Introduction à l'informatique

Les fichiers

G. Santini, J.-C. Dubacq

IUT de Villetaneuse

S1 2016

Plan

Plan

Un fichier

De l'information au stockage

Les informations utilisées dans un ordinateur sont stockées dans la *mémoire de masse*, qui se distingue de la *mémoire vive* par sa résistance à l'extinction et de la *mémoire morte* (et plus tard, du *firmware*) par sa mutabilité.
Les performances des systèmes de stockage de masse sont meilleures chaques années, mais l'ordre de grandeur reste la ms ou 100 μs.

De l'information au fichier

L'information est découpée en petites unités qui s'appellent des fichiers, sémantiquement cohérentes — ce sont des informations qui « vont ensemble ». Ces éléments de base du stockage informatique peuvent ne contenir que très peu d'information ou représenter plusieurs Go de données par fichier.
Un fichier est lié à la façon dont on y accède (son *nom* et son *chemin*), mais nous verrons que ce n'est pas un identifiant : il peut y avoir plusieurs accès différents à un même fichier (*liens*).

Noms et contenu des fichiers

La décomposition traditionnelle d'un nom de fichier

Deux parties séparées par un point :

- La 1ère partie informe sur la nature du contenu du fichier,
- La 2ème partie informe sur le format ou la finalité des données.

nom.extension
prefix.suffix
description.format

Selon les systèmes, certains caractères sont interdits. Par exemple ∗ sous Windows, / sous Linux.

Exemples de noms de fichiers

Extension	Contenu
.c	Sources C
.html	Document Web
.pdf	Document Mis en page
.txt	Texte brut

Enigmatique	Informatif
e3.c	teste_boucle_for.c
New.pdf	2011_IntroSys_cours_1.pdf
toto.sh	test_boucle_for.sh

Choix des noms

Ils doivent être choisis minutieusement pour être informatifs.

Choisir un nom : réfléchir pour un gain de temps pour retrouver le fichier ou le répertoire concerné.

Importance de la casse (Linux), tolérance ailleurs (OS X, Windows).

Plan

Des fichiers et des répertoires

Les fichiers... en vrac ?

Les fichiers sont regroupés dans des répertoires (en anglais *directory* ou *folders*). Les répertoires peuvent contenir des fichiers ou d'autres répertoires. L'organisation des fichiers est réglée par le *système de fichiers* (ang. *filesystem*).

- Cette organisation arborescente permet de faciliter la recherche d'un fichier,
- Les fichiers sont regroupés par application, par thème, par format, par fonction, ...
- Organisation *hiérarchique* qui permet d'organiser les données et de faciliter leur accès.

De très nombreux fichiers et répertoires

Le nombre de fichiers enregistrés sur un disque dur peut aisément dépasser 100.000 fichiers,

- Dans un même répertoire le nom est un identifiant.
- Les répertoires et les fichiers partagent les mêmes noms.

Sous Windows, pas d'extension pour les répertoires.

Remarque

Avec tous les fichiers au même *endroit*, il est très difficile de les lister (trop à lire).

Plan

Exemple d'arborescence Linux

- / — Répertoire racine *(Root Directory)*
  - bin/
    - (…)
  - home/
    - moi/ — Répertoire personnel *(User directory)*
      - Mes Documents/
        - ListeDesCourses.txt
        - Exercice_1.sh
      - (…)
    - anonymous/
      - LisezMoi.txt
      - Telechargements/
        - (…)
  - (…)

Les répertoires importants

- La **racine** (*Root directory*) contient tous les répertoires et fichiers accessibles depuis le système.
- Le **répertoire personnel** (*User Directory* ou *Home Directory*) est le répertoire dans lequel l'utilisateur peut faire ce qu'il veut (écrire, modifier, supprimer, installer …).

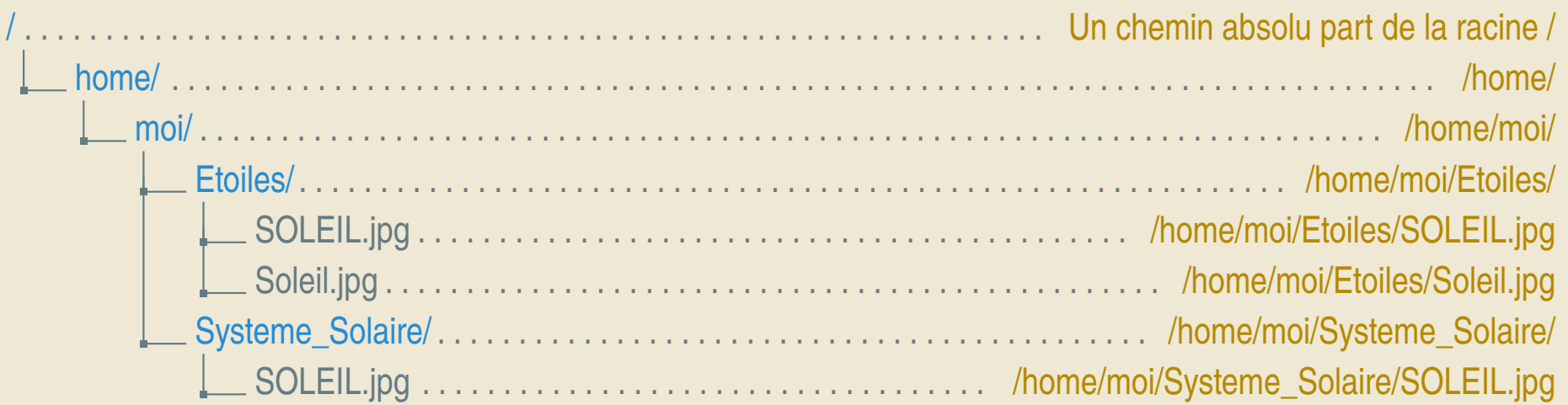
Plan

La notion de chemin

Le chemin définit un accès unique à partir de la racine

- Deux fichiers ou répertoires ne peuvent pas porter le même nom si ils sont dans un même répertoire.
- Sous Linux, les noms des fichiers et répertoires différencient les caractères MAJUSCULES et minuscule. Les fichiers Essai.txt et essai.txt peuvent donc être dans le même répertoire.

Exemples de chemins absolus



Syntaxe d'un chemin absolu

Le chemin *absolu* d'un élément du système de fichier est unique (sauf avec un *lien*). Il donne la liste des répertoires et sous-répertoires en partant de la racine `/` (la référence de l'arborescence) jusqu'à la cible.

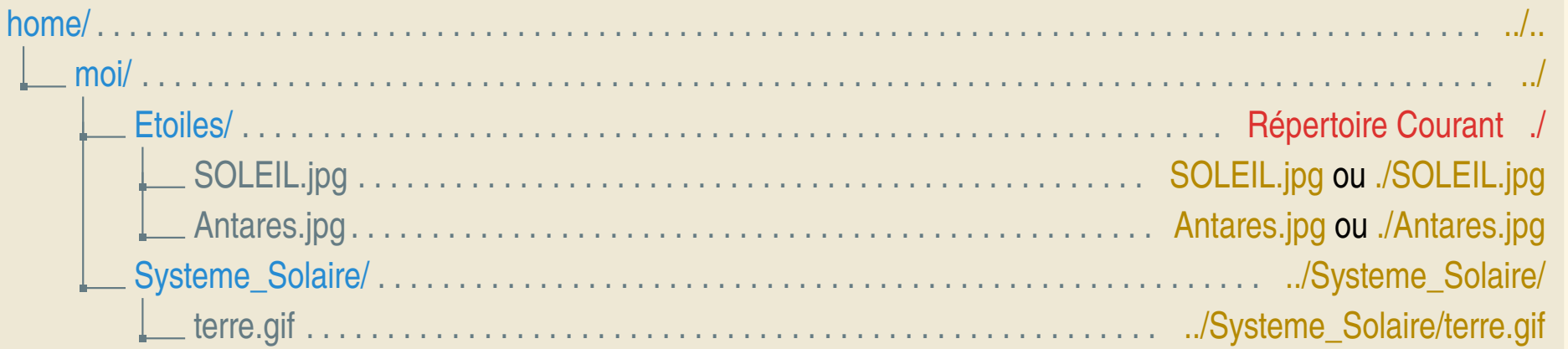
Plan

Répertoire courant et chemins relatifs

Le répertoire courant

- Le répertoire courant est un répertoire de référence d'où sont lancées les commandes du shell.
- Par défaut, le répertoire courant est le répertoire personnel de l'utilisateur,
- Naviguer dans l'arborescence équivaut à modifier le répertoire courant.

