périphériques

Les fichiers permettant d'interagir avec les périphériques se trouvent dans `/dev`.

Exception : les périphériques réseau n'ont pas de fichiers leurs correspondant (pas de `/dev/eth0` ou de `/dev/wlan0`).

Remarque: quand on exécute la commande `ls -l`, la colonne correspondant à la taille du fichier est remplacée par deux nombres :

- majeur indique quel pilote (en: driver) utilise le noyau pour manipuler (en: handle) le périphérique. La correspondance est écrite dans `/proc/devices`. Une liste étendue se trouve sur : https://www.kernel.org/doc/html/latest/admin-guide/devices.html
- mineur identifie le périphérique, par rapport aux autres du même majeur.

Périphériques en mode caractère

Un *périphérique en mode caractère* (en:*caracter device*) est un périphérique auquel on accède octet par octet.

Ces fichiers sont des interfaces avec des périphériques qui reçoivent ou émettent des flux de données (souris, camera, terminal, etc), mais aussi la mémoire vive.

Exemples (démo):

```
$ cat /dev/input/mouse0 | hexdump

$ echo Hello > /dev/tty1

$ echo Hello > /dev/pts/<num>

$ vlc v4l2:///dev/video0
```

Périphériques en mode caractère

Il y a aussi certains fichiers spéciaux au comportement étrange, mais bien utiles:

```
/dev/zero
/dev/null
/dev/full
/dev/random
/dev/urandom
```

Voir la section 4 du manuel, pour la liste:

```
$ man 4 [TAB][TAB]
```

Périphériques en mode bloc

Les périphériques en mode bloc correspondent aux périphériques de stockage (et à leur dérivés), et seront traités spécifiquement.

Stock vs Flux

Le tour du proprio

Demo.