Premiers pas avec le terminal

Formation UNIX

Philippe Veber

11 septembre 2017

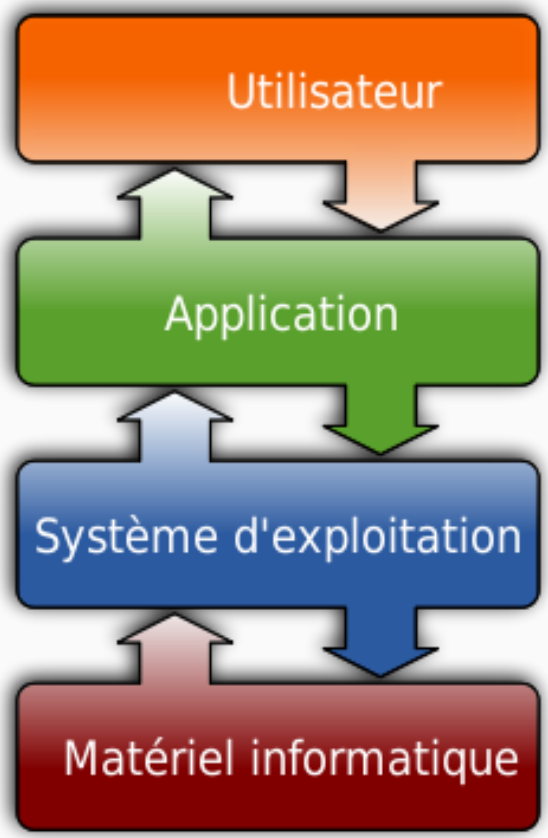
PRABI/LBMC

Premières notions

Système d'exploitation

Deux types de programmes :

- applications (programmes utilisateurs) = services directement utiles (p. ex. traitement de texte, calcul scientifique)
- programmes système, utilisés pour interagir avec le matériel, p. ex. :
  - lancer un programme
  - écrire sur un disque dur
  - afficher un texte à l'écran



Wikipedia

Système de fichiers

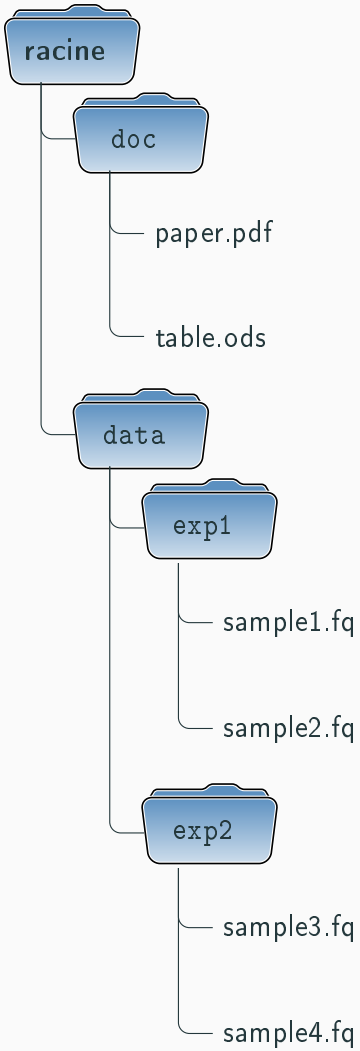
En première approximation

➔ façon d'organiser le stockage des données

Ingrédients typiques :

- le fichier (= document)
- le répertoire (« tiroir » pouvant contenir des fichiers et des répertoires)

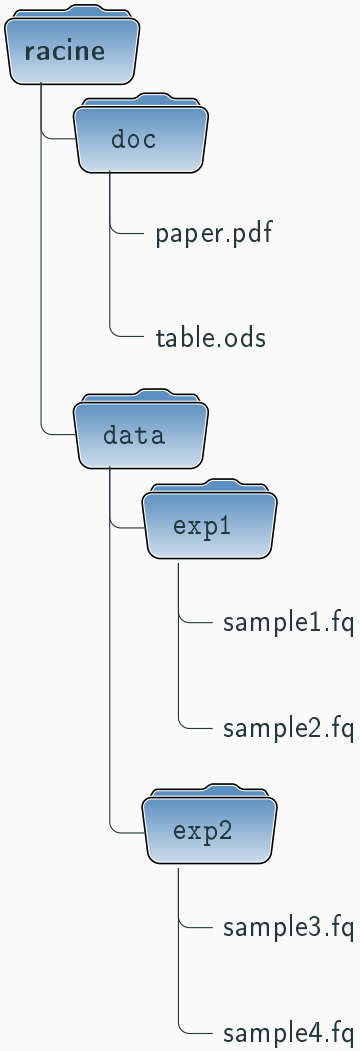
Arborescence et chemin



- répertoires et fichiers forment un arbre (à l'envers)
- le répertoire contenant l'ensemble des autres répertoires et fichiers est appelé **racine**
- chaque fichier ou répertoire peut être identifié par la suite de répertoires menant de la racine à lui :

