

Gérer les fichiers sous Linux

Attribution - Partage dans les Mêmes Conditions :
<http://creativecommons.org/licenses/by-sa/3.0/fr/>

Table des matières

I - Contexte	3
II - Fonctionnement des fichiers	4
III - Exercice : Appliquer la notion	6
IV - Arborescence des fichiers	7
V - Exercice : Appliquer la notion	10
VI - Éditer des fichiers	11
VII - Exercice : Appliquer la notion	14
VIII - Déplacer des fichiers	15
IX - Exercice : Appliquer la notion	18
X - Liens symboliques	19
XI - Exercice : Appliquer la notion	21
XII - Rechercher des fichiers	22
XIII - Exercice : Appliquer la notion	24
XIV - Afficher des fichiers	25
XV - Exercice : Appliquer la notion	29
XVI - Quiz	30
Solutions des exercices	32
Crédits des ressources	37

I Contexte

Durée : 2h

Environnement de travail : Terminal Linux

Pré-requis : Aucun

Lorsque l'on utilise un ordinateur, on manipule souvent des fichiers. Linux ne fait pas exception, et nous allons voir qu'il existe de nombreux outils en ligne de commande qui permettent de créer, consulter, modifier, et organiser les fichiers et dossiers de l'ordinateur. Nous allons d'abord voir comment ils sont gérés et organisés, puis nous verrons les outils pour les manipuler.

II Fonctionnement des fichiers

Objectifs

- Comprendre l'organisation en fichiers sur Linux
- Connaître les premières commandes pour naviguer dans les dossiers

Lister les fichiers : `ls`

 Syntaxe

La commande `ls` permet de lister les fichiers et dossiers de l'endroit où l'on se trouve dans son terminal. Par défaut, lorsque l'on ouvre la console, on se trouve dans son dossier d'utilisateur. Si on fait un `ls` on voit donc les fichiers et dossiers classiques de ce répertoire.

```
1 kyane@europa:~$ ls
2 Documents Images Musique Téléchargements Vidéos
```

 Rappel

On peut utiliser l'option `-p` ou `-F` pour visualiser les dossiers

```
1 kyane@europa:~$ ls -p
2 Documents/ Images/ Musique/ Téléchargements/ Vidéos/ firefox.config
```

Naviguer dans le terminal : `cd`

 Syntaxe

Si l'on souhaite se rendre dans le dossier `Documents` on utilise la commande `cd` (*change directory*, changer de dossier) suivi du nom du dossier dans lequel on veut se rendre.

```
1 kyane@europa:~$ cd Documents
2 kyane@europa:~/Documents$ ls
3 Boulot Etudes Finances Logement Picasoft
```

On constate que le prompt est à jour et que l'on est dans le dossier `~/Documents`.

Séparateur

 Fondamental

Sous Linux le caractère de séparation de dossier est `/`. Ainsi le fichier `/home/kyane/firefox.config` est le fichier nommé `firefox.config` se trouvant dans le dossier `kyane` qui est lui-même dans le dossier `home`.

Fichiers cachés : `ls -a`

 Syntaxe

Tous les fichiers dont le nom commence par un `.` sont des fichiers cachés : ils ne s'affichent pas par défaut lorsque qu'on utilise la commande `ls`.

C'est utile pour des fichiers de configuration qui ne sont pas souvent modifiés, pour ne pas "polluer" l'interface avec pleins de fichiers de ce type que l'on ne souhaite pas voir.

Pour lister les fichiers cachés, on utilise l'option `-a`.

```
1 kyane@europa:~$ls -a
2 .  ..  .bash_logout  .bashrc Documents  Images  Musique  .profile Téléchargements
   Vidéos  firefox.config
```

On voit ici les fichiers `.bash_rc` et `.profile` qui sont des fichiers cachés du répertoire utilisateur (en l'occurrence qui servent à configurer certains éléments du terminal).

Raccourcis : `.` et `..`

 Remarque

On constate que la commande `ls -a` retourne systématiquement deux dossiers nommés `"."` et `".."`.

```
1 kyane@europa:~/Documents$ ls -a
2 .  ..  Boulot  Etudes  Finances  Logement  Picasoft
```

Ce sont des raccourcis pour désigner :

- le dossier courant dans le cas de `.`
- le dossier parents dans le cas de `..`

Par exemple cela nous permet de retourner facilement au dossier parent.

```
1 kyane@europa:~$ cd Documents
2 kyane@europa:~/Documents$ cd ..
3 kyane@europa:~$
```

Dossier utilisateur : `~`

 Remarque

Le dossier de base de l'utilisateur (on parle de *home directory*) est désigné par le caractère `"~"`. Par écrire `~/firefox.config` est équivalent à écrire `/home/kyane/firefox.config`, `"~"` remplace le nom complet de mon dossier utilisateur, à savoir `/home/kyane`.

À retenir

Sous Linux le chemin des fichiers utilise le caractère `"/"` pour séparer les noms de dossier, et les fichiers ayant un `"."` en début de nom sont considérés comme cachés.

Naviguer avec un terminal est très simple en utilisant la commande `cd` et en s'appuyant sur des raccourcis comme `"."`, `".."` et `"~"`.

III Exercice : Appliquer la notion

Question 1

[solution n°1 p. 32]

Sur votre machine, rendez vous dans le dossier nommé `log` qui se trouve lui-même dans le dossier nommé `/var`.

Indice :

On utilise la commande `cd`

Question 2

[solution n°2 p. 32]

Une fois dans le dossier en question, listez les fichiers qui s'y trouve. Les dossier doivent être facilement identifiables.

Indice :

L'option `-p` ou `-F` de `ls` devrait aider.

IV Arborescence des fichiers

Objectifs

- Savoir ce qu'est la racine
- Savoir faire la différence entre un chemin relatif et absolu
- Comprendre l'arborescence classique sous Linux

La racine

Az Définition

La racine dans un système de fichier est le dossier de base qui contient tout les autres dossiers, toute l'arborescence. Le cas de de Linux, la racine est le dossier /. Il n'y a rien au dessus, aucun dossier ne contient la racine, et tout les fichiers, dossiers et sous-dossiers sont contenus dans /.