Exemples de chemins relatifs



Syntaxe d'un chemin relatif

- Le chemin *relatif* d'un fichier ou d'un répertoire donne la liste des répertoires et sous-répertoires en partant du répertoire courant (la référence *relative* dans l'arborescence) jusqu'à la cible.
- Il est relatif, car lorsque le répertoire courant change, le chemin relatif change.

Plan

Notation spéciales

Les chemins des répertoires de référence

Répertoire	Notation
Répertoire racine	/
Répertoire personnel	~

Répertoire	Notation
Répertoire courant	.
Répertoire parent	..

La notation ~ est un chemin absolu, remplacée par le vrai chemin avant l'exécution des commandes. C'est un raccourci *au niveau du shell, pas au niveau du système d'exploitation*.

Exemple de chemins valides pointant le fichier cible

/ ..... Répertoire Racine
└── home/
    └── moi/ ..... Répertoire Personnel
        └── Etoiles/ ..... Répertoire Courant
            └── Soleil.jpg ..... Fichier cible

Chemins Absolus
`/home/moi/Etoiles/Soleil.jpg`
`~/Etoiles/Soleil.jpg`
`/home/moi/../moi/Etoiles/Soleil.jpg`
`/home/moi/../../home/moi/Etoiles/Soleil.jpg`
Chemins Relatifs
`Soleil.jpg`
`./Soleil.jpg`
`../Etoiles/Soleil.jpg`
`../../moi/Etoiles/./Soleil.jpg`

L'archivage

D'une arborescence à un fichier

Une technique souvent utilisée consiste à transformer une partie de l'arborescence en un fichier qui n'est pas utilisable directement. Ce fichier peut ensuite être retransformé en une arborescence.

Le format tar

Utilisé depuis les années 80, le format tar est un pilier du monde Unix. Il est parfaitement libre. Il servait initialement aux sauvegardes sur bande magnétique (*t*ape *ar*chive).
Le format tar ne permet pas la compression, mais la commande `tar` donne accès à des programmes de compression qui permettent de réduire la taille de l'archive. Une archive au format tar est appelée un(e) *tarball*.
Le compresseur le plus connu est `gzip` dont les fichiers compressés ont un suffixe `.gz`.
Souvent on combine les deux suffixes : une archive compressée peut ainsi s'appeler `textes2015.tar.gz` ou `textes2015.tgz`.

Le format zip

Principalement utilisé pour son universalité depuis 1986, le format zip est plus ou moins libre (il y a des doutes sur la possibilité de brevet sur les techniques employées). Le format zip n'est pas uniquement caractérisé par son extension : plusieurs autres formats de fichier sont en fait une archive ZIP qui contient divers documents (par exemple, un fichier `docx` pour Microsoft Word est en fait un ZIP qui contient divers fichiers XML et images).
Le format zip, en plus de l'archivage permet aussi la compression. La commande `zip`/`unzip` doit donc permettre la décompression.

Plan

Conventions

Noms et chemins

- Un chemin peut être absolu ou relatif. Il peut utiliser les notations spéciales.
- Par convention la notion de fichier sera comprise dans son sens large. Par exemple, le chemin d'un fichier devra être interprété sans distinction comme le chemin vers un fichier ordinaire ou comme le chemin vers un répertoire (sauf mention contraire explicite).

Commandes, options, paramètres

Commande c'est le nom d'un programme qui exécute une action.

Options ce sont des paramètres optionnels. Ils peuvent être omis. L'ajout d'options modifie le comportement de la commande (le résultat). Les options sont montrées encadrées par les caractères `[ ... ]` (qu'il ne faut pas mettre).

Paramètres ce sont des arguments que la commande évalue.

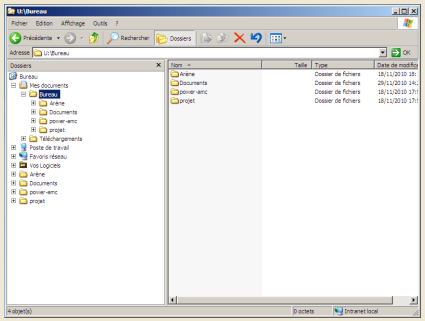
Sources et destination

Les commandes de déplacement acceptent une ou des *sources* qui sont des fichiers ou répertoires d'origine, et une *destination* qui est un fichier ou un répertoire.

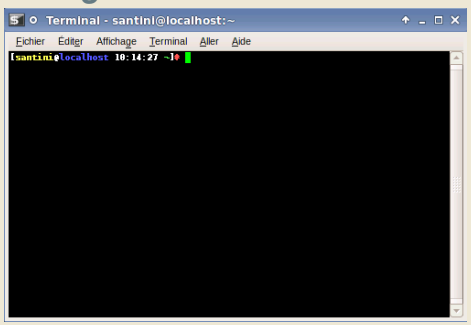
Manipulation de l'arborescence en ligne de commande

Alternatives pour naviguer dans l'arborescence et manipuler les fichiers

Interface Graphique



Ligne de Commande



Boîte à outils : manipuler l'arborescence

Commande	Fonction principale
`pwd`	Afficher le nom du répertoire courant
`cd`	Changer de répertoire courant
`ls`	Afficher le contenu d'un répertoire
`cat`	Afficher le contenu d'un fichier
`touch`	Créer un fichier
`mkdir`	Créer un répertoire
`rm`	Supprimer fichier(s) ou répertoire(s)
`cp`	Copier fichier(s) ou répertoire(s)
`mv`	Déplacer/Renommer fichier(s) ou répertoire(s)

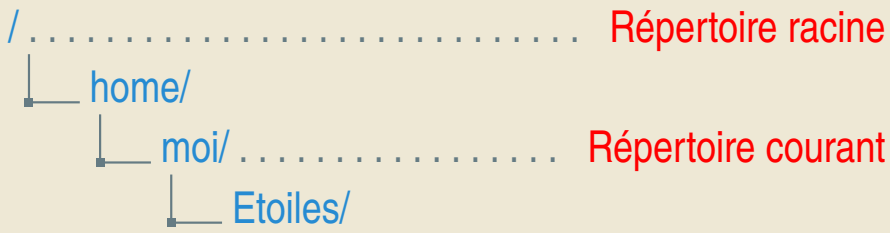
Syntaxe pour pwd

pwd

Description

- Affiche le nom du répertoire courant.

Exemple d'utilisation:




```
login@host:~$ pwd
/home/moi
```


```
login@host:~/Etoiles$ pwd
/home/moi/Etoiles
```

Syntaxe pour cd

```
cd <cible>
```

Description

- Change le répertoire courant (permet de naviguer dans l'arborescence).
- Si le chemin du répertoire cible est omis, le répertoire courant redevient par défaut le *répertoire personnel*.

Ce n'est pas une commande, mais une fonctionnalité du shell.

Exemple d'utilisation:

```
/ ..... Répertoire Racine
└── home/ ..... Répertoire courant initial
    └── moi/ ..... Répertoire courant cas #1
        └── Etoiles/ ..... Répertoire courant cas #2
```

Commande cas #1 :

```
login@host:/home$ cd
login@host:~$ ■
```

Commande cas #2 :

```
login@host:/home$ cd moi/Etoiles
login@host:~/Etoile$ ■
```

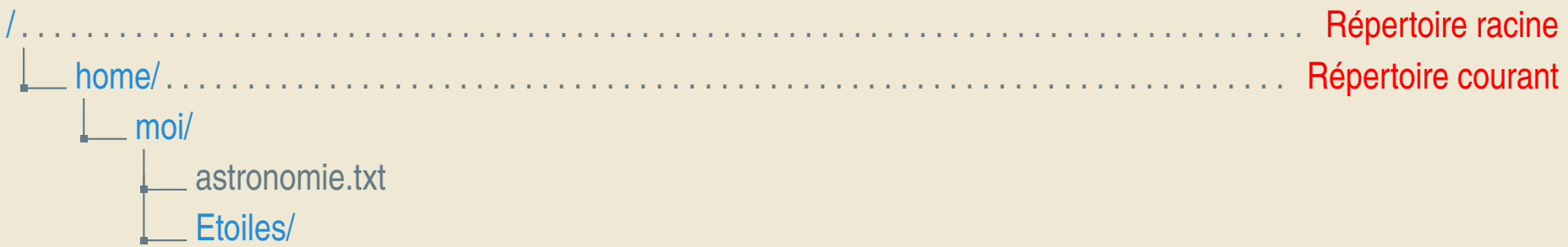
Syntaxe pour ls

```
ls  <source>
```

Description

- Affiche le contenu d'un répertoire.
- Par défaut si aucune source n'est indiquée, la commande affiche le contenu du répertoire courant.

Exemple d'utilisation:




```
login@host:/home/$ ls
moi/
```

```
login@host:/home/$ ls moi/
Etoiles/ astronomie.txt
```

Syntaxe pour ls(bis)

```
ls -a <source>
```

Description

- Affiche le contenu d'un répertoire y compris les fichiers et répertoires cachés.
- Les fichiers et répertoires cachés ont un nom dont le premier caractère est un point.
- Les fichiers et répertoires cachés sont utilisés par le système ou certaines applications.

Exemple d'utilisation:

moi/ . . . . . Répertoire courant

- ./ssh/
  - id_rsa
  - id_rsa.pub
  - known_hosts
- .bashrc
- astronomie.txt
- Etoiles/
  - soleil.jpg

Sans option *-a*

```
login@host:~$ ls
astronomie.txt  Etoiles/
```

Avec option *-a*

```
login@host:~$ ls -a
.          ..          .bashrc .ssh/
astronomie.txt  Etoiles/
```

Syntaxe pour cat

```
cat  fichier [fichier_2 ...]
```

Description

- Affiche le contenu des fichiers les uns à la suite des autres.
- Les fichiers sont concaténés dans l'ordre des paramètres.

Exemple d'utilisation:

Cette commande est en générale utilisée pour concaténer des fichiers textes. On l'utilise avec une commande de redirection (cf. Partie Redirections) pour enregistrer le résultat de la concaténation dans un nouveau fichier.

Soient les deux fichiers suivants :

tellur.txt

```
Mercure, Venus
Terre, Mars
```

jov.txt

```
Jupiter, Saturne
Uranus, Neptune
```

La commande :

```
login@host:~$ cat tellur.txt jov.txt
Mercure, Venus
Terre, Mars
Jupiter, Saturne
Uranus, Neptune
login@host:~$
```

Syntaxe pour touch

```
touch  chemin [chemin_2 ...]
```

Description

- Si le chemin est occupé par un fichier ou un répertoire, mise à jour de la date de dernière modification.
- Sinon, création d'un ou de plusieurs fichiers vides à l'endroit spécifié par le chemin.

Exemple d'utilisation:

```
moi/ ..... Répertoire Courant
├── astronomie.txt
├── lisezmoi.txt ..... Création Commande #1
└── Stars/
    ├── TCeti.txt ..... Création Commande #2
    └── ACentauri.txt ..... Création Commande #2
```

```
login@host:~$ touch lisezmoi.txt
login@host:~$ touch Stars/TCeti.txt Stars/ACentauri.txt
```

Syntaxe pour mkdir

```
mkdir  chemin [chemin_2 ...]
```

Description

- Création d'un ou de plusieurs répertoires aux endroits spécifiés par les chemins.
- Si le chemin est occupé par un fichier ou un répertoire, il y a un message d'erreur.
- Si le chemin n'est pas déjà créé à part le dernier élément, il y a un message d'erreur.

Exemple d'utilisation:

```
moi/ ..... Répertoire courant
├── Systeme_Solaire/ ..... Création commande #1
└── Etoiles/
    ├── Rouges/ ..... Création commande #2
    └── Bleues/ ..... Création commande #2
```

```
login@host:~$ mkdir Systeme_Solaire
login@host:~$ mkdir Etoiles/Rouges Etoiles/Bleues
login@host:~$ mkdir Galaxies/M91
mkdir: impossible de créer le répertoire
« Galaxies/M91 »: Aucun fichier ou dossier de ce type
```

Syntaxe pour mkdir(bis)

```
mkdir -p chemin <chemin_2 ...>
```

Description

- Création d'un ou de plusieurs répertoires aux endroits spécifiés par les chemins.
- Si depuis la racine en suivant un chemin, on rencontre un fichier, il y a un message d'erreur.
- Si depuis la racine en suivant un chemin, il n'y pas de répertoire, il est créé.

Exemple d'utilisation:

chez_moi/ ..... Répertoire Courant
- astronomie.txt
- Etoiles/
- Galaxies/ ..... Création Commande #1
  - M91/ ..... Création Commande #1
    - highres/ ..... Création Commande #1

```
login@host:~$ mkdir -p Galaxies/M91/highres
```

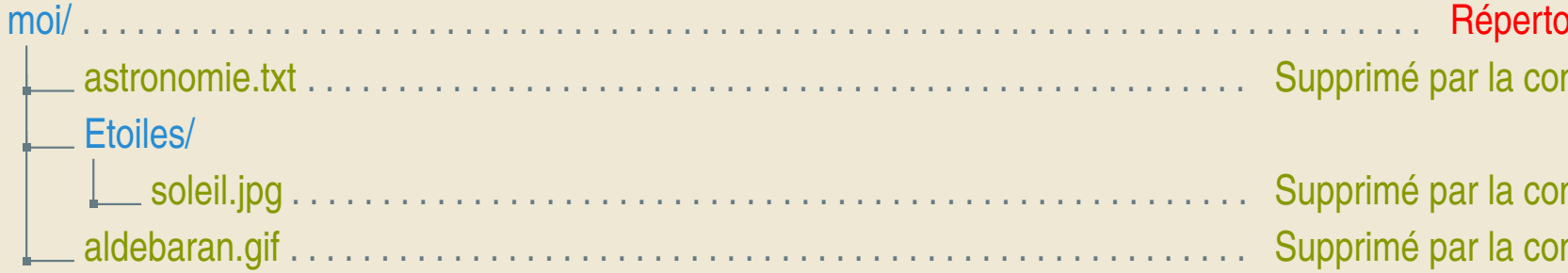
Syntaxe pour rm

```
rm  chemin [chemin_2 ...]
```

Description

- La commande supprime le fichier pointé par le(s) chemin(s).
- Si le chemin pointe sur un répertoire, la commande affiche un message d'erreur.

Exemple d'utilisation:




```
login@host:~$ rm astronomie.txt
login@host:~$ rm aldebaran.gif Etoiles/soleil.jpg
```

Syntaxe pour rm(bis)

```
rm -r chemin <chemin_2 ...>
```

Description

- L'option `-r` (comme **r**écursif) permet de supprimer un répertoire et tout son contenu.

L'option `-f` (comme **f**orce) permet d'ignorer certaines questions.

Exemple d'utilisation:

```
chez_moi/ ..... Répertoire Courant
├── astronomie.txt
├── Etoiles/ ..... Supprimé par la commande #1
│   ├── soleil.jpg ..... Supprimé par la commande #1
│   └── Galaxie/ ..... Supprimé par la commande #1
│       └── Andromede.pdf ..... Supprimé par la commande #1
└── aldebaran.gif
```

```
login@host:~$ rm -r Etoiles
```

Syntaxe pour cp

```
cp  source cible
```

Description

- Copie le fichier source vers la cible.
- La source doit être un fichier ordinaire (pas un répertoire),
- Si la source est un répertoire la commande produit un message d'erreur.
- Si la cible :
  - est le chemin d'un répertoire existant, le fichier sera copié dans ce répertoire et conservera son nom,
  - ne correspond pas à un répertoire existant, le fichier sera copié avec le nom (chemin) `cible`.

Exemple d'utilisation:

```
moi/ ..... Répertoire courant
├── astronomie.txt ..... Fichier source commandes #1 et #2
├── Etoiles/ ..... Répertoire cible commande #1
│   ├── astronomie.txt ..... Copié/Créé par la commande #1
│   └── info.txt ..... copié/créé par la commande #2
└── cv.pdf
```

```
login@host:~$ cp astronomie.txt Etoiles
login@host:~$ cp astronomie.txt Etoiles/info.txt
```

Syntaxe pour cp(bis)

```
cp  source <source_2 ...> cible
```

Description

- Copie plusieurs fichiers sources vers la cible.
- Les sources doivent être des fichiers ordinaires, et la cible un répertoire.

Exemple d'utilisation:

moi/ . . . . . Répertoire courant
- cv.pdf . . . . . Fichier source
- motivations.pdf . . . . . Fichier source
- Candidature/ . . . . . Répertoire cible
  - cv.pdf . . . . . Copié/créé par la commande
  - motivations.pdf . . . . . Copié/créé par la commande

```
login@host:~$ cp cv.pdf motivations.pdf Candidature
login@host:~$ cp cv.pdf motivations.pdf Candidature/ #
Moins ambigu
```

Syntaxe pour cp(ter)

```
cp -r source <source_2 ...> cible
```

Description

- L'option `-r` (**R**écursif) permet de copier un répertoire et son contenu si il apparait dans le(s) source(s).
- Attention : si le répertoire n'existe pas et qu'on copie un répertoire, il y a renommage

Exemple d'utilisation:

```
moi/ ..... Répertoire Courant
├── Galaxie/ ..... Source commandes
│   └── Andromede.pdf
├── Etoiles/ ..... Répertoire cible #1
│   └── Galaxie/ ..... Copié/créé par la commande #1
│       └── Andromede.pdf ..... Copié/créé par la commande #1
└── Top10/ ..... Copié/créé par la commande #2 (renommage)
    └── Andromede.pdf ..... Copié/créé par la commande #2
```

```
login@host:~$ cp -r Galaxie Etoiles
login@host:~$ cp -r Galaxie Top10
```

Syntaxe pour mv

```
mv  source cible
```

Description

Déplace/renomme un fichier ou répertoire.

- modifie le chemin d'accès à la `source` qui devient le chemin `cible`.
- Le chemin `source` disparait et le chemin `cible` est créé.
- Le fichier ou répertoire pointé reste le même.
- La cible doit être un chemin non occupé ou un répertoire.

Exemple d'utilisation: Renommer un fichier

État Initial de l'arborescence :

```
moi/ ..... Répertoire courant
└── AstroNomIe.TXT ..... Fichier Source
```

État Final de l'arborescence :

```
moi/ ..... Répertoire courant
└── astronomie.txt ..... Fichier Renommé
```

```
login@host:~$ mv AstroNomIe.TXT astronomie.txt
```

Exemple d'utilisation: Déplacer un Répertoire

État Initial de l'arborescence :

```
moi/ ..... Répertoire courant
├── astronomie.txt ..... Fichier source
└── Etoiles/ ..... Répertoire cible
```

État Final de l'arborescence :

```
moi/ ..... Répertoire courant
└── Etoiles/ ..... Répertoire cible
    └── astronomie.txt ..... Fichier déplacé
```

```
login@host:~$ mv astronomie.txt Etoiles
```

Exemple d'utilisation: Renommer un répertoire

État Initial de l'arborescence :

```
moi/ ..... Répertoire courant
└── Etoiles/ ..... Répertoire Source
    └── astronomie.txt
```

État final de l'arborescence :

```
moi/ ..... Répertoire courant
└── Relativite/ ..... Répertoire Renommé
    └── astronomie.txt
```

```
login@host:~$ mv Etoiles Relativite
```

Exemple d'utilisation:

État Initial de l'arborescence :

```
moi/ ..... Répertoire courant
├── astronomie.txt ..... Fichier Source
├── relativite.pdf ..... Fichier Source
└── Etoiles/ ..... Répertoire Cible
```

État Final de l'arborescence :

```
moi/ ..... Répertoire courant
└── Etoiles/ ..... Répertoire Cible
    ├── astronomie.txt ..... Fichier Déplacé
    └── relativite.pdf ..... Fichier Déplacé
```

```
login@host:~$ mv astronomie.txt relativité.pdf Etoiles
```

Exemple d'utilisation:

État Initial de l'arborescence :

```
moi/ ..... Répertoire courant
├── relativite.pdf ..... Fichier Source
├── Etoiles/ ..... Répertoire Source
│   └── astronomie.txt
└── Espace/ ..... Répertoire Cible
```

État Final de l'arborescence :

```
moi/ ..... Répertoire courant
└── Espace/ ..... Répertoire Cible
    ├── relativite.pdf ..... Fichier Déplacé
    └── Etoiles/ ..... Répertoire Déplacé
        └── astronomie.txt ..... Fichier Déplacé
```

```
login@host:~$ mv relativité.pdf Etoiles Espace
```

Syntaxe pour tar

```
tar cvf nom_archive fichier_ou_repertoire [autres_sources]
```

Description

- Crée un fichier archive dont le nom (chemin) est donné en premier argument et porte classiquement l'extension `.tar`.
- Les fichiers sources qui servent a créer l'archive sont préservés par la commande `tar`.
- L'option `c` (comme **c**reate), indique que la commande `tar` doit utiliser un algorithme d'archivage.
- L'option `v` (**v**erbose), permet d'afficher le déroulement de l'archivage.
- L'option `f` (**f**ile), permet de préciser juste derrière un fichier d'archivage.