  `racine` → `data` → `exp1` → `sample2.fq`

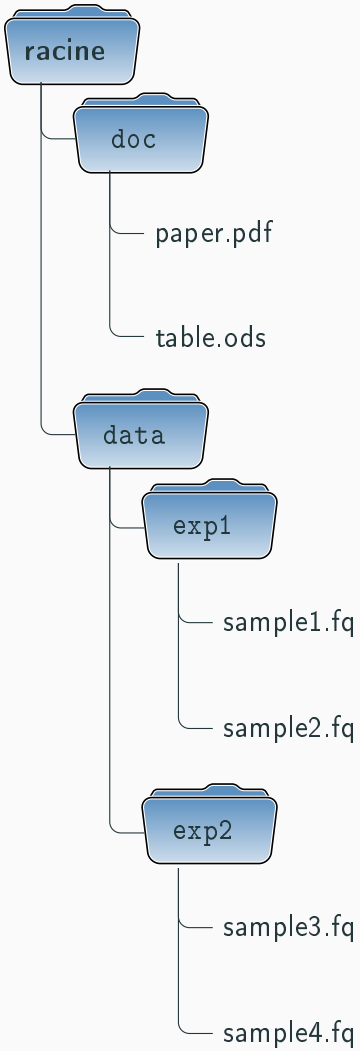
Arborescence et chemin



- répertoires et fichiers forment un arbre (à l'envers)
- le répertoire contenant l'ensemble des autres répertoires et fichiers est appelé **racine**
- chaque fichier ou répertoire peut être identifié par la suite de répertoires menant de la racine à lui :

`racine/data/exp1/sample2.fq`

Arborescence et chemin



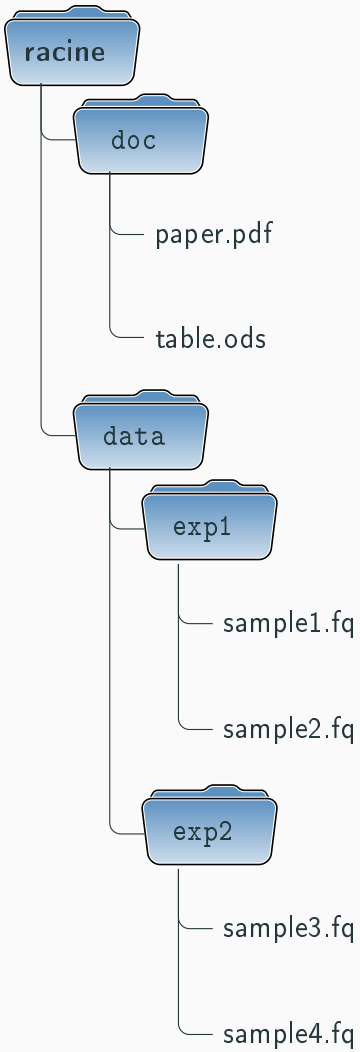
- répertoires et fichiers forment un arbre (à l'envers)
- le répertoire contenant l'ensemble des autres répertoires et fichiers est appelé **racine**
- chaque fichier ou répertoire peut être identifié par la suite de répertoires menant de la racine à lui :

`/data/exp1/sample2.fq`

Chemins absolus et relatifs (1/2)

- un chemin à partir de la racine est dit « absolu »
- il commence donc nécessairement par /
- on peut désigner un objet à partir d'une autre position (dite position courante) en utilisant deux symboles :
  - . désigne le répertoire courant
  - .. désigne le répertoire contenant le répertoire courant

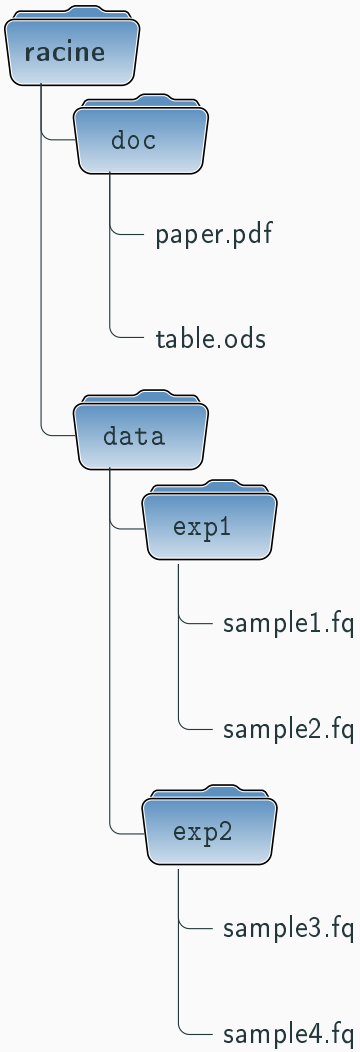
Chemins absolus et relatifs (2/2)



- posons que le répertoire courant est `/data/exp1`
- le chemin *relatif* vers `/data/exp2/sample3.fq` est :

..