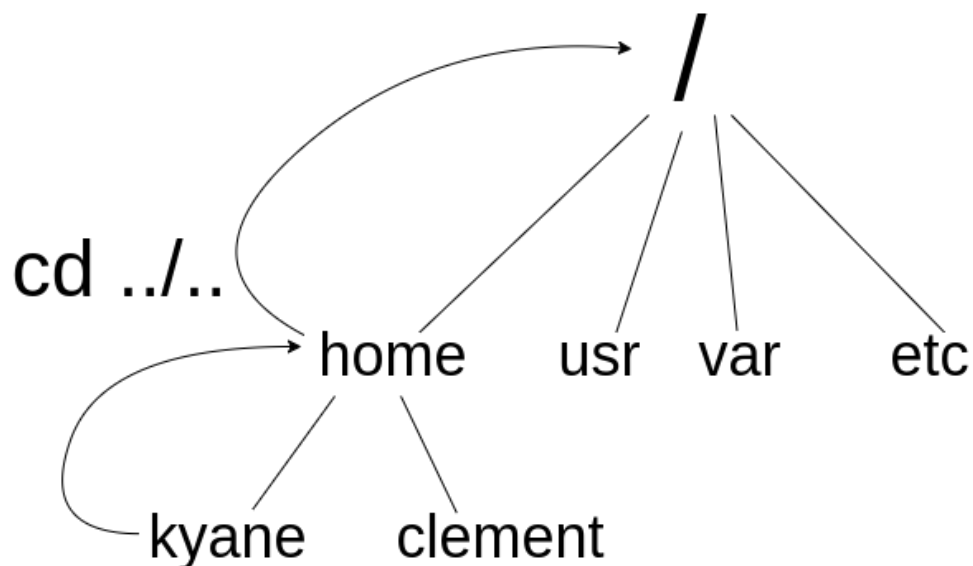
Exemple

Dans n'importe quel dossier, je peux utiliser la commande pwd pour connaître l'endroit où je me trouve.

```
1 kyane@europa:~$ pwd
2 /home/kyane
```

Ici c'est le dossier de l'utilisateur (on le remarque grâce au "~" dans le prompt), c'est à dire /home/kyane. Si l'on veut se rendre à la racine, on peut donc remonter 2 dossiers en arrière.

```
1 kyane@europa:~$ cd ../../
2 kyane@europa:/$ pwd
3 /
4 kyane@europa:/$ ls
5 bin  dev  home  lib  lib64  media  opt  root  sbin  sys  usr  vmlinuz
6 boot  etc  initrd.img  lib32  libx32  mnt  proc  run  srv  tmp  var
```



Chemin absolu ou relatif

Dans l'exemple précédent on a utilisé `cd ../..` pour se rendre à la racine, mais ça ne fonctionne que parce que l'on avait besoin de remonter 2 dossiers en arrière pour atteindre la racine. Si on avait été dans le dossier `/var/log/docker` on aurait eu besoin de faire `cd ../../..`, ou alors simplement `cd /`. Il y a donc 2 manières de se déplacer d'un dossier à l'autre.

Chemin absolu

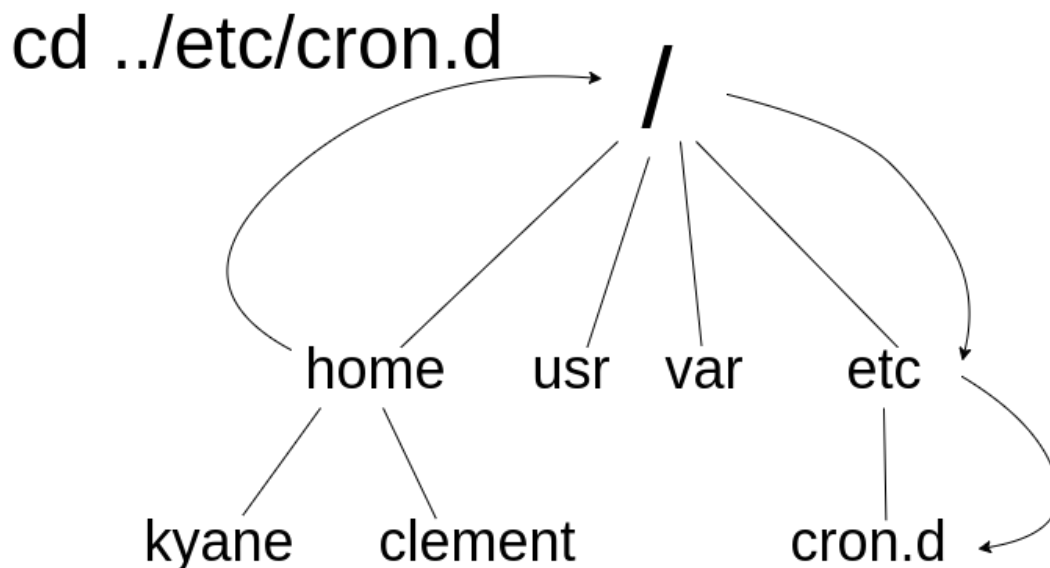
Az Définition

Un chemin absolu est un chemin qui permet de désigner un fichier (ou dossier) en fonction de sa localisation par rapport à la racine. On le reconnaît car il commence tout le temps par la caractère `/`, la racine. Par exemple `/etc/cron.d` est un chemin absolu : quelque soit le dossier dans lequel je me trouve initialement je peux m'y déplacer en faisant `cd /etc/cron.d`.

Chemin relatif

Az Définition

Un chemin relatif est un chemin qui permet de désigner un fichier (ou dossier) en fonction de sa localisation par rapport au dossier dans lequel on se trouve actuellement. Par exemple `cd ../etc/cron.d` permet de se rendre, à l'aide d'un chemin relatif, dans le dossier `/etc/cron.d` si l'on se trouve dans le dossier `/home`.



L'arborescence Linux

À la racine se trouve une grande quantité de dossiers différents, comme nous avons pu le voir.

```
1 kyane@europa:/etc/cron.d$ ls /
2 bin  dev  home  lib  lib64  media  opt  root  sbin  sys  usr
3 boot  etc  lib32  libx32  mnt  proc  run  srv  tmp  var
```

Ces dossiers sont souvent les mêmes d'une machine ou d'une distribution à une autre. En effet Linux a tendance à suivre une certaine logique d'organisation des fichiers et sous-dossiers, chaque dossier présent à la racine va contenir des fichiers selon certaines thématiques. Par exemple :

- `bin` est un dossier qui va contenir beaucoup de programmes et commandes qui peuvent s'exécuter (des fichiers BINaires donc) ;
- `dev` va contenir des dossiers particuliers qui vont permettre de gérer les périphériques : disque, clef USB, écran, etc ;
- `home` va contenir les dossiers d'utilisateurs. Par exemple `/home/kyane` est le dossier d'utilisateur de kyane, tandis que `/home/clement` est celui de clement ;
- `var` va contenir tout un tas de données variables : des messages du système (*logs*), du cache pour certains logiciels, etc.
- `etc` contient la plupart des fichiers de configurations.

Il n'est pas nécessaire de connaître tout les dossiers, mais il est important de savoir qu'il y a une logique d'organisation. Avec l'habitude, cela permettra d'identifier rapidement si un fichier est de la configuration ou un programme, en fonction de là où on le trouve.

À retenir

Sous Linux les fichiers et dossiers sont organisés selon une certaine logique et forment une arborescence. Celle-ci commence à la racine, qui est le dossier parent de tout le système de fichier. Pour désigner un fichier, on utilise son chemin, soit absolu (à partir de la racine), soit relatif (en fonction du dossier courant).

V Exercice : Appliquer la notion

Question 1

[solution n°3 p. 32]

Exécutez la commande `cd` sans paramètres, dans quel dossier est-ce que l'on se retrouve ?

Indice :

On utilise la commande `pwd`

Question 2

[solution n°4 p. 32]

On se rend dans le dossier `/var/log` puis on lance la commande `cd ../cache/apt/`. Quel est le chemin absolu du dossier dans lequel on se retrouve ?

VI Éditer des fichiers

Objectifs

- Savoir créer un fichier ou un dossier
- Savoir modifier un fichier texte dans son terminal

Mise en situation

Sous Linux, la quasi-totalité des fichiers de configuration sont de simples fichiers textes. Sur un serveur par exemple, on crée et modifie quotidiennement des fichiers permettant de configurer les différents logiciels de celui-ci. Il est donc particulièrement important de savoir comment faire ceci dans un terminal.

Créer un fichier

 Méthode

Créer un fichier est très simple avec la commande `touch`.

```
1 kyane@europa:~/librecours$ touch roman.txt
2 kyane@europa:~/librecours$ ls
3 roman.txt
```

La commande a bien créé un fichier nommé `roman.txt` dans le dossier où je me trouve (en l'occurrence `~/librecours`).

 Remarque

Si on regarde le manuel de la commande `touch`, on constate qu'il est indiqué « *touch - change file timestamps* », ce qui veut dire que c'est une commande qui permet de changer la date de modification d'un fichier. On pourrait se demander pourquoi on utilise cette commande pour créer un fichier dans ce cas, mais la suite du manuel nous indique « *A FILE argument that does not exist is created empty, unless -c or -h is supplied.* ». Si le fichier n'existe pas, il est donc créé, c'est pour cela que cette commande est celle utilisée pour créer un fichier sous Linux.

Créer un dossier

 Méthode

Créer un dossier est tout aussi simple qu'un fichier, on utilise simplement la commande `mkdir`.

```
1 kyane@europa:~/librecours$ mkdir photos
2 kyane@europa:~/librecours$ ls -F
3 photos/  roman.txt
4 kyane@europa:~/librecours$ cd photos/
5 kyane@europa:~/librecours/photos$
```

Il est aussi possible de créer plusieurs dossiers en même temps, en spécifiant plusieurs arguments.

```
1 kyane@europa:~/librecours$ mkdir films musiques
```

Ou encore de créer une arborescence de sous-répertoires d'un seul coup.

```
1 kyane@europa:~/librecours$ mkdir -p documents/factures/gaz
```

Nano, un éditeur de texte en console

Un éditeur de texte est un programme qui permet de modifier des fichiers texte, on connaît par exemple le fameux Bloc-Notes sous Windows. À la différence du traitement de texte, qui est un logiciel qui édite des fichiers aux formats complexes, l'éditeur de texte est très léger, à tel point qu'il en existe de nombreux disponibles directement dans le terminal. Le plus simple pour débuter est sans doute Nano.

Méthode

Pour lancer l'édition d'un texte avec Nano, il suffit d'utiliser la commande du même nom et de spécifier le fichier.

```
1 kyane@europa:~/librecours$ nano roman.txt
```

Une fenêtre s'ouvre alors dans le terminal, c'est l'éditeur de texte.



Écrire du texte

 Méthode

Pour écrire dans notre fichier, rien de plus simple, il suffit de taper au clavier. La touche entrée permet d'aller à la ligne, et les flèches directionnelles de se déplacer dans le texte.



Extrait du roman Traces (Stéphane Crozat, Framabook, 2018)

Les raccourcis

 Méthode

Pour effectuer les opérations principales (enregistrer le fichier, copier/coller du texte, etc.) Nano s'appuie sur des raccourcis clavier qui sont, en partie, spécifiés en bas de la fenêtre.



Le symbole ^ désigne la touche Ctrl. Pour enregistrer le fichier on fera donc Ctrl + o. Nano propose ensuite, en bas de fenêtre, de confirmer le nom du fichier.

De la même manière on peut utiliser Ctrl + x pour fermer la fenêtre.

À retenir

Les commandes touch et mkdir permettent très simplement de créer fichiers et dossier. Les fichiers peuvent ensuite être édités directement en console à l'aide de Nano, qui est un éditeur particulièrement adapté pour débuter, avec son interface minimaliste et ses raccourcis explicitement indiqués dans la fenêtre.

VII Exercice : Appliquer la notion

Dans le dossier de votre choix, on souhaite créer des sous-dossiers qui vont contenir des fichiers.

Question 1

[solution n°5 p. 32]

Créez 3 dossiers en utilisant **une seule commande**

- documents
- documents/finances
- images

Indice :

L'option -p de la commande `mkdir` devrait permettre de tout faire en une fois.

Question 2

[solution n°6 p. 32]

Dans le dossier `documents`, créez 2 fichiers vides nommés `todo.txt` et `cv.tex`. Listez ensuite le contenu du dossier pour que l'on fasse la différence entre les dossiers et fichiers.

Question 3

[solution n°7 p. 33]

Dans un fichier `documents/finances/compta.txt` écrivez quelques lignes de texte.

Indice :

Le texte en lui même n'est pas important, l'essentiel est d'utiliser l'outil Nano correctement.

VIII Déplacer des fichiers

Objectifs

- Savoir copier un fichier
- Savoir déplacer un fichier
- Savoir supprimer un fichier

Copier un fichier

 Méthode

Pour copier un fichier on utilise la commande `cp` (abréviation de *copy*). Elle prend en paramètre le fichier source puis le fichier (ou dossier) de destination.

```
1 kyane@europa:~/librecours$ cp documents/compta-2019.txt archives/
```

Cette commande va copier le fichier `documents/compta-2019.txt` dans le dossier `archives`.

```
1 kyane@europa:~/librecours$ ls archives/
2 compta-2019.txt
```

Copier un dossier

 Méthode

Pour copier un dossier on utilise la même commande, mais il est nécessaire d'utiliser le paramètre `-r` (pour *recursive*). Cela permet d'indiquer que l'on souhaite copier le répertoire, mais aussi tout les fichiers et sous-dossiers qu'il contient

```
1 kyane@europa:~/librecours$ ls images/
2 photo.png
3 kyane@europa:~/librecours$ cp -r images/ archives/
4 kyane@europa:~/librecours$ ls -F archives/
5 compta-2019.txt images/
6 kyane@europa:~/librecours$ ls -F archives/images/
7 photo.png
```

Ici tout le dossier `images` et son contenu sont copiés dans le dossier `archives`.

Déplacer un fichier

 Méthode

La commande `mv` est utilisée pour déplacer un fichier ou un dossier, et donc indirectement aussi pour les renommer. Elle s'utilise de la même manière que la commande `cp`.

```
1 mv vieuxfichier nouveaufichier
```

Par exemple pour déplacer un fichier dans un autre dossier

```
1 kyane@europa:~/librecours$ mv roman.txt archives/
```

Supprimer un fichier

 Méthode

Enfin, la commande `rm` (abréviation de *remove*) permet de supprimer un fichier sur la machine.

```
1 kyane@europa:~/librecours$ ls archives/
2 compta-2019.txt  images  roman.txt
3 kyane@europa:~/librecours$ rm archives/roman.txt
4 kyane@europa:~/librecours$ ls archives/
5 compta-2019.txt  images
```

Certains fichiers sont protégés en écriture, de telle sorte qu'il est nécessaire de confirmer la suppression, pour éviter les erreurs.

```
1 kyane@europa:~/librecours$ rm protected.txt
2 rm : supprimer 'protected.txt' qui est protégé en écriture et est du type
  « fichier vide » ? y
3
```

Forcer la suppression

 Remarque

Pour éviter de devoir confirmer la suppression d'un fichier il est possible d'ajouter l'option `-f` à la commande, pour forcer la suppression du fichier. Mais attention, si le fichier est protégé en écriture c'est peut-être qu'il y a une bonne raison...

Suppression définitive

 Attention

Il est très important de noter que `rm` est une commande qui supprime définitivement le fichier de la machine. Il n'y a pas de système de corbeille ou de sauvegarde que l'on peut restaurer lorsque l'on utilise cette commande. Il est donc important de réfléchir avant de lancer cette commande.

Supprimer un dossier

 Méthode

Pour supprimer un dossier, on utilise la même commande mais il est nécessaire d'utiliser le paramètre `-r` (comme avec `cp`)

```
1 kyane@europa:~/librecours$ rm -r archives/
```

À retenir

Avec nos 3 nouvelles commandes, on sait désormais comment copier, déplacer, renommer ou supprimer des fichiers et dossiers sur notre ordinateur. Elles sont simple d'utilisation, et généralement la différence entre fichiers et dossiers se traduit simplement par l'utilisation de l'option `-r`. On prend garde cependant à la commande `rm`, qui supprime de manière définitive.

Métacaractère

 Méthode

Un métacaractère est un caractère qui peut être substitué par une sélection de motifs, vous donnant plus de flexibilité dans la recherche. Vous pouvez utiliser des métacaractères, dans les commandes que nous avons vu, pour plus de flexibilité, par exemple pour copier plusieurs fichiers en même temps. Il en existe plusieurs :

- `*` le métacaractère des métacaractères, il est utilisé pour représenter n'importe quel caractère ou n'importe quelle chaîne de caractères ;
- `?` utilisé pour représenter un unique caractère ;
- `[ ]` utilisé pour représenter n'importe quel caractère dans les crochets.

Le plus courant est d'utiliser `*`, par exemple pour copier tout mes fichiers images dans un autre dossier.

```
1 kyane@europa:~/librecours$ cp images/*.png archives/
```

Cela copie tout les fichiers terminant par `.png` depuis le dossier `images` vers le dossier `archives`.

 Attention

L'utilisation de métacaractères peut être dangereuse. On suppose qu'il existe un fichier appelé `-R`.

Si on exécute `rm *`, la commande construite sera `rm -R *`, qui efface récursivement tous les fichiers.

Pour prévenir ce comportement, on peut séparer explicitement les options des arguments en utilisant l'option `--` :

`rm *` exécutera `rm -- -R *` et l'argument `-R` sera bien interprété comme un nom de fichier et non comme une option.

IX Exercice : Appliquer la notion

Créez un dossier vide dans lequel réaliser l'exercice. Une fois placé dans ce dossier, copiez/collez les commandes suivantes pour préparer "l'environnement".

```
1 mkdir -p imgaes/vacances-2019 imgaes/noel-2018 imgaes/a-developper
2 touch imgaes/vacances-2019/photo1.jpg imgaes/vacances-2019/photo2.jpg
  imgaes/vacances-2019/photo3.jpg imgaes/noel-2018/cadeau-papa.png imgaes/noel-
  2018/cadeau-sarah.png imgaes/noel-2018/cadeau-jonathan.png
```

Question 1

[solution n°8 p. 33]

Pour commencer on constate que le dossier `imgaes` porte un nom un peu bizarre, sans doute une faute de frappe. Renommez-le correctement.

Question 2

[solution n°9 p. 33]

On souhaite développer toutes les photos de Noël 2018, ainsi que la "photo 1" des vacances de 2019. Pour cela on veut **les copier** dans le dossier `images/a-developper/` que l'on va envoyer au développeur de photos.

Indice :

Il y a plusieurs manières de procéder, mais il est possible de tout faire en une seule commande.

X Liens symboliques

Objectifs

- Comprendre la notion de lien symbolique
- Utiliser la commande `ln`

Lien symbolique

Az Définition

Certains fichiers ne sont que des pointeurs vers d'autres fichiers. On les appelle des **liens symboliques**.

Un lien symbolique contient le chemin relatif ou absolu d'un fichier ou d'un répertoire qui peut être n'importe où dans l'arborescence. Il permet de faire référence à ce fichier ou ce répertoire à un autre endroit.

```
1 bonnetst@vivaldi:~/api$ ls -l
2 total 16
3 -rw-rw-r-- 1 bonnetst bonnetst 13 janv. 25 20:34 fichier.txt
4 lrwxrwxrwx 1 bonnetst bonnetst 11 janv. 25 20:41 lien.txt -> fichier.txt
5 bonnetst@vivaldi:~/api$ cat fichier.txt
6 Vive Linux !
7 bonnetst@vivaldi:~/api$ cat lien.txt
8 Vive Linux !
```

Création de liens symboliques

Méthode

La commande `ln -s <nom du fichier> <nom du lien>` permet de créer des liens symboliques.

```
1 bonnetst@vivaldi:~/api$ ln -s fichier.txt lien2.txt
2 bonnetst@vivaldi:~/api$ ls -l
3 total 20
4 -rw-rw-r-- 1 bonnetst bonnetst 13 janv. 25 20:34 fichier.txt
5 lrwxrwxrwx 1 bonnetst bonnetst 11 janv. 25 20:45 lien2.txt -> fichier.txt
6 lrwxrwxrwx 1 bonnetst bonnetst 11 janv. 25 20:41 lien.txt -> fichier.txt
```

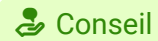
Chemins dans les liens symboliques

Attention

Un lien symbolique fait référence à un fichier ou répertoire **exactement** comme il est saisi dans la commande `ln`. Il faut que cette référence soit un chemin correct quelque soit l'endroit d'où le lien est utilisé.

```
1 bonnetst@vivaldi:~/api$ mkdir subdir
2 bonnetst@vivaldi:~/api$ ln -s fichier.txt subdir/lien.txt
3 bonnetst@vivaldi:~/api$ ls -l subdir
4 total 4
5 lrwxrwxrwx 1 bonnetst bonnetst 11 janv. 25 20:54 lien.txt -> fichier.txt
6 bonnetst@vivaldi:~/api$ cat subdir/lien.txt
7 cat: subdir/lien.txt: Aucun fichier ou dossier de ce type
```

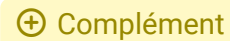
En suivant le lien, le système cherche le fichier dans le répertoire `subdir` et ne le trouve pas. Il aurait fallu donner dans le lien le chemin relatif correct de `fichier.txt`

Comment toujours créer des liens symboliques corrects.

Pour créer un lien symbolique correct, il faut toujours utiliser :

- Le chemin **absolu** vers la cible
- Le chemin relatif vers la cible **depuis l'emplacement du lien.**

```
1 bonnetst@vivaldi:~/api$ ln -sf ../fichier.txt subdir/lien.txt
2 bonnetst@vivaldi:~/api$ cat subdir/lien.txt
3 Vive Linux !
```

Manipulation des liens symboliques

Les liens symboliques peuvent être supprimés par `rm`. Seul le lien est alors supprimé, pas le fichier vers lequel il pointe.

En revanche, la copie d'un lien par `cp` copie le fichier et non lien par défaut. L'option `-d` permet de copier le lien lui-même.

```
1 bonnetst@vivaldi:~/api/subdir$ cp lien.txt lien2.txt
2 bonnetst@vivaldi:~/api/subdir$ ls -l
3 total 4
4 lrwxrwxrwx 1 bonnetst bonnetst 14 janv. 25 21:02 lien.txt -> ../fichier.txt
5 bonnetst@vivaldi:~/api/subdir$ cp lien.txt lien2.txt
6 bonnetst@vivaldi:~/api/subdir$ ls -l
7 total 16
8 -rw-rw-r-- 1 bonnetst bonnetst 13 janv. 25 21:10 lien2.txt
9 lrwxrwxrwx 1 bonnetst bonnetst 14 janv. 25 21:02 lien.txt -> ../fichier.txt
10
```

`lien2.txt` est une copie du fichier, pas du lien.

```
1 bonnetst@vivaldi:~/api/subdir$ cp -d lien.txt lien3.txt
2 bonnetst@vivaldi:~/api/subdir$ ls -l
3 total 20
4 -rw-rw-r-- 1 bonnetst bonnetst 13 janv. 25 21:10 lien2.txt
5 lrwxrwxrwx 1 bonnetst bonnetst 14 janv. 25 21:11 lien3.txt -> ../fichier.txt
6 lrwxrwxrwx 1 bonnetst bonnetst 14 janv. 25 21:02 lien.txt -> ../fichier.txt
```

`lien3.txt` est bien une copie du lien.

XI Exercice : Appliquer la notion

Question

[solution n°10 p. 33]

Dans votre répertoire personnel, créez un fichier `api.txt` dans un répertoire `api` puis, dans votre répertoire personnel, créez un lien `api.txt` vers ce fichier.

Indice :

On utilise successivement les commandes `mkdir`, `touch`, et `ln`.

XII Rechercher des fichiers

Objectifs

- Savoir utiliser la commande `locate`
- Savoir utiliser la commande `find`

locate et find

Il y a des centaines de milliers de fichiers sur une distribution Linux classique. Il peut être difficile de s'y retrouver. `locate` et `find` permettent de s'y retrouver.

`locate` recherche des fichiers à partir d'une base de données du contenu de l'arborescence pré-remplie

`find` recherche des fichiers dans l'arborescence directement, éventuellement avec des filtres précis.

La commande locate

`locate motif` recherche le motif dans la base de données `/var/lib/mlocate/mlocate.db`. Si `locate` ne renvoie rien :

- rien n'a été trouvé
- la base de données est vide

```
1 $ locate issue
2 /etc/issue
3 /etc/issue.net
4 [...]
```

Initialiser la base de données pour locate

 Méthode

La commande `updatedb` permet d'initialiser la base de données utilisée par `locate`. Il faut l'exécuter en utilisateur root avec la commande `sudo`.

```
1 $ sudo updatedb
```

La base données n'est pas mise à jour en temps réel, mais elle est régulièrement mise à jour par le système.

La commande find

`find` recherche dans un dossier donné tous les fichiers qu'il contient en fonction de critères de recherches spécifiques. On peut filtrer la recherche :

- par nom de fichier,
- par taille,
- par date de modification,
- par type,
- et par bien d'autres critères listés dans le manuel (`man find`).

Il suffit de préciser le point de départ de la recherche et ce qu'on cherche.

```
1 $ find . -name "chats*"
2 ./chats
3 ./chats/chats.jpg
```

Ici on recherche dans le répertoire courant et tous ses sous-répertoires les fichiers et répertoires dont le nom débute par « *chats* ».

```
1 $ find . -name "chats*" -type d
2 ./chats
```

Ici on recherche tous les fichiers de type "répertoire" (d, pour *directory* en anglais) qui de plus commencent par le mot « *chats* ».

Exécution de commandes par find

⊕ Complément

find permet d'exécuter une commande sur chaque fichier trouvé avec l'option -exec.

```
1 $ find . -name "photo*" -exec cp {} {}.bak \;
```

- cp est la commande exécutée
- {} est remplacé par le nom du fichier trouvé
- \; indique la fin de la commande.

Ici, on crée une copie de chaque fichier trouvé en ajoutant .bak au nom de fichier.

XIII Exercice : Appliquer la notion

On souhaite connaître la liste de tous les fichiers sur le système dont la taille dépasse 10 Mio (Mébioctets, soit $1024 * 1024$ octets).

Question

[solution n°11 p. 33]

Donnez la commande pour réaliser cette opération.

Indice :

On utilisera `find` avec les bonnes options. Vous pouvez consulter le manuel pour les trouver.

XIV Afficher des fichiers

Objectifs

- Savoir afficher tout ou partie des fichiers dans la console

Afficher des fichiers

Nous savons comment ouvrir des fichiers avec Nano pour les éditer directement dans la console. On pourrait donc se demander l'intérêt d'utiliser d'autres commandes pour afficher le contenu d'un fichier dans la console, puisque l'on peut le consulter avec Nano. Cependant cet outil reste un éditeur de texte, il est prévu pour éditer des fichiers et possède une interface qui lui est propre. Avoir une commande pour afficher le contenu d'un fichier **et c'est tout** permet de facilement récupérer le contenu d'un fichier pour le manipuler (même si l'on ne sait pas encore comment faire des scripts).

La commande cat

 Méthode

On peut utiliser `cat` (abréviation de *concatenate*) pour afficher le contenu d'un fichier dans le terminal.

```
1 kyane@europa:~/librecours$ cat roman.txt
2 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ornare vitae sem
  sed cursus. Maecenas sed mi eget ligula ullamcorper dapibus.
3 Nullam eu elit non dui maximus vulputate. Ut dolor est, commodo eu finibus nec,
  imperdiet et lacus. Morbi at mauris eros. Nullam pharetra aliquet mauris id
  finibus.
4
```

Comme son nom l'indique, la commande peut servir à afficher (et donc concaténer) plusieurs fichiers.

```
1 kyane@europa:~/librecours$ cat roman.txt suite.txt
2 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ornare vitae sem
  sed cursus. Maecenas sed mi eget ligula ullamcorper dapibus.
3 Nullam eu elit non dui maximus vulputate. Ut dolor est, commodo eu finibus nec,
  imperdiet et lacus. Morbi at mauris eros. Nullam pharetra aliquet mauris id
  finibus.
4 Pellentesque lectus dui, iaculis ac lorem in, vehicula porttitor urna. Mauris
  tincidunt, neque et rutrum placerat, leo dolor tempus felis, ac accumsan odio
  lacus nec nisl.
5 Nullam vulputate mattis aliquam. Vestibulum cursus dictum metus.
6 In iaculis orci eget mauris maximus, non mollis augue lacinia. Nulla viverra
  lorem lectus, vel cursus odio mollis quis. Aenean iaculis dui pellentesque,
  gravida turpis at, porta nibh.
7
```

 Exemple

On se rend compte cependant que `cat` est adapté lorsque l'on veut visualiser de petits fichiers. Essayons avec le fichier `/var/log/syslog` qui contient des messages d'information du système d'exploitation.

```
1 cat /var/log/syslog
```

Énormément de texte s'affiche dans notre console, ce n'est pas pratique à lire puisque qu'il faut remonter tout en haut du fichier.

La commande less



Pour nous aider à visualiser le fichiers qui sont très grand, on utilise la commande `less`. Cet outil ouvre une interface très basique qui permet de visualiser le fichier et de se déplacer dedans.

[illegible]

On peut utiliser quelques commandes :

- parcourir le fichier à l'aide des flèches Haut/Bas, mais aussi faire de plus grand sauts avec les flèches Page Up/Page Down ;
- quitter l'interface avec la touche q;
- g permet de se rendre au début du fichier ;
- G permet de se rendre en fin de fichier ;
- écrire / suivi d'un mot permet de rechercher le mot dans le document.

Récupérer des portions de fichier

Afficher un fichier dans sa totalité c'est bien, mais dans certains cas on ne souhaite pas forcément récupérer tout le fichier. Par exemple pour notre fichier de log `/var/log/syslog`, on pourrait ne vouloir consulter que les 20 dernières lignes, pour voir les 20 événements les plus récents (chaque message supplémentaire de l'OS est inscrit à la fin du fichier). On peut utiliser `cat` et ne regarder que la fin, mais ce n'est pas très pratique si le fichier est très gros `cat` mettra du temps à l'afficher, et si l'on ne veut consulter que les premières lignes à l'inverse on est toujours embêté.

tail



La commande `tail` (qui signifie *queue* en anglais), permet justement de n'afficher que les dernières lignes d'un fichier.

```
1 kyane@europa:~/librecours$ tail /var/log/syslog
2 Nov 27 14:40:42 europa org.freedesktop.thumbnails.Thumbnailer1[20118]:
  Registered thumbnailer evince-thumbnailer -s %s %u %o
3 Nov 27 14:40:42 europa org.freedesktop.thumbnails.Thumbnailer1[20118]:
  Registered thumbnailer dia -t png -e %o -s %s %i
4 Nov 27 14:40:42 europa org.freedesktop.thumbnails.Thumbnailer1[20118]:
  Registered thumbnailer /usr/bin/gdk-pixbuf-thumbnailer -s %s %u %o
5 Nov 27 14:40:42 europa org.freedesktop.thumbnails.Thumbnailer1[20118]:
  Registered thumbnailer atril-thumbnailer -s %s %u %o
6 Nov 27 14:40:42 europa org.freedesktop.thumbnails.Thumbnailer1[20118]:
  Registered thumbnailer /usr/bin/gdk-pixbuf-thumbnailer -s %s %u %o
7 Nov 27 14:40:43 europa dbus-daemon[1976]: [session uid=1000 pid=1976]
  Successfully activated service 'org.freedesktop.thumbnails.Cache1'
8 Nov 27 14:40:43 europa dbus-daemon[1976]: [session uid=1000 pid=1976]
  Successfully activated service 'org.freedesktop.thumbnails.Thumbnailer1'
9 Nov 27 14:40:43 europa tumblerd[20118]: Name org.freedesktop.thumbnails.Cache1
  lost on the message dbus, exiting.
10 Nov 27 14:40:43 europa tumblerd[20118]: Name org.freedesktop.thumbnails.Manager1
  lost on the message dbus, exiting.
11 Nov 27 14:40:43 europa tumblerd[20118]: Name
  org.freedesktop.thumbnails.Thumbnailer1 lost on the message dbus, exiting.
12
```

Par défaut la commande ne retourne que les 10 dernières lignes, mais on peut utiliser le paramètre `-n` pour en obtenir un nombre précis.

```
1 kyane@europa:~/librecours$ tail -n 5 /var/log/syslog
2 Nov 27 14:40:43 europa dbus-daemon[1976]: [session uid=1000 pid=1976]
  Successfully activated service 'org.freedesktop.thumbnails.Cache1'
3 Nov 27 14:40:43 europa dbus-daemon[1976]: [session uid=1000 pid=1976]
  Successfully activated service 'org.freedesktop.thumbnails.Thumbnailer1'
4 Nov 27 14:40:43 europa tumblerd[20118]: Name org.freedesktop.thumbnails.Cache1
  lost on the message dbus, exiting.
5 Nov 27 14:40:43 europa tumblerd[20118]: Name org.freedesktop.thumbnails.Manager1
  lost on the message dbus, exiting.
6 Nov 27 14:40:43 europa tumblerd[20118]: Name
  org.freedesktop.thumbnails.Thumbnailer1 lost on the message dbus, exiting.
7
```

head



L'opposée de `tail` est la commande `head` (qui signifie *tête*) qui peut-être utilisée de la même manière, mais pour afficher le début d'un fichier.

```
1 kyane@europa:~/librecours$ head -n 5 /var/log/syslog
2 Nov 27 08:14:53 europa blueman-mechanism[701]: Unable to init server: Could not
  connect: Connection refused
3 Nov 27 08:14:53 europa systemd[1]: logrotate.service: Succeeded.
4 Nov 27 08:14:53 europa systemd[1]: Finished Rotate log files.
5 Nov 27 08:14:53 europa containerd[858]: time="2020-11-
  27T08:14:53.758802087+01:00" level=info msg="loading plugin
  \\"io.containerd.content.v1.content\\"..." type=io.containerd.content.v1
6 Nov 27 08:14:53 europa systemd[1]: Started User Login Management.
7
```

À retenir

Pour afficher des fichiers dans la console, on utilise des commandes dédiées plutôt que d'ouvrir le fichier avec Nano. On utilise `cat` plutôt pour les petits fichiers, et `less` pour les plus grands. Pour récupérer uniquement le début ou la fin d'un fichier, on s'appuie sur les commandes `head` et `tail`.

XV Exercice : Appliquer la notion

Nous allons manipuler un fichier texte contenant un Lorem ipsum : un texte latin aléatoire utilisé par organiser le texte d'une page Web lors des phases de développement. Pour cela téléchargez le fichier suivant et placez le dans un dossier de votre choix.

(cf. loremipsum.txt)

Question 1

[solution n°12 p. 34]

Sans utiliser Nano, visualisez le fichier et trouvez la première occurrence du mot bibendum.

Indice :

On va s'appuyer sur la commande `less`.

Question 2

[solution n°13 p. 34]

Comment récupérer les 2 dernières lignes du fichier dans notre terminal ?

XVI Quiz

Exercice 1 : Quiz - Culture

[solution n°14 p. 34]

Exercice

Sous Linux, quel est le caractère utilisé comme séparateur pour les chemins de fichier ?

☒ A /

☐ B \

☐ C \$

☐ D ~

Exercice

Qu'est-ce que la racine du système de fichier sous Linux ?

☒ A C'est le disque dur principal utilisé pour stocker les fichiers

☐ B C'est le nom du dossier le plus haut dans l'arborescence, le dossier / qui contient tout les autres dossiers.

☐ C C'est la langue utilisée dans le terminal

Exercice

Parmi ces propositions, lesquelles sont des chemin relatifs ?

☒ A photos/vacances

☐ B /etc/ssh

☐ C /var/log

☐ D ../var/log

Exercice

Sous Linux, quel est le caractère utilisé pour désigner le dossier de l'utilisateur courant ?

☒ A /

B

\

C

\$

D

~

Exercice 6 : Quiz - Méthode

[solution n°15 p. 36]

Exercice

Quelle commande devriez vous utiliser pour voir les fichiers cachés ?

Exercice

Comment créer un fichier avec le nom "monfichier" ?

Exercice

Comment créer un dossier nommé "photos" ?

Exercice

Quelle est la touche à utiliser pour quitter l'interface de less ?

Exercice

Quel paramètre utiliser avec la commande cp pour copier un répertoire entier ?

Exercice

Comment renommer le dossier "photos" en "images" ?

Solutions des exercices

Solution n°1

[exercice p. 6]

```
1 kyane@europa:~$ cd /var/log/
```

Solution n°2

[exercice p. 6]

```
1 kyane@europa:/var/log$ ls -F
2 alternatives.log      apache2/      btmp          debug          mail.err
   auth.log            dpkg.log      lastlog       messages       syslog
```

Bien entendu les fichiers affichés peuvent varier en fonction de la machine.

Solution n°3

[exercice p. 10]

On se retrouve dans son dossier d'utilisateur.

```
1 kyane@europa:/etc/cron.d$ cd
2 kyane@europa:~$ pwd
3 /home/kyane
```

Solution n°4

[exercice p. 10]

Le dossier dans lequel on se retrouve est `/var/cache/apt`, car on remonte d'un dossier puis on descend dans `cache/apt`.

On commence par se positionner dans le dossier de départ.

```
1 kyane@europa:~$ cd /var/log
```

Puis on se rend dans le dossier de destination.

```
1 kyane@europa:/var/log$ cd ../cache/apt/
2 kyane@europa:/var/cache/apt$
3
```

On constate dans le prompt (mais on peut utiliser la commande `pwd`) que le chemin absolu est bien `/var/cache/apt`.

Solution n°5

[exercice p. 14]

```
1 mkdir -p documents/finances images
```

Solution n°6

[exercice p. 14]

On crée les 2 fichiers

```
1 kyane@europa:~/librecours$ touch documents/todo.txt
2 kyane@europa:~/librecours$ touch documents/cv.tex
```

Puis on liste le contenu du dossier.

```
1 kyane@europa:~/librecours$ ls -F documents/
2 cv.tex  finances/  todo.txt
```

Solution n°7

[exercice p. 14]

On utilise Nano pour créer et ouvrir le fichier

```
1 kyane@europa:~/librecours$ nano documents/finances/compta.txt
```

Après avoir écrit quelques lignes, on utilise `Ctrl + o` pour enregistrer le fichier, et `Ctrl + x` pour quitter Nano

Solution n°8

[exercice p. 18]

```
1 $ mv imgaes images
```

Solution n°9

[exercice p. 18]

La méthode la plus rapide est d'indiquer plusieurs paramètres à la commande `cp` : le premier pour récupérer toutes les photos de Noël, le second pour prendre la "photo 1" des vacances, le dernier pour spécifier la destination des 2 premiers paramètres.

```
1 $ cp images/noel-2018/*.png images/vacances-2019/photo1.jpg images/a-developper/
```

Mais il était aussi possible de déplacer les fichiers un par un, c'est beaucoup plus fastidieux et c'est là qu'on voit la puissance de la ligne de commande (et des métacaractères).

Solution n°10

[exercice p. 21]

```
1 bonnetst@vivaldi:~$ mkdir api
2 bonnetst@vivaldi:~$ touch api/api.txt
3 bonnetst@vivaldi:~$ ln -s api/api.txt
4 bonnetst@vivaldi:~$ ls -l api.txt
5 lrwxrwxrwx 1 bonnetst bonnetst 11 janv. 25 21:31 api.txt -> api/api.txt
```

Solution n°11

[exercice p. 24]

Il suffit d'utiliser la commande `find` avec le filtre de taille `-size` et l'option de taille `+10M` :

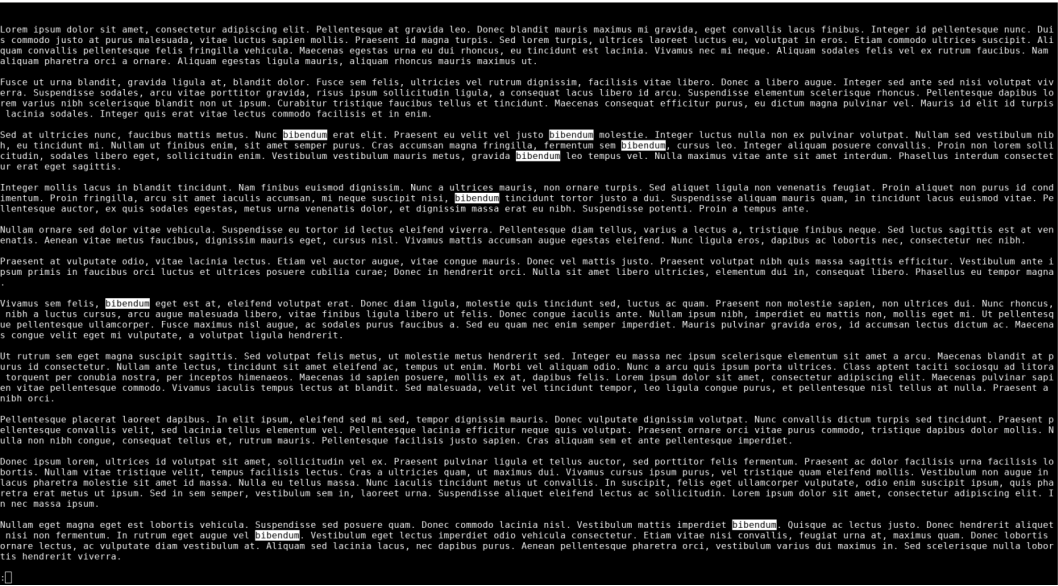
```
1 $ find / -size +10M
```

Solution n°12

On ouvre le fichier avec less pour le visualiser.

```
1 $ less loremipsum.txt
```

Une fois dans l'interface, on tape /bibendum pour trouver les occurrences du mot.



Solution n°13

On utilise la commande tail.

```
1 $ tail -n 2 loremipsum.txt
2
3 Pellentesque quis blandit erat. Proin elit urna, imperdiet a vehicula non,
consectetur eget urna. Aenean sit amet nunc leo. Etiam in porta orci, quis
condimentum justo. Pellentesque pulvinar facilisis odio, sed posuere nunc porttitor
at. Praesent vitae urna at arcu malesuada finibus. Mauris pulvinar lectus eu
venenatis laoreet. Praesent nec nisl nec lectus ornare consectetur. Nunc id augue
sit amet enim gravida pellentesque. Cras porttitor ornare varius. Cras blandit velit
nec mi tempus luctus a eu lectus. Sed id ex sed augue convallis consectetur at at
nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per
inceptos himenaeos. Phasellus imperdiet libero et suscipit auctor
```

Solution n°14

Exercice

Sous Linux, quel est le caractère utilisé comme séparateur pour les chemins de fichier ?

A

/

B

\

C

\$

D

~



En effet on écrit les chemins sous la forme `/home/kyane/hello.txt`

Exercice

Qu'est-ce que la racine du système de fichier sous Linux ?

A C'est le disque dur principal utilisé pour stocker les fichiers



C'est le nom du dossier le plus haut dans l'arborescence, le dossier `/` qui contient tout les autres dossiers.

C C'est la langue utilisée dans le terminal

Exercice

Parmi ces propositions, lesquelles sont des chemin relatifs ?



A `photos/vacances`

B `/etc/ssh`

C `/var/log`



D `../var/log`



Les chemins relatifs sont ceux qui dépendent du contexte, et qui ne commence donc pas à la racine (donc pas par un `/`)

Exercice

Sous Linux, quel est le caractère utilisé pour désigner le dossier de l'utilisateur courant ?

A `/`

B `\`

C `$`



D `~`



En effet on utilise le `"~"`, par exemple `~/Documents` désigne le dossier Documents qui se trouve dans mon dossier d'utilisateur.

Solution n°15

Exercice

Quelle commande devriez vous utiliser pour voir les fichiers cachés ?

ls -a

Exercice

Comment créer un fichier avec le nom "monfichier" ?

touch monfichier

Exercice

Comment créer un dossier nommé "photos" ?

mkdir photos

Exercice

Quelle est la touche à utiliser pour quitter l'interface de less ?

q

Exercice

Quel paramètre utiliser avec la commande cp pour copier un répertoire entier ?

-r

Exercice

Comment renommer le dossier "photos" en "images" ?

mv photos images

Crédits des ressources

Extrait du roman Traces (Stéphane Crozat, Framabook, 2018) p. 13

Universel - Transfert dans le Domaine Public - Kyâne Pichou

p. 26

Universel - Transfert dans le Domaine Public - Kyâne Pichou