Exemple d'utilisation:

moi/ ..... Répertoire courant
- astronomie.txt
- Images/
  - soleil2.jpg
  - Terre1.jpg
- espace.tar ..... Créé par la commande #1

Regroupe dans la même archive `espace.tar` le fichier `astronomie.txt` et le répertoire `Images/` et son contenu :

Syntaxe pour tar(bis)

```
tar tvf nom_archive
tar xvf nom_archive [chemin(s) dans l'archive]
```

Description

- Examine une archive ou crée des fichiers à partir de l'archive.
- Le fichier archive est préservé par la commande `tar`.
- L'option `x` (e**x**tract), permet de désarchiver.
- L'option `t` (lis**t**), permet de lister le contenu d'une archive

Exemple d'utilisation:


```
login@host:~$ tar tvf espace.tar
astronomie.txt
Images/
Images/soleil2.jpg
Images/Terre1.jpg
login@host:~$ tar xvf espace.tar
astronomie.txt
astronomie.txt
login@host:~$ tar xf espace.tar
login@host:~$
```

Exercices

Préparation

Q1 Ouvrez un terminal. Vérifiez que le répertoire dans lequel vous êtes est bien `/home/usager/123456789/`. Quelle est la commande qui permet de le faire ? (123456789 = votre identifiant)

Q2 Vérifiez le contenu du répertoire `Documents` qui est dans votre répertoire personnel. Quelle est la commande qui permet de le faire ? Est-ce qu'il y a quelque chose ?

Q3 Faites la vérification de trois façons différentes : chemin absolu, utilisation du raccourci `~`, utilisation d'un chemin relatif.

Q4 Changez le répertoire courant pour aller dans `Documents`. Quelle est la commande pour le faire ?

Q5 Créez en ligne de commande un répertoire `m1101` dans `~/Documents`. À partir de maintenant, assurez-vous que le répertoire courant est ce répertoire `m1101`.

Q6 Téléchargez l'archive contenant les données pour ce TP : Allez sur la page `http://lipn.fr/~dubacq/m1101.html`. Téléchargez le fichier `photos.tar`. Recherchez où le fichier a été écrit dans l'arborescence de votre répertoire personnel.

Q7 Donnez la (suite de) commande(s) permettant de déplacer le fichier d'archive dans le répertoire `m1101` que vous venez de créer. À la fin des commandes, le répertoire `m1101` sera toujours votre répertoire courant et ne contiendra que le fichiers photos.tar.

Q8 Quelle commande permet de vérifier que l'archive est bien dans le répertoire

Exercices

Examen de fichiers

Q9 Quelles sont les informations données par le nom du fichier ?

Q10 Les commandes `less`, `cat` et `hexdump` permettent d'afficher le contenu d'un fichier. Analysez la différence de comportement entre ces deux commandes sur le fichier `photos.tar`. Qu'en concluez-vous ? Quel est le programme le plus adapté pour voir le contenu de ce fichier ?

Q11 Relisez le manuel de la commande `tar`. Vérifiez la liste des fichiers contenus dans l'archive. Combien y en a-t-il ?

Q12 Sortez les fichiers de l'archive.

Q13 Avec les commandes de la question 10, regardez le fichier contenu dans un répertoire. Analysez la différence de comportement entre ces commandes. Qu'en concluez-vous ?

Remarques : si un affichage prend trop de temps, utilisez le raccourci clavier adéquat pour suspendre l'exécution de la commande courante. Si l'affichage de votre terminal est durablement perturbé, dans le menu Terminal →Réinitialiser le terminal.

Plan

Le métacaractère ∗

Le caractère ∗

Le shell traduit la ligne de commande en `commande argument1 argument2 ...`
Avant l'exécution, il traduit certains caractères selon des règles précisées ici.

- Le cataractère ∗ est utilisé comme un *joker* pour remplacer une chaîne de caractères,
- Il est utilisé dans un chemin pour pointer plusieurs fichiers ou répertoires existants dont le chemin partage un motif commun.
- Le caractère ∗ peut être n'importe où dans le chemin, plusieurs fois si nécessaire.

Exemple de manipulation avec la commande mv

```
login@host:~$ mv *.jpg Images/
```

Ici, le chemin `*.jpg` pointe tous les fichiers du répertoire courant dont le nom se fini par l'extension `.jpg`. Il pointe donc les fichiers `etacentauri.jpg` et `aldebaran.jpg` et exclue les autres fichiers (ici le fichier `alphacentauri.gif`).

moi/ ..... Répertoire Courant
- aldebaran.jpg ..... Fichier ciblé
- alphacentauri.gif
- etacentauri.jpg ..... Fichier ciblé
- Images/ ..... Répertoire final

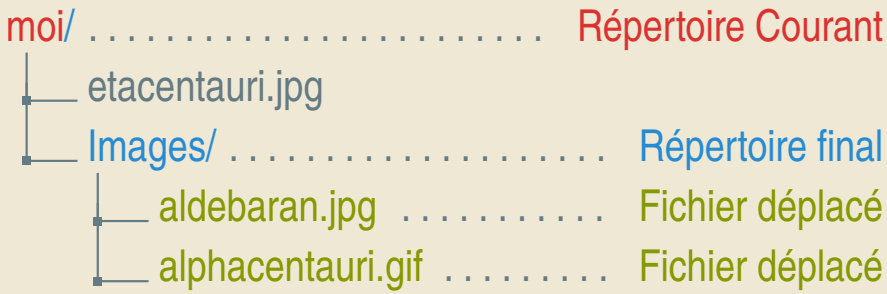
moi/ ..... Répertoire Courant
- alphacentauri.gif
- Images/ ..... Répertoire final
  - aldebaran.jpg ..... Fichier déplacé
  - etacentauri.jpg ..... Fichier déplacé

Exemples d'utilisation de l'étoile

Utilisation simple avec la commande mv

```
login@host:~$ mv al* Images/
```

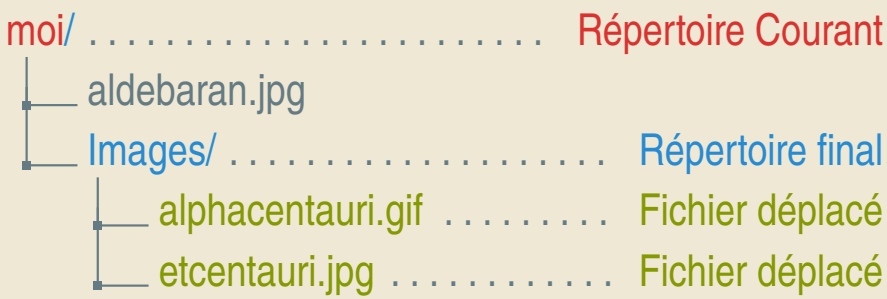
Ici, le chemin `al*` pointe tous les fichiers du répertoire courant dont le nom commence par les caractères `al`. Il pointe donc les fichiers `aldebaran.jpg` et `alphacentauri.gif` et exclue les autres fichiers (ici le fichier `etacentauri.jpg`).



Utilisation double avec la commande mv

```
login@host:~$ mv *centauri* JPG/
```

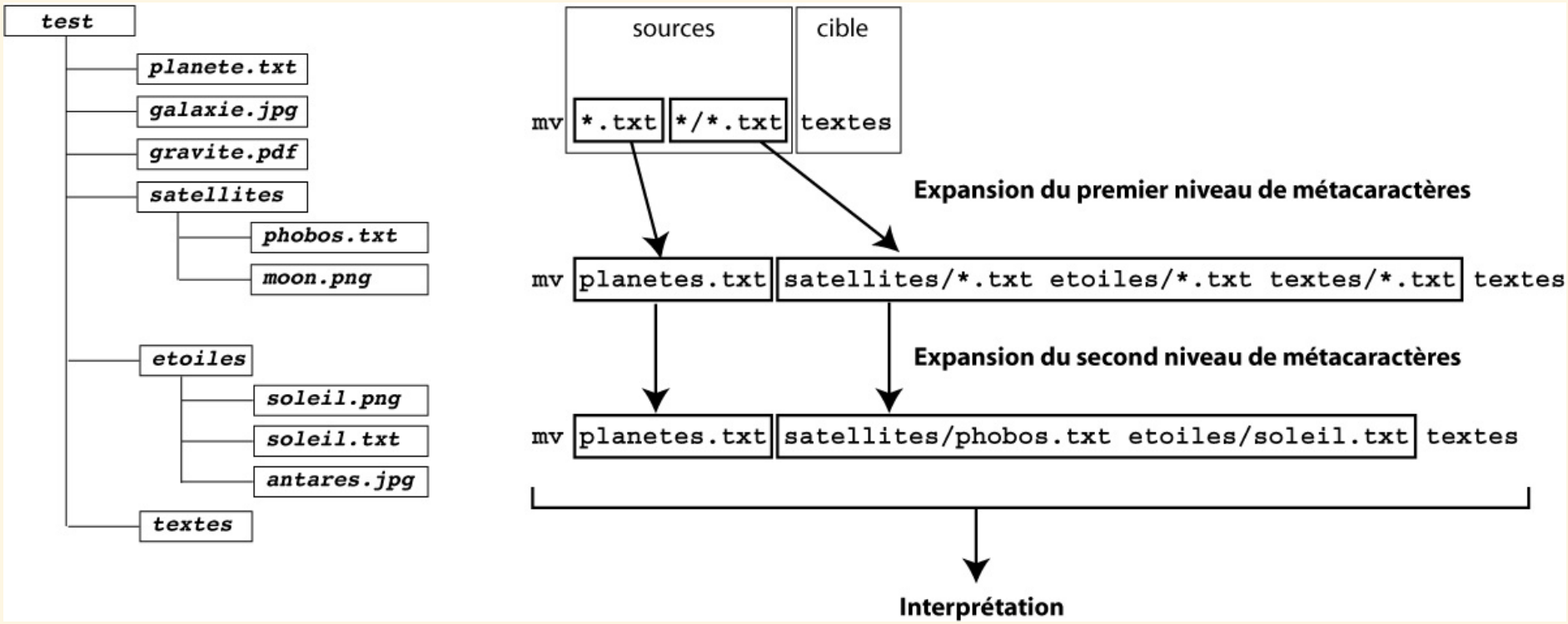
Ici, le chemin `*centauri*` pointe tous les fichiers du répertoire courant dont le nom contient la chaîne de caractères `centauri`. Il pointe donc les fichiers `alphacentauri.gif` et `etacentauri.jpg` et exclue les autres fichiers (ici le fichier `aldebaran.jpg`).



Métacaractère et chemins ciblés

Exemple plus complexe et détails de l'interprétation

- Le cararctère $*$ est développé lors de l'interprétation.



Autres métacaractères

Du shell aux programmes

Il faut bien se souvenir que les métacaractères sont interprétés par le shell. Cela a deux conséquences :

- Le programme appelé ne sait pas si les noms ont été tapés en entier ou si des métacaractères ont été utilisés. Il n'a que le résultat final.
- Dans un programme, on ne peut pas utiliser les métacaractères.

Les jokers

Ce sont des motifs simples. Lorsqu'ils ne peuvent pas être instanciés, ils ne sont pas supprimés, mais passés tels quels. Exemple : `mkdir -p toto/*` selon que `toto` est un répertoire non-vide ou autre chose.
On y trouve `?` qui remplace une lettre, `[a-c][0-2]` qui remplace a0 b0 c0 a1 b1 c1 a2 b2 c2,

Les raccourcis

Le motif `{fourch,brou}ette` est remplacé par `fourchette` et `brouette` indépendamment de l'existence ou nom de chemins correspondants.
Le motif `~` a déjà été vu et est remplacé par le chemin absolu du répertoire personnel de l'utilisateur courant. `~user` est remplacé de la même façon mais pour l'utilisateur *user*.

Exercices

Copie et déplacement

Q14 Quelle commande permet la création "simultanée" de trois répertoires `GIF` et `Photos/Portugal`, `Photos/Marseille` et `Photos/Montagne` ?

Q15 Quelle commande permet de *déplacer* depuis le répertoire `images` tous les fichiers présentant l'extension `gif` dans le répertoire `GIF` nouvellement créé ?

Q16 Quelle commande permet de *copier* depuis le répertoire `images` tous les fichiers présentant l'extension `jpg` dans le répertoire `Photos` nouvellement créés ?

Q17 Définissez le répertoire `Photos/Montagne` comme votre répertoire courant. Quelle commande permet de déplacer la photo de chalet dans ce répertoire ?

Q18 En vous mettant dans `Photos`, déplacez les photos restantes dans le bon répertoire (Marseille est supérieure à 2000). Si possible, faites usages de jokers.

Exercices

Suppressions

Q19 Quel est le résultat de la séquence de commandes suivante :

```
cd ..
rm images
```

Q20 Comment modifier la dernière commande pour supprimer le répertoire `images/` ? Comment modifier la commande pour éviter les invites de confirmation ?

Q21 Quelle commande permet de copier le répertoire `GIF` et son contenu dans un répertoire nommé `images_GIF` ?

Q22 Quelle est la différence entre les deux commandes suivantes :

```
cd ~
cd /home/usager/votre_identifiant/
```

Q23 Fabriquez une archive qui contient le répertoire Photos (et uniquement celui-ci). Vérifiez son contenu.

Plan

Le partitionnement

Du disque aux partitions

- Un disque est souvent divisé en plusieurs zones d'usage distinct (par exemple, système et données utilisateurs).
- Chacun de ces zones est appelée une *partition*. Elle est un système de fichiers indépendant des autres, et peut être combinée avec d'autres.
- Sous Windows, chaque partition est désignée par une lettre en fonction de son ordre de découverte par le système. Cette lettre fait partie du chemin.

L'ordre des partitions peut changer et donc la lettre ; ça pose problème pour les mises à jour.

Montage et démontage

- Un système d'exploitation peut rendre accessible une partition : c'est le *montage* de la partition.
- Inversement : c'est le *démontage* de la partition.

Une partition montée peut être utilisée normalement par les programmes.

Une partition démontée doit utiliser une interface spéciale plus compliquée qui contourne le *système de fichiers* et permet d'accéder directement au disque.

Les partitions sous Linux

L'arbre unique

- Sous Linux, les partitions sont toutes dans regroupés dans une seule arborescence.

Les partitions qui ne sont pas la racine sont accrochées dans la partition racine (ou une autre déjà accrochée) au niveau d'un répertoire qui sert de *point de montage*.

Le contenu du point de montage est alors inaccessible et remplacé par le contenu du système de fichier qui a été monté

- Le chemin absolu d'un élément du système monté est le chemin du point de montage suivi du chemin dans le système de fichiers monté.

Exemple : fichier `moi/toto.txt` dans un système monté sur `/home`, le chemin absolu est `/home/moi/toto.txt`.

Le pseudo-système `/dev`

Sous Linux, les périphériques sont accessibles par une interface de type fichier. Leur chemin est `/dev/codeperipherique`.
Le sous-arbre à partir de `/dev` est un système de fichiers indépendant d'un périphérique physique. On parle de *système de fichiers virtuel*.