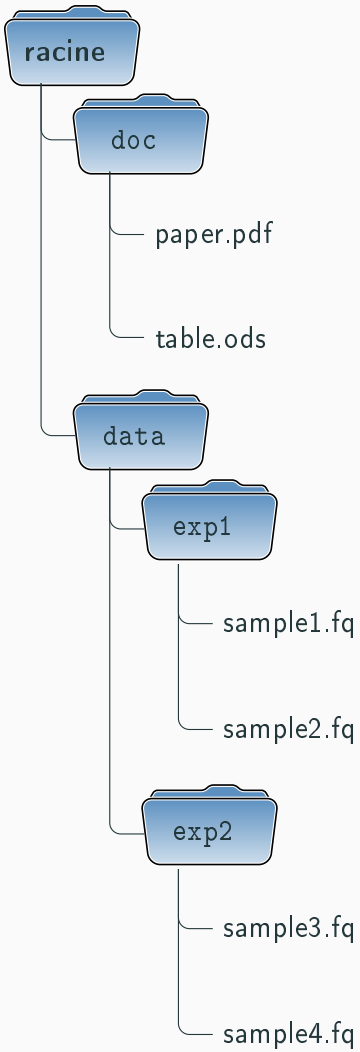
Chemins absolus et relatifs (2/2)



- posons que le répertoire courant est `/data/exp1`
- le chemin *relatif* vers `/data/exp2/sample3.fq` est :

  `../exp2`

Chemins absolus et relatifs (2/2)



- posons que le répertoire courant est `/data/exp1`
- le chemin *relatif* vers `/data/exp2/sample3.fq` est :

  `../exp2/sample3.fq`

Le terminal

- programme permettant d'interagir avec le système
- il s'agit d'un **interpréteur**, *i.e.* un programme répétant la boucle :
  1. lecture d'une commande
  2. exécution de la commande
  3. affichage des sorties de la commande

- on tape la commande après l'invite
- puis ENTRÉE pour démarrer l'exécution

Premières commandes

pwd *(Print Working Directory)*
Retourne la « position » courante du terminal dans l'arborescence

ls *(List Segments)*
Affiche la liste des fichiers et répertoires présents dans le répertoire courant

Arguments

Le comportement de certaines commandes peut être modifié en ajoutant des paramètres (ou arguments) à la commande.

ls ./foo

Affiche le contenu du répertoire `./foo`

ls -l

Affiche la liste détaillée des fichiers présents dans le répertoire courant

ls -l ./foo

Affiche la liste détaillée du contenu du répertoire `./foo`

Exercice

- ouvrir un terminal
- déterminer sa position courante
- trouver le fichier le plus récent du répertoire
- trouver le fichier de plus grande taille du répertoire

Se déplacer dans l'arborescence

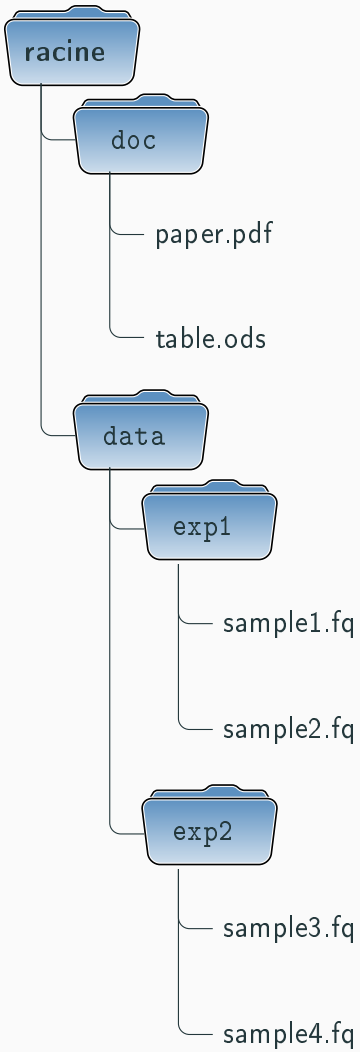
cd *Change Directory*

```
cd CHEMIN
```

Modifie le chemin courant du terminal

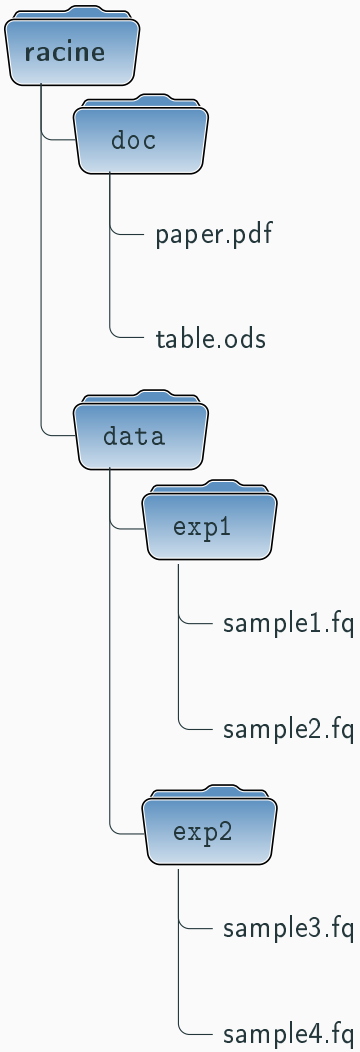
- déplacement absolu : `cd /data/exp2`
- déplacement relatif : `cd ../../doc`
- que fait `cd .` ?

Exercