N. Ferry, H. Renard, S. Lavirotte, D. Lopez
Polytech'Nice – Sophia
PeiP 1
2019-2020

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

Ce TD est destiné à vous familiariser avec vos premières commandes. Fortement axé sur la pratique, il aborde la notion de répertoire et de fichier ainsi que les commandes pour la création, manipulation et suppression de ces entités sous l'environnement UNIX.

1 Environnement de travail

Il est souvent utile de savoir utiliser un ordinateur à l'aide de commandes au lieu de cliquer. C'est souvent beaucoup plus efficace. Vous allez donc apprendre à utiliser un interpréteur de commandes dans un terminal.

1.1 Interpréteur de commandes

Mais qu'est-ce que c'est un interpréteur de commandes ? C'est un programme pour exécuter des commandes tapées au clavier par un utilisateur dans un terminal. Les interpréteurs de commandes sont aussi appelés « *Shells* » (coquilles en anglais). Il existe de nombreux interpréteurs de commandes : `sh`, `bash`, `csh`, `ksh`, `zsh`, qui ont chacun des particularités. Dans ce cours, nous exposerons les concepts communs à tous les interpréteurs de commandes et nous ne nous intéresserons pas aux particularités de chacun.

Les commandes sont tapées dans un terminal en mode texte, constitué d'une fenêtre dans un environnement graphique, ou d'une console sur un écran en texte seul (sans environnement graphique). Pour vous indiquer que l'interpréteur de commandes est prêt à recevoir une commande, il affiche un message en début de ligne appelé ***prompt*** en anglais. Ce prompt est généralement constitué du nom de l'utilisateur puis du nom de la machine et enfin du nom du dossier où vous vous trouvez, souvent suivi du caractère `$`.

```
utilisateur@machine:chemin$
```

Dans ce support de cours, nous simplifierons le prompt au simple affichage du symbole `$` en début de ligne. Cela permettra de distinguer la commande que l'on tape, du résultat que l'on obtient.

Les résultats sont aussi affichés sur le terminal. Une sortie graphique n'est pas obligatoire (pour la plupart des commandes la sortie est textuelle). Un interpréteur de commandes analyse et exécute les commandes que vous tapez. Mais il peut aussi exécuter automatiquement une suite de commandes que vous avez programmée dans un « *script Shell* ». Mais nous verrons cela plus tard. Pour le moment, apprenons à utiliser les commandes.

1.2 Commande

Une commande Unix est un mot ou une phrase à la syntaxe bien particulière entrée dans un interpréteur de commandes et donnant l'ordre d'actions à exécuter par l'ordinateur. Une commande UNIX se décompose en 3 parties :

- la commande elle-même (c'est toujours le premier élément sur la ligne),
- des options commençant par le symbole moins « - » qui, comme le nom l'indique, sont optionnelles (zéro, une ou plusieurs options),
- des arguments : zéro, un nombre fixe ou variable d'arguments.

```
$ commande -option1 -option2 -option3 argument1 argument2
```

Par exemple, pour connaître la date, tapez dans un terminal la commande `date`; le système vous afficher un texte contenant la date du jour et l'heure. Dans ce cas, la commande est utilisée sans option ni argument.

Si nous prenons l'exemple de la commande `cal`, suivant son utilisation, vous pourrez avoir plusieurs résultats à partir de cette même commande :

- `cal` : commande sans option ni argument, elle imprimera le calendrier du mois courant
- `cal -y` : avec l'option `-y`, elle imprimera les calendriers de l'année entière (year)
- `cal 2018` : avec l'argument `2018`, elle imprimera le calendrier de l'année spécifiée (2018)

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

Dans les exemples précédents, le comportement de la commande `cal` est modifié en fonction des options ou des arguments qui lui sont passés.

La commande, les options et les arguments sont séparés par au moins un espace. Il est possible d'accéder aux commandes précédemment tapées grâce à la flèche vers le haut.

Mais pourquoi utiliser un terminal et taper des commandes ? Vous verrez dans la suite de ce cours tout l'intérêt d'un interpréteur de commandes. En effet, l'énorme avantage, quand on le maitrise, est que l'on peut appliquer une commande à un nombre très important de fichiers ou bien enchainer plusieurs commandes pour réaliser des opérations complexes. En effet, il n'est pas réaliste de faire certaines opérations à l'aide de la souris (ou alors vous perdez votre temps). De plus, certaines fois, c'est le seul moyen d'interaction avec la machine. Quand on n'a pas un ordinateur de bureau, un portable ou une tablette mais un objet informatique sans écran ou un serveur à distance, souvent, seules les commandes sont utilisables pour interagir avec le système.

1.3 Aide sur les commandes

Il est bien sûr impossible de connaître toutes les commandes par cœur, ni toutes les options et arguments possibles pour chacune des commandes. Heureusement, UNIX est bien fait et a tout prévu.

1.3.1 A propos

Si vous désirez connaître les commandes qui concernent un sujet particulier, vous pouvez utiliser la commande `apropos`. Par exemple, si vous souhaitez connaître toutes les commandes qui traitent du calendrier :

```
$ apropos calendar
```

1.3.2 Manuel des commandes

Pour tout connaître sur une commande, une page de manuel est fournie avec le système. Ces pages sont souvent disponibles dans plusieurs langues, mais la page en anglais est toujours la référence ; les autres ne sont que des traductions de plus ou moins à jour et de bonne qualité. Pour tout savoir sur une commande :

```
$ man commande
```

Ce n'est pas toujours drôle de devoir lire le descriptif d'une commande, mais c'est le passage obligé, même pour les plus chevronnés. Une expression anglaise résume bien la situation : RTFM (*Read The Fucking Manual*) ! Vous n'y échapperez pas.

1.3.1 Comprendre une commande avec ses options

Un site Web permet de réaliser l'opération inverse qui consiste à savoir ce que fait une commande en fonction des options et arguments fournis. Cela peut être utile pour vous aider dans un premier temps :

http://explainshell.com/

Et maintenant que vous avez les informations minimums, commençons !

2 Système de fichiers

2.1 Définition

L'ensemble des répertoires et fichiers constitue le **système de fichiers** (en anglais *File System*).

N. Ferry, H. Renard, S. Lavirotte, D. Lopez
Polytech'Nice – Sophia
PeiP 1
2019-2020

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

Illustration : Imaginez qu'une grande commode soit votre système de fichiers. Il contient des tiroirs (les répertoires) dans lesquels pourraient se trouver des documents (les fichiers) mais aussi d'autres tiroirs (des sous-répertoires). Un répertoire peut donc contenir :

- des fichiers (les papiers rangés dans un tiroir de la commode);
- d'autres répertoires (des tiroirs).

Mais la commande elle-même a un rôle un peu spécial par rapport aux tiroirs, elle les contient tous : elle ne peut pas se trouver elle-même dans un tiroir !

Dans le cas de l'informatique, on appelle la commode, la **racine** du système de fichiers (en anglais *Root Directory*) : il s'agit de l'entité de plus haut niveau, car elle peut contenir des fichiers ou des répertoires mais ne peut pas se trouver elle-même dans un répertoire ! C'est la première entité du système de fichiers.

Donc le système de fichiers est organisé hiérarchiquement sous la forme d'un arbre (une structure arborescente) et il est constitué de deux éléments : les répertoires et les fichiers. Les répertoires sont les nœuds de l'arborescence et les fichiers sont situés dans les répertoires. Le système de fichier sous Unix dispose d'une racine unique (contrairement à Windows) qui notée `/`. Cette organisation permet d'unifier la manipulation des fichiers et répertoires indépendamment de l'organisation matérielle, c'est-à-dire des supports physiques sur lesquels sont stockés ces fichiers (disques durs locaux ou d'une autre machine sur le réseau, lecteur de CD ou DVD, clé USB, etc...).

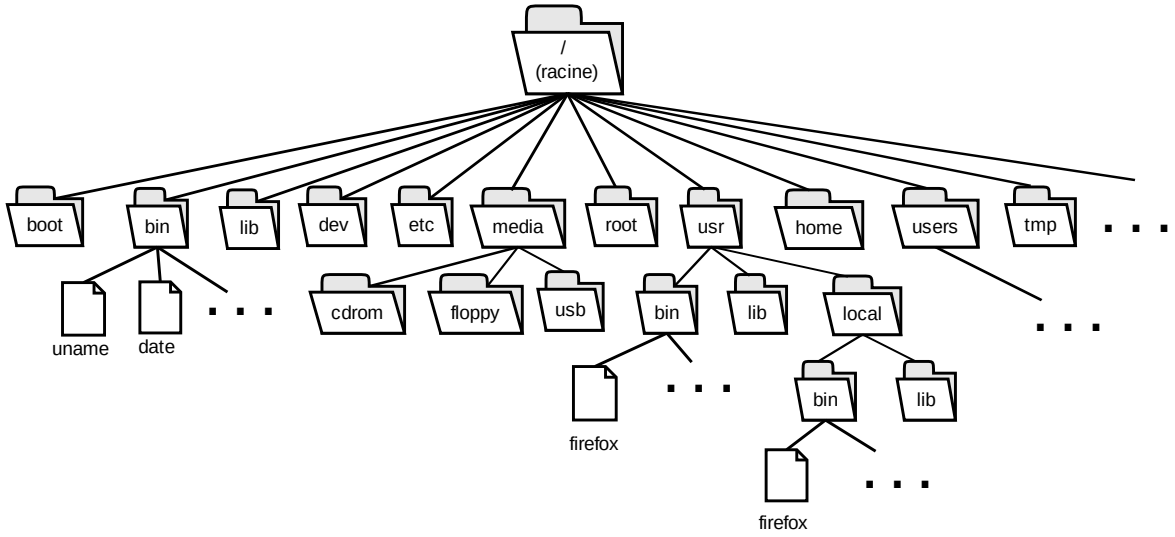


Figure 1: Exemple d'organisation d'un système de fichiers (source : Univ Grenoble Alpes, IUT2)

En résumé :

- **Arborescence** : Structure hiérarchisée des répertoires et des sous-répertoires. Sur les représentations graphiques (comme celle-ci-dessus), l'arbre est toujours représenté « à l'envers ». La racine de l'arbre est en haut et toute les branches (répertoires) partent de cette racine puis se subdivisent en sous-branches (les sous-répertoires), et ainsi de suite. Les fichiers sont rangés dans les répertoires.
- **Racine** : Point de départ d'une arborescence. Sous UNIX, elle est notée `/` (signe divisé du clavier, prononcé, *slash* en anglais) et est unique. Sous Windows, elle peut être multiple (il y a plusieurs systèmes de fichiers) et est notée sous la forme d'une lettre suivie de deux points : `C:`, `D:`, ...
- **Répertoire** : Une liste de descriptions de fichiers. Du point de vue du système de fichiers, il est traité comme un fichier. Un répertoire a donc les mêmes types de propriétés qu'un fichier comme le nom, la taille, la date, les droits d'accès et les divers autres attributs que nous étudierons plus tard dans ce cours.

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

2.2 Visualisation des répertoires et fichiers

La commande `ls` (abréviation du mot anglais *list*) vous permet de visualiser le contenu du système de fichiers. Elle affiche les fichiers et le nom des sous-répertoires qui se trouvent dans un répertoire. De nombreuses options sont disponibles pour cette commande. Voici les plus courantes :

Option	Objectif
`-a` (abréviation de mot « all » : tous)	Affiche tous les fichiers en incluant ceux qui commencent par un point (i.e. les fichiers cachés)
`--color`	Affiche en couleur suivant le type des éléments
`-l` (abréviation de « long »)	Affiche en format long (type, date, taille, propriétaire, permissions)
`-R` (abréviation de « récursive »)	Affiche les contenus des sous-répertoires
`-S` (abréviation de « size » : taille)	Liste les fichiers par taille (les fichiers les plus gros en premier)
`-t` (abréviation de « temps »)	Liste les fichiers selon la date de la dernière modification
`-u` (abréviation de « update » : mise à jour)	Liste les fichiers selon la date du dernier accès
`-r` (abréviation de « reverse » : inversé)	Affiche les fichiers en ordre inverse

Ces options peuvent être utilisées individuellement ou combinées. Par exemple, la commande suivante affichera les noms des fichiers et dossiers en format long, en ordre de modification chronologique inverse.

```
$ ls -ltr
```

Vous noterez qu'il est possible de contracter les options qui ne font qu'une lettre (on n'a pas tapé : `ls -l -t -r`). Dans le tableau précédent sur les options de la commande `ls`, vous noterez aussi que l'option `color` débute par deux tirets « `--` ». Ce n'est pas une erreur. Cela permet de distinguer l'option `color` de la concaténation d'options à une lettre que pourrait être `-c -o -l -o -r`.

3 Répertoire

Un répertoire (ou dossier) est donc un objet informatique qui contient des fichiers. Pour être plus précis, sous Unix, un répertoire est lui-même un fichier contenant la liste des fichiers et des sous-répertoires qu'il contient, mais nous reviendrons sur ce point un peu plus tard.

Répertoire personnel : Il y a un répertoire un peu particulier, c'est le vôtre (associé à votre compte utilisateur). Il est dénommé en anglais « *home directory* » ou « *homedir* ». C'est l'équivalent du répertoire « Mes documents » sous Windows.

3.1 Nom des répertoires et noms « spécifiques »

Chaque répertoire dispose d'un nom, comme pour les fichiers. Depuis le début d'Unix, aucune limitation majeure n'est faite pour le nom d'un fichier ou d'un répertoire. Tout caractère, y compris les espaces, peut être utilisé dans le nom. Mais attention, les majuscules et les minuscules ne sont pas identiques. Donc le nom *Directory* et *directory* sont deux noms différents.

Certains symboles sont utilisés pour désigner des répertoires particuliers.

3.1.1 Le répertoire courant : .

Le **répertoire courant**. « . » désigne le répertoire dans lequel vous vous trouvez. Il peut être utilisé pour une commande lorsque l'on a besoin de spécifier le répertoire où l'on se trouve.

3.1.2 Le répertoire parent : `..`

Le **répertoire parent**. « `..` » désigne le répertoire qui se trouve juste au-dessus dans l'arborescence ; c'est le parent, celui d'où l'on vient. Par exemple, sur la Figure 1, le répertoire parent du répertoire `cdrom` est le répertoire `media`.

3.1.3 Le répertoire personnel : `~`

Le caractère « ~ » (`AltGr-2`) permet de désigner votre répertoire personnel. Les interpréteurs de commande le remplacent par le nom du répertoire de l'utilisateur courant. Mais attention, ~ pour vous, ne désigne pas le même répertoire pour une autre utilisateur.

3.1.4 Le répertoire personnel d'un utilisateur particulier : `~helene` (ou `~login`)

De façon analogue, il est remplacé par l'interpréteur de commandes par le répertoire personnel de l'utilisateur `helene`.

3.2 Chemin

Pour identifier de façon non ambiguë une entrée (un répertoire ou un fichier) que l'on veut manipuler par une commande, il faut la désigner par son nom précédé par le chemin qui permet d'y accéder au sein du système de fichiers. Le nom seul ne suffit pas puisque deux entrées situées dans deux répertoires différents peuvent tout à fait avoir le même nom. Par exemple sur la Figure 1, il y a deux fichiers qui s'appellent `firefox`. Un chemin désigne donc le répertoire père de l'entrée.

Pour construire ce chemin, on énumère la liste des répertoires qu'il faut traverser au sein du système de fichiers pour atteindre l'entrée. Il y a deux points de départ possibles pour ce chemin :

- le répertoire racine (`/`) : on parle alors de **chemin absolu**
- le répertoire courant : on parle alors de **chemin relatif**

Les noms des différents répertoires qui composent un chemin seront donc énumérés, séparés par le caractère `/` (le même que celui qui désigne le dossier racine). Par exemple, si je me trouve dans le répertoire `local` (qui se trouve dans le répertoire `usr`, lui-même à la racine sur la Figure 1) et que je veux désigner le fichier `date` (qui contient le code du programme `date` que nous avons utilisé tout au début), je peux le faire avec :

- un chemin absolu : `/bin/date`
- un chemin relatif : `../../bin/date`

On peut toujours spécifier un emplacement par l'un ou l'autre des chemins. L'utilisation de l'un ou de l'autre type de chemin dépend souvent de la longueur de celui-ci (et donc du nombre de caractères à taper).

Tout chemin qui débute par « / » est un chemin absolu. Un chemin qui commence par « ~ » est un chemin absolu pour un utilisateur donné (un autre utilisateur arrivera bien sûr dans son propre dossier).

En résumé :

- **Chemin** (« `path` ») : Séquence de répertoires imbriqués (séparés par le caractère `/`) désignant une entrée unique (un fichier ou un répertoire) sur le système de fichiers.

3.3 Commandes manipulant les répertoires

3.3.1 `cd` : se déplacer dans l'arborescence de répertoires

Pour se déplacer à l'intérieur de l'arborescence des répertoires, il faut utiliser la commande `cd`. Elle s'utilise en spécifiant en argument le nom du répertoire qui doit devenir celui courant pour les actions suivantes.

```
$ cd répertoire
```

Faites bien attention de séparer par un espace « `cd` » et le nom du répertoire. UNIX exige une grande précision dans la syntaxe des commandes. Soumettez la commande au système grâce à la touche « Entrée », évidemment !

Le répertoire peut être indiqué de manière absolue (par rapport à la racine en commençant le chemin par `/`) ou relative (par rapport au répertoire courant). On peut aussi utiliser « `..` » qui désigne le répertoire parent d'un répertoire ou « `.` » qui désigne le répertoire courant. Par exemple pour remonter dans l'arborescence, on utilise :

```
$ cd ..
```

La commande `cd` peut être utilisée avec l'argument « `-` » (moins) qui désigne le répertoire où l'on se trouvait précédemment. Mais cela ne se fait que sur un seul niveau de mémorisation. Une fois revenu dans le répertoire précédent, `-` désigne celui d'où on vient.

```
$ cd -
```

3.3.2 `pwd` : afficher le chemin du répertoire où je me trouve

Si vous ne savez plus dans quel répertoire vous vous trouvez, utilisez la commande `pwd`. Elle affichera le chemin absolu du répertoire dans lequel vous vous trouvez.

3.3.3 `mkdir` : créer un répertoire

La commande `mkdir` permet de créer un (ou plusieurs) répertoires. Il suffit de spécifier le nom du (ou des) répertoire(s) que vous souhaitez créer en argument de la commande.

```
$ mkdir rep1 rep2
```

Cette commande créera les deux répertoires `rep1` et `rep2` dans le répertoire courant (où est exécutée la commande). Vous pouvez bien entendu spécifier un chemin spécifique où créer `rep1` ou `rep2`. Par exemple :

```
$ mkdir rep1 /tmp/rep2
```

Cette commande créera le répertoire `rep1` dans le dossier courant et `rep2` dans le dossier `tmp` qui se trouve à la racine.

3.3.4 `rmdir` : supprimer un répertoire

Pour supprimer un répertoire, utilisez la commande `rmdir`. Cette commande ne permet de supprimer que des répertoires vides (qui ne contiennent aucun fichier et aucun sous-répertoire). Nous verrons que la commande `rm` (permettant de supprimer des fichiers) permet aussi de supprimer un répertoire et tout ce qu'il contient, si on lui ajoute les bonnes options.

4 Fichiers

Un fichier contient une suite d'informations binaires, c'est-à-dire une suite de `0` et de `1`. Un fichier texte est un fichier composé de caractères stockés sous la forme de valeurs numériques codées en binaire. Vous n'y comprenez rien ?? C'est normal ! Nous verrons cela plus tard dans le cours sur le codage de l'information. Sachez toutefois pour le moment qu'un fichier peut contenir des données (qui peuvent être de différents types : texte, image, son, vidéo, ...) ou des programmes (suite d'instruction à exécuter par la machine).

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

4.1 Type de données dans un fichier, Type de fichiers

Un fichier est enregistré sur le système de fichiers sous la forme « `nom_du_fichier.ext` ». « `.ext` » représente l'extension c'est un moyen d'identifier le type de données dans un fichier et de lui associer le bon programme pour interpréter ces données (attention cela ne garantit pas le type de données du fichier : on peut changer l'extension du nom de fichier, cela ne change pas les données du fichier pour autant ! Mais nous reviendrons sur ce point ultérieurement dans le cours).

Lorsque vous faites la commande `ls` dans un répertoire avec l'option `-l`, vous obtenez la liste détaillée des fichiers contenus dans un répertoire. Pour savoir de quel type est un fichier, il suffit de regarder la première lettre de la ligne. Pour les autres données, nous verrons cela ultérieurement.

Désignation	Type	Description
`-`	Fichier standard	C'est un fichier ordinaire tel qu'un fichier texte ou un programme
`d`	*Directory* / Répertoire	C'est l'élément qui contient les fichiers, en référence au dossier d'un système d'exploitation.
`l`	*Link* / Lien	Un élément qui est soit un lien hard, soit un lien soft.

Il existe aussi d'autres types de fichiers sous Unix (`b` et `c` par exemple), mais nous n'étudierons pas ceux-ci dans le cadre de ce cours.

Il faut qu'un répertoire existe déjà pour y ranger un fichier !

4.2 Fichiers cachés

Lorsque vous utilisez la commande `ls`, celle-ci n'affiche pas tous les fichiers et répertoires ; elle masque certains fichiers et répertoires. En fait, les noms de fichiers ou de répertoires qui commencent par le caractère « `.` » ne sont pas affichés. On les appelle donc les fichiers cachés. Si on souhaite aussi les lister, on ajoutera l'option `-a` à la commande `ls` qui a pour but d'afficher les fichiers (et répertoires) cachés, ceux qui commencent donc par « `.` ».

Dans un répertoire vide (qui ne contient aucun fichier ou sous-répertoire), voici le résultat des commandes suivantes :

```
$ ls -l

$ ls -la
drwxrwxr-x  2 user user 4096 sept.  9 10:44 ./
drwxr-xr-x 25 user user 4096 sept.  9 10:44 ../
```

En effet, le répertoire courant et le répertoire parent commencent tout deux pas « `.` », donc ne sont pas affichés si l'option `-a` n'est pas utilisée.

4.3 Commandes manipulant les fichiers

Voici les premières commandes à maîtriser pour la manipulation des fichiers.

4.3.1 `touch` : modifier la date de dernière mise à jour d'un fichier ou le créer

La commande `touch` fixe la date de dernière modification du fichier au moment présent. Si le fichier n'existe pas, la commande crée un fichier vide.

L'autre méthode dépend du type de fichier à créer. Si c'est un fichier texte, utilisez un des éditeurs de texte, par exemple `gedit`.

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

4.3.2 `gedit` : éditer le contenu d'un fichier texte

Vous pouvez créer ou éditer un fichier texte grâce à la commande `gedit`. Vous lancez cette application en utilisant le menu Ubuntu (la spirale en haut à gauche) et en tapant son nom. Vous pourrez utiliser cette commande pour créer les fichiers textes correspondant à votre compte-rendu de TD.

4.3.3 `cp` : copier un fichiers et/ou un répertoire

La commande `cp` (abréviation du mot « `copy` ») copie un fichier existant vers un autre. Elle a deux arguments :

- le fichier à copier, qui doit exister et pouvoir être lu (nous verrons les permissions plus tard),
- le fichier résultat, qui doit être placé dans un répertoire où on a le droit d'écriture.

Si le fichier existe déjà, alors il est remplacé par un autre nom, par exemple :

```
$ cp VariableAléatoire Laplace
```

Si on met un nom de répertoire comme destination, le fichier copié a le même nom que le fichier à copier.

```
$ cp VariableAléatoire khi2/
$ ls khi2/
  VariableAléatoire
```

ou :

```
$ cd khi2/
$ cp ../VariableAléatoire    .
$ ls
  VariableAléatoire
```

C'est la même chose au répertoire près !

La commande `cp` peut avoir plusieurs arguments :

```
$ cp -r VariableAléatoire khi2/ LoiCauchy LoiBernoulli Probabilités/
```

(`cp -r` : copie récursive qui prend le répertoire et tout ce qu'il contient, y compris les autres répertoires) . On copie (duplique) toute la branche d'un arbre, i.e. tout un tiroir et ce qu'il contient !

```
$ ls Probabilités/
  VariableAléatoire   khi2/   LoiCauchy    LoiBernoulli
```

ou :

```
$ cd Probabilités/
$ cp -r ../VariableAléatoire  ../khi2/ ../LoiCauchy ../LoiBernoulli
(cp -r :  copie récursive qui prend le répertoire et tout ce qu'il contient, y compris les autres répertoires))
$ ls
  VariableAléatoire   khi2/   LoiCauchy    LoiBernoulli
```

En règle générale, la commande `cp` a au moins deux arguments :

- les `n-1` premiers sont les noms de fichiers (ou de répertoires) à copier,
- le dernier est le nom du répertoire où copier,

4.3.4 `mv` : déplacer ou renommer un fichier et/ou un répertoire

La commande `mv` (abréviation du mot « `move` ») déplace un fichier existant ou le renomme. Sous sa forme la plus simple, elle a `2` arguments :

- le fichier source, qui doit exister et pouvoir être lu et supprimé,
- le fichier destination, qui doit être dans un répertoire où on a le droit d'écriture.

Renommer un fichier, c'est le déplacer (même s'il reste dans le même répertoire) !

Comme pour la commande `cp`, `mv` peut avoir `n` arguments. Si `n` est strictement supérieur à `2`, le dernier argument est obligatoirement un répertoire (le répertoire destination où déplacer les fichiers ou répertoires indiqués par les `n-1` arguments).

4.3.5 `rm` : supprimer un fichier

La commande `rm` (abréviation du mot « `remove` ») permet de supprimer un ou des fichiers, mais aussi des répertoires qui ne sont pas vides (contenant un ou des fichiers et/ou un ou des sous-répertoires).

```
$ rm fichier1 fichier2
```

Cette commande supprimera les deux fichiers `fichier1` et `fichier2`.

Option	Description
`-f`	Supprime des éléments sans demander de confirmation
`-i`	Demande confirmation avant de supprimer quoi que ce soit
`-r`	Accède aux sous-répertoires et supprime les éléments qui s'y trouvent
`-v`	Affiche des informations au cours du processus de suppression

Ces options peuvent être utilisées séparément ou combinées.

La commande suivante permet de supprimer un répertoire et tous les fichiers ou sous-répertoires qu'il contient.

```
$ rm -r repertoire
```

La commande `rm -rf` est peut-être la plus dangereuse des commandes Linux, en particulier si vous êtes connecté en tant que « super utilisateur » (root). Les fichiers supprimés ne pourront pas être restaurés ! Quand on utilise les commandes, il n'y a pas de corbeille. Si vous tapez la commande `rm -rf *` à la racine et que vous êtes super-utilisateur, vous effacez toute l'arborescence (répertoires et fichiers), sans possibilité de retour en arrière.

4.4 Références à plusieurs fichiers - Caractères génériques

Beaucoup de commandes acceptent une liste de fichiers dont les noms sont séparés par des espaces et constituent autant d'arguments. Le Shell fournit une notation pour référencer plusieurs fichiers à la fois :

- le joker « `*` » représente une chaîne quelconque, éventuellement vide,
  `*.c` `doc*unix.html`
- le joker « `?` » dénote un caractère quelconque,
  `?nix` `*.s??`
- les caractères « `.` » (en début de nom de fichier) et « / » ne peuvent pas être reconnus par un joker.
- [...] représente un caractère appartenant à un ensemble
  `[Uu]nix` `*.sx[ci]`
- [^...] dénote un caractère n'appartenant pas à un ensemble
  `.[^co]` `[^[:upper:]]*`

Le Shell possède la caractéristique de terminer les noms de fichiers à votre place. Si vous écrivez un nom de fichier long sur la ligne de commandes, entrez les premières lettres, puis appuyez sur la touche Tab, le Shell complète le reste en fonction des fichiers se trouvant dans le chemin désigné (ou le répertoire courant si rien n'a été spécifié) ! C'est pareil pour les commandes.

N. Ferry, H. Renard, S. Lavirotte, D. Lopez
Polytech'Nice – Sophia

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

Exercices

Exercice n°1: Lister le contenu de votre répertoire personnel

Testez la commande `ls` en affichant, depuis votre répertoire personnel, la liste de tous vos fichiers et sous-répertoires :

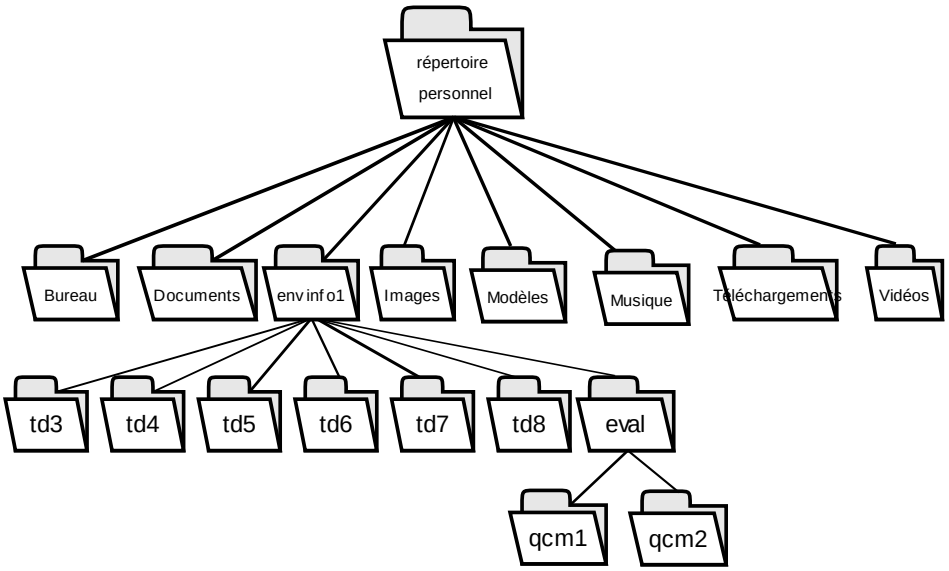
1. sous un format *condensé* (uniquement les noms)
2. sous un format long (affichant les permissions, le propriétaire, la taille, ...)
3. en affichant aussi les fichiers cachés (dont le nom commence par un point)
4. en colorant le nom des fichiers suivant leur type et en ordre inverse
5. avec un format long et en affichant les fichiers cachés, mais du plus récent au plus ancien (au lieu de l'ordre alphabétique qui est utilisé par défaut)
6. avec un format long et en affichant les fichiers cachés, mais du plus ancien au plus récent

Exercice n°2: Répertoires

Effectuer l'ensemble des actions suivantes et donnez pour chaque action la (les) commandes correspondante(s).

1. Dans votre répertoire personnel, créez un dossier `envinfo1`.
2. En restant dans votre répertoire personnel, créez les dossier `td3 td4 td5` dans le répertoire `envinfo1` à l'aide d'**une seule** commande.
3. Déplacez-vous dans le répertoire `envinfo1`.
4. Créez les répertoires `td6 td7 td8` à l'aide d'**une seule** commande.
5. Quelle(s) différence(s) faites-vous entre ces deux commandes pour créer les répertoires ?
6. En restant dans le répertoire `envinfo1`, créez en une seule commande un dossier `eval`, lui-même contenant `qcm1` et `qcm2`.
7. Si vous avez donné 3 arguments à la commande précédente, essayer de faire cette même action avec seulement 2 arguments (et en utilisant une option en plus des arguments).

Vous devez obtenir l'arborescence suivante :



Exercice n°3: Arborescence et chemins

À la suite de l'exercice précédent, vous vous trouvez toujours dans le répertoire `envinfo1`.

1. Déplacez-vous dans le répertoire `qcm1` en **une seule** commande.

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

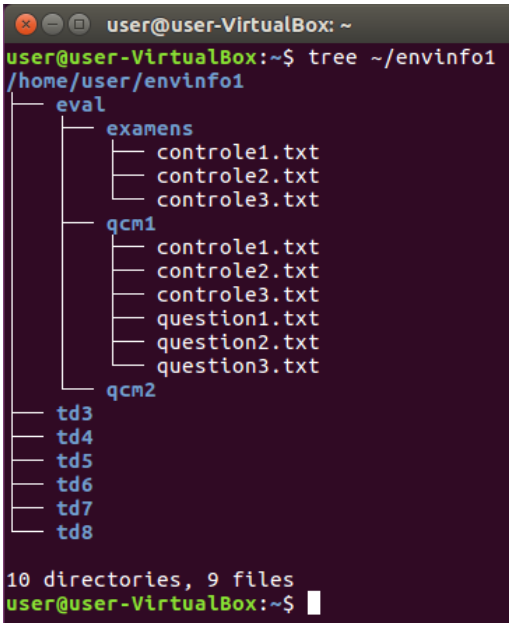
2. Utilisez la commande `cd` sans paramètre. Où vous retrouvez-vous ?
3. Déplacez-vous dans le répertoire `qcm1`. Avez-vous tapé la même commande qu'à première question de cet exercice ? Pourquoi ?
4. Allez dans le répertoire « grand-parent » de `qcm1` (donc dans le répertoire `envinfo1`). Donnez le chemin relatif par rapport au répertoire où vous vous trouvez (`qcm1`) et donnez le chemin absolu pour vous rendre dans ce répertoire.
5. Donnez la commande pour trouver le chemin absolu de votre répertoire personnel. Quel est ce chemin ?
6. Pour vous déplacer dans le répertoire `eval` donnez les chemins absolus (à partir de la racine puis à partir de votre répertoire personnel) et le chemin relatif.

Exercice n°4: Fichiers et Répertoires

À la suite de l'exercice précédent, vous vous trouvez toujours dans le répertoire `eval`.

1. Créez les fichiers vides `controle1.txt`, `controle2.txt` et `controle3.txt` en **une seule** commande.
2. Créez le répertoire `controles`
3. Renommez le répertoire `controles` en `examens`
4. Déplacer les fichiers `controle1.txt`, `controle2.txt` et `controle3.txt` dans le répertoire `examens`.
5. Déplacez-vous dans le répertoire `examens` puis copiez tous les fichiers dans le répertoire `qcm1`.
6. Déplacez-vous dans le répertoire `qcm1` et créez en **une seule** commande les fichiers `question1.txt`, `question2.txt` et `question3.txt`.

A ce stade, pour visualiser facilement l'arborescence et les fichiers quelle contient, vous pouvez utiliser la commande `tree`. Vous obtiendrez l'affichage suivant :



Exercice n°5: Suppression de répertoire et de fichiers

À la suite de l'exercice précédent, vous vous trouvez dans le répertoire `qcm1`.

1. Essayer d'effacer le répertoire examens à l'aide de la commande `rmdir`. Pourquoi cela ne fonctionne pas ?
2. Supprimer le répertoire `qcm2`.
3. Déplacer-vous dans le dossier `envinfo1`. En une seule commande. Dupliquez le répertoire `qcm1` et tout ce qu'il contient sous le nom `qcm2`.
4. Déplacez-vous dans le répertoire `qcm2`.

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

5. Supprimez tous les fichiers dont le nom commence par `controle` (utilisez pour cela un « joker »).
6. Créez le fichier caché `reponses.txt`.
7. Affichez l'ensemble des fichiers du répertoire `qcm2` avec les fichiers cachés et en format long.
8. Supprimez le dossier `qcm1` et tout ce qu'il contient en **une seule** commande.

Exercice n°6: Joker

A la suite de l'exercice précédent vous vous trouvez dans le répertoire `qcm2`.

1. Dans le répertoire `eval` créez les fichiers suivants : `qcm1.txt` `Qcm2.doc` `QCm3.txt` et `QCM4.doc`.
2. Affichez tous les fichiers de `eval` dont le nom commence par la lettre Q majuscule.
3. Affichez les noms de fichiers donc la troisième lettre est un M majuscule.
4. Affichez les noms de fichiers qui se terminent par l'extension `.txt`
5. Supprimez **tous les fichiers** que vous avez créés lors de la première question de cet exercice en essayant de spécifier le moins d'arguments possible.

Exercice n°7: Pour conclure, on utilise un peu tout

A la suite de l'exercice précédent vous vous trouvez dans le répertoire `eval`.

1. Déplacez-vous dans le répertoire `qcm2`.
2. Sans changer de répertoire, affichez uniquement les dossiers `td4`, `td5`, `td6`, `td7` et `td8`. Pour réaliser cette opération, utilisez une seule commande et un seul argument
3. Supprimez les dossiers `td4`, `td5`, `td6`, `td7` et `td8`. Pour réaliser cette suppression, utilisez la commande précédente sans la retaper.
4. A l'aide de la commande ls, affichez l'ensemble des répertoires, des sous-répertoires et des fichiers du répertoire `envinfo1` (sans changer de répertoire courant). Cet affichage devra être réalisé en format long et avec les fichiers cachés. Vous utiliserez un chemin absolu et un chemin relatif. Utilisez aussi la commande `tree` pour visualiser l'arborescence.
5. Allez dans la corbeille. Pouvez-vous récupérer les répertoire et fichiers que vous avez supprimés avec les commandes précédentes.
6. Allez dans votre dossier personnel, supprimer le répertoire envinfo1

N. Ferry, H. Renard, S. Lavirotte, D. Lopez
Polytech'Nice – Sophia
PeiP 1
2019-2020

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

Exercices complémentaires

Exercice A :

1. Créez un répertoire `system` dans votre répertoire personnel **puis** un répertoire `tp1` dans `system`
2. Effacez le répertoire `system` avec la commande `rmdir`. Que constatez-vous ?
3. Après avoir effacé les répertoires `tp1` et `system`, créez à l'aide d'une seule commande les répertoires `system`, `system/tp1`, `system/tp2`
4. Renommez le répertoire `system` en `test`
5. Copiez un fichier de votre choix du répertoire `/bin` dans le répertoire `test/tp1` de votre dossier personnel :
    1. En faisant la copie depuis le répertoire `/bin`
    2. En faisant la copie depuis le répertoire `test/tp1`
    3. En faisant la copie depuis votre *homedir*, en utilisant des chemins absolus
    4. En faisant la copie depuis votre *homedir*, en utilisant des chemins relatifs
6. Effacez à l'aide d'une seule commande les répertoires `test/tp1` et `test/tp2`

Exercice B :

1. Combien y a-t-il de noms de répertoires dans la racine ?
2. Donnez un exemple de nom de fichier se trouvant dans votre répertoire personnel :
    - par un **chemin relatif**;
    - par un **chemin absolu**.

Exercice C :

Déterminez les commandes permettant de réaliser les actions suivantes :

- Déterminer le répertoire par défaut dans la hiérarchie des répertoires ?
- Y a t-il des fichiers, des répertoires dans ce répertoire ?
- Entrer du texte dans un fichier nommé « `Mon_fichier` » que vous avez créé au préalable.
- Lister le contenu de « `Mon_fichier` ».
- Lister le répertoire courant.
- Lister les répertoires `/bin` et `/dev`.
- Créer sous votre répertoire deux sous-répertoires : « `Source` » et « `Data` ».
- Se positionner sous « `Source` ».
- Listez le répertoire courant.
- Revenir sous le répertoire de départ et détruire « `Source` ».
- Créer un deuxième fichier nommé « `Mon_fichier_2` ».
- Copier chaque fichier en `nom_de_fichier.old`.
- Créer un répertoire « `Old` ».
- Déplacer les fichiers avec l'extension `.old` vers le répertoire « `Old` ».
- Effacer tous les fichiers créés dans `Old` sans effacer le répertoire `Old`.

Exercice D

Exploration de l'arborescence Linux.

`ls, cp, mv, rm, cd, pwd, mkdir, rmdir`

- Indiquez par une commande dans quel répertoire vous vous trouvez.
- Allez dans le répertoire `/usr/share/doc`, puis vérifiez le chemin de votre répertoire courant.
- Remonter dans le répertoire parent puis vérifier.
- Allez dans votre répertoire personnel sans taper son chemin.

TD séances n° 3 et n° 4
Répertoires et Fichiers sous Unix

- Retournez dans votre répertoire précédent sans taper son chemin.
- Retourner dans votre répertoire personnel et listez les fichiers présents.
- Listez maintenant tous les fichiers (même ceux cachés).
- Affichez de façon détaillée le contenu du répertoire `/usr` sans changer le répertoire de travail.
- Affichez l'arborescence de fichiers contenue dans `/var` sans changer le répertoire de travail.
- Affichez de façon détaillée le contenu du répertoire `/var/log` en classant les fichiers du plus vieux au plus récent.

Exercice E

Répertoires et consultation de fichiers.

- Allez dans votre répertoire personnel.
- Créez un répertoire portant le nom de `CommandesLinux`.
- Allez dans votre répertoire `CommandesLinux`.
- Créez l'arborescence `cours1/cours2/cours3/cours4`.
- Listez le contenu du répertoire courant de façon récursive.
- Supprimez le répertoire `cours1`. Est-ce possible ?
- Supprimez l'arborescence de répertoire `cours1/cours2/cours3/cours4`, puis vérifiez en listant le répertoire de façon récursive.
- Allez dans le dossier `CommandesLinux` et créez les répertoires suivants :

```
Code :
.
 |--couleur
 |   `--froide
 `--forme
    |--angle
    `--courbe
```

- Copiez le fichier `/etc/services` dans votre répertoire `CommandesLinux`.
- À qui appartient le fichier que vous venez de copier ? Quelle est sa date de sa dernière modification ?
- Créez les fichiers ne contenant aucune donnée et dont les noms sont les suivants : `rond.txt, triangle.txt, carre.txt, rectangle.txt, vert.txt et bleu.txt`
- Déplacez le fichier `rond.txt` dans le répertoire `courbe` et les fichiers `triangle.txt, carre.txt, rectangle.txt` dans le répertoire `angle`.
- Déplacez les fichiers `vert.txt` et `bleu.txt` dans le répertoire `froide`.
- Allez dans le répertoire `couleur` et afficher le contenu du répertoire de façon récursive.
- Copier le répertoire sous le nom `chaude`. Est-ce possible ? Comment ?
- Allez dans le répertoire `chaude` et renommez le fichier `bleu.txt` en `rouge.txt` et `vert.txt` en `jaune.txt`.
- Remontez dans le répertoire `CommandesLinux` et renommez le répertoire `couleur` en `peinture`. Est-il besoin de spécifier une option particulière à la commande `mv`.
- Listez la totalité de l'arborescence contenue dans le répertoire `CommandeLinux`.
- Affichez le contenu du fichier `/etc/issue`. Que contient-il ?