Syntaxe pour mount

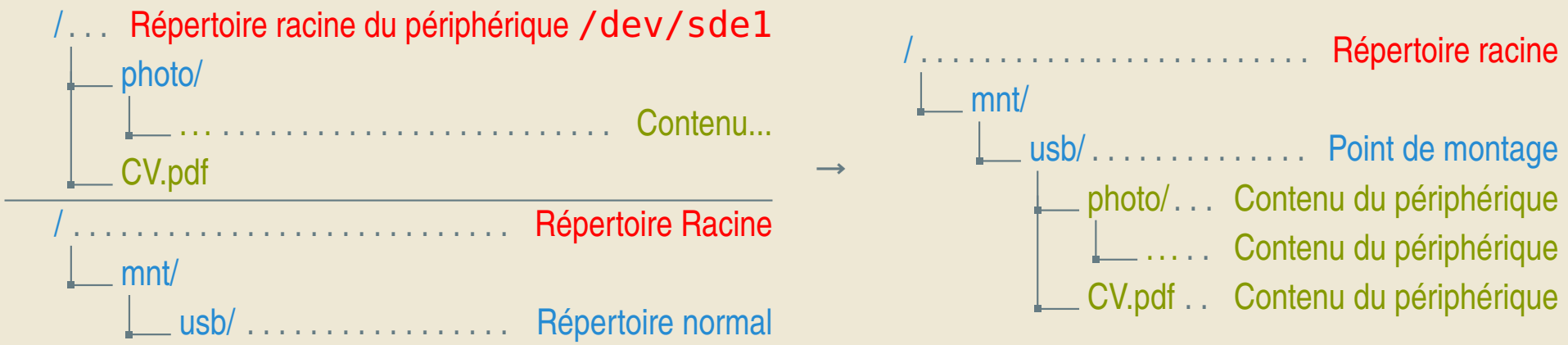
```
mount  [[périphérique] point_de_montage]
```

Description

- `périphérique` correspond à périphérique (`/dev/xxx`). Il y a plusieurs syntaxes possibles.
- `point_de_montage` correspond à un nom de répertoire valide dans l'arborescence principale donnant accès au contenu de l'arborescence du périphérique.
- Sans argument, la commande liste tous les montages en cours

Exemple d'utilisation:

```
login@host:/home$ mount /dev/sde1 /mnt/usb
```



Exercices

Analyse de périphériques

Q24 Le périphérique `zero` est un périphérique virtuel. La commande `dd if=/dev/zero count=1 | hexdump -v` permet de voir les 512 premiers octets de ce périphérique virtuel (en hexadécimal). Regardez-le. Qu'est-ce qu'il a de particulier ?

Q25 Le périphérique `urandom` est un périphérique virtuel. La commande `dd if=/dev/urandom count=1 | hexdump -v` permet de voir les 512 premiers octets de ce périphérique virtuel (en hexadécimal). Regardez-le. Recommencez. Qu'est-ce qu'il a de particulier ?

Q26 Le périphérique `sda1` est une des partitions du disque dur. La commande `dd if=/dev/sda1 count=1 | hexdump -v` permet de voir les 512 premiers octets de ce périphérique virtuel (en hexadécimal). Regardez-le. Que se passe-t-il ? Un programme normal comme `dd` peut-il examiner le disque dur en outrepassant le système de fichiers ?

Q27 En utilisant la commande `mount`, analysez les différentes partitions présentes dans votre système. Identifiez celles qui correspondent à un vrai périphérique et les systèmes de fichier virtuel.

Espace libre

Une partition occupe une taille fixe. La plupart des systèmes de fichiers sont **de taille fixe**. Elles peuvent accueillir uniquement une certaine quantité de données.

L'espace de travail

Comme dans un parking, la quantité de données que l'on peut mettre dans un disque ne doit pas être égale à la quantité de données qu'il peut accueillir, sinon, on ne peut pas faire un certain nombre d'opérations. En plus de l'espace réservé à la signalisation (index et tables divers), on réserve aussi un peu d'espace pour les programmes importants du système.

La fragmentation

Les fichiers sont posés par petits blocs dans la partition (qui est elle-même un gros bloc dans l'ensemble du disque). Parce qu'un fichier est plus rapide à lire si les blocs sont les uns à côté des autres, les systèmes de fichiers essayent de maintenir cet état. Sous Windows, on peut aider le système en procédant à une opération de rangement : la *défragmentation*.
Les systèmes de fichiers utilisés sous Linux n'ont quasiment pas de fragmentation si on utilise moins de 95% de leur espace.

Syntaxe pour df

`df [-h] [emplacement]`

Description

- Affiche les disques montés et leur capacité de mémoire (celui de la partition qui correspond à l'emplacement, tous s'il n'y en a pas).
- L'option `-h` (human readable) convertie l'affichage des tailles mémoires en unités conventionnelles binaires (en nombre de blocs par défaut). Avec `-H` c'est en unités décimales.

Exemple d'utilisation:

```
login@host:~$ df -h
 Sys. de fichiers     Taille   Uti.   Disp.   Uti%   Monté sur
 /dev/sda1            56G      16G    37G     31%    /
 myserver:/home/moi   1,8T     1,6T   192G    90%    /users/moi
 ...                  ...      ...    ...     ...    ...
login@host:~$
```

Exercices

Partitions et espace disque

Q28 Analysez l'espace disque disponible et utilisé sur toutes les partitions de votre ordinateur. Comparez les unités décimales et binaires.

Q29 En utilisant `df /` et `df /etc`, vérifiez que ces deux répertoires sont bien sur la même partition. Comparez avec `df ~` et `df /boot`.

Q30 Si un utilisateur remplit complètement la partition qui abritent ses données, qu'est-ce qui cesse de fonctionner ? Qu'est-ce qui peut continuer à fonctionner ?

Q31 L'administrateur `root` a pour répertoire personnel `/root`. En regardant sur quelle partition c'est, expliquez pourquoi ce n'est pas dans `/home`.

Les nœuds d'index

Les nœuds d'index ou inodes

- Les données des fichiers sont stockées dans des blocs numérotés.

L'organisation des fichiers et répertoires est elle stockée dans des blocs spéciaux appelés nœuds d'index (ou *inodes*). Un chemin est associé à un inode unique qui va contenir la liste des numéros de blocs de données qu'il utilise.

- Un répertoire est un inode qui pointe vers un bloc de données qui contient un tableau de type nom → inode
- Un fichier est un inode qui pointe vers un ou plusieurs blocs de données qui contiennent... les données.

L'inode, c'est le représentant du fichier. Il contient aussi les *métadonnées* associées.

Lire la structure des inodes

Les inodes sont simplement désignés par un numéro. La commande `ls -i` ou `stat` permet d'accéder à cette information.

Syntaxe pour stat

```
stat  chemin [chemin_2 ...]
```

Description

- Si le chemin est occupé par un fichier ou un répertoire, affiche les métadonnées relatives au chemin.

Exemple d'utilisation:

```
login@host:~$ stat /etc/fstab
Fichier : « /etc/fstab » Taille : 672 Blocs : 8
Blocs d'E/S : 4096 fichier Périphérique : 801h/2049d
Inœud : 5244539 Liens : 1 Accès : (0644/-rw-r-r-)
UID : ( 0/ root) GID : ( 0/ root) Accès : 2015-09-22
22:04:47.768711256 +0200 Modif. : 2013-11-13
09:12:51.763972183 +0100 Changt : 2015-08-09
10:57:36.553313285 +0200 Créé : -
login@host:~$
```

Exercices

Découvrir les métadonnées

Q32 En utilisant la commande `stat ~`, trouvez le numéro d'inode de votre répertoire personnel.

Q33 Quelles autres sortes de métadonnées arrivez-vous à comprendre ?

Q34 Avec la commande `ls -ial ~`, regardez les numéros d'inodes de vos répertoires. Sont-ils différents ? Comment l'expliquez-vous ?

Q35 Maintenant, regardez le numéro d'inode avec `stat` du répertoire `/home/usager`. Est-ce que vous le reconnaissez ?

Q36 Regardez avec ces commandes les valeurs des numéros d'inode de `/`, `/.` et `/..`. Expliquez.

Q37 Recommencez avec `stat /home` et `stat /`. Que remarquez-vous à propos de ces numéros d'inodes ? Est-ce que ces répertoires sont identiques ? Comme l'expliquer ?

Q38 Dans le répertoire `~/Documents/m1101/textes`, il y a un fichier texte. Regardez ses métadonnées. Dans une autre fenêtre, lisez-le. Puis regardez encore. Est-ce que quelque chose a changé ?

Q39 Faites une copie de ce fichier. Modifiez la copie avec `gedit copie.txt`. Vérifiez que les métadonnées changent encore. Lesquelles ? Est-ce qu'il y a une commande qui permet de faire ce changement de métadonnées sans vraiment changer le fichier ? (après, supprimez la copie).

Les liens durs

Un fichier, deux chemins (et plus si affinités)

Vu la structure utilisée, il est possible de mettre le même numéro d'inode dans deux répertoires différents (même nom ou pas) ou sous deux noms dans le même répertoire. On obtient ainsi deux chemins qui pointent vers le même fichier. C'est ce qu'on appelle un *lien* ou *lien dur* (hardlink).
Lorsqu'on édite le premier « fichier », le deuxième est aussitôt modifié (normal, c'est le même).
Lorsqu'on supprime l'un des deux fichiers, l'autre reste. Pour savoir quand effacer vraiment le fichier, il utiliser le compteur de liens (lorsqu'il est à zéro, on peut effacer).
On ne peut pas faire de liens durs entre partitions différentes.q

Boîte à outils : les partitions

Commande	Fonction principale
`mount`	Manipuler les partitions
`df`	Afficher l'espace restant
`ls`	Afficher le contenu d'un répertoire
`stat`	Afficher les métadonnées d'un chemin
`ln`	Créer un lien (dur ou symbolique)

Syntaxe pour ln

```
ln  source [cible]
```

Description

- Crée un lien dur entre la référence source et le chemin cible.

Exemple d'utilisation:

```
login@host:~$ ln astronomie.txt Etoiles/lisezmoi.txt
```

Le lien sur un fichier crée une deuxième entrée pointant vers le même inode.

Exercices

Liens durs

Q40 Dans votre répertoire `~/Documents/m1101/textes`, créez un petit fichier `source.txt`. Insérez quelques lignes de texte dedans (avec `gedit source.txt`), puis quittez l'éditeur de texte.

Q41 Vérifiez dans la console son numéro d'inode, et son contenu. Quelles sont les commandes pour cela ?

Q42 Faites une copie de `source.txt` vers `copie.txt`, et un lien de `source.txt` vers `lien.txt`. Vérifiez le contenu de la copie et du lien. Vérifiez que les numéros d'inode et les compteurs de liens sont comme ce à quoi vous vous attendiez.

Q43 Avec l'éditeur de texte, comme plus haut, modifiez le fichier source. Regardez les trois fichiers dans le terminal. Est-ce conforme à vos attentes ? Essayez ensuite en changeant la copie.

Q44 Faites un lien de `lien.txt` vers `lienlien.txt`. Vérifiez le compteur de liens. Effacez ensuite `lien.txt`, et vérifiez encore.

Q45 Modifiez `lienlien.txt`, puis regardez tous les contenus. Effacez le fichier `source.txt`. Le fichier `lienlien.txt` est-il toujours là ?

Q46 Essayez de faire un lien entre `copie.txt` et `/tmp/copie.txt`. Que se passe-t-il ? Pourquoi ? Expliquez.

Les liens symboliques

Une redirection

Sous Windows ou sous Unix, on peut créer des « raccourcis » qui lient un chemin spécifique à un autre endroit dans l'arborescence.

- Unix les appelle des liens symboliques.
- Windows les appelle des raccourcis.
- Un lien symbolique est un chemin (relatif ou absolu) qui indique un autre point de l'arbre. Il fonctionne au niveau chemin et pas au niveau inode.
- Un lien symbolique peut traverser les partitions.

Un lien symbolique peut pointer sur un chemin qui ne correspond pas à un fichier ou un répertoire. On dit qu'il est brisé.

Exercices

Liens symboliques

Q47 Dans votre répertoire `~/Documents/m1101/textes`, recréez un petit fichier `source.txt`, effacez `lienlien.txt` et `copie.txt`.

Q48 Faites un lien symbolique de `source.txt` vers `lien.txt`. Vérifiez le contenu du lien. Regardez les métadonnées associées.

Q49 Avec l'éditeur de texte, comme plus haut, modifiez le fichier source. Regardez les deux fichiers dans le terminal. Est-ce conforme à vos attentes ?

Q50 Faites un lien de `lien.txt` vers `lienlien.txt`. Vérifiez le contenu des trois fichiers (en modifiant).

Q51 Modifiez `lienlien.txt`, puis regardez tous les contenus. Effacez le fichier `source.txt`. Le fichier `lienlien.txt` est-il toujours là ?

Q52 Essayez de faire un lien entre `commande.txt` et `/tmp/story.txt`. Que se passe-t-il ? Pourquoi ? Expliquez.

Q53 Faites un lien symbolique vers un répertoire. Que se passe-t-il ? Quel est le danger ?

L'archivage et les liens

Les formats zip et tar

- Le format `tar` permet d'archiver autant les liens durs que les liens symboliques
- Le format `zip` ne permet que d'archiver les liens symboliques
- Le programme `zip` remplace, par défaut, les liens symboliques par des copies.
- L'option `-symlinks` permet de conserver les liens symboliques dans les archives `zip`.
- Le programme `tar` a une option `-dereference` qui transforme les liens symboliques en copies. Les liens durs ne sont archivés qu'une seule fois par défaut.

Un lien symbolique archivé en tant que tel pointant en dehors de l'archive peut être brisé !

Exemple (Création d'archives)

```
login@host:~$ zip --symlinks -r sel.zip sel/
adding: selection/ (stored 0%)
adding: selection/best.jpg (stored 0%)
adding: selection/img_1363.jpg (deflated 2%)
adding: selection/img_1221.jpg (deflated 1%)
login@host:~$ unzip sel.zip
```

Exercices

Liens et archives

Q54 Dans votre répertoire `~/Documents/m1101`, créez un répertoire `selection`.

Q55 Faites un lien symbolique dans `selection` d'une image de `images`, un lien dur et une copie de fichier. Rajoutez un lien dur dans `selection` vers la copie de fichier sous un autre nom (par exemple `lameilleure.jpg`.

Q56 Archivez le répertoire `selection` au format `tar`. Vérifiez avec `tar vvf selection.tar` que les fichiers sont tous présents, sauf le lien symbolique.

Q57 Archivez avec `zip` le même répertoire. Vérifiez (avec `stat sel.zip`) que la taille de l'archive est cohérente avec la présence de quatre images (et non pas trois).

Q58 Archivez dans un autre fichier `zip` le même répertoire avec la conservation des liens symboliques. Comparez les tailles et expliquez.

Q59 Créez trois répertoires `d1`, `d2`, `d3`. Dans chacun de ces répertoires, décompressez les archives crées précédemment. Regardez ce qui arrive aux liens symboliques et aux liens durs dans chacun des cas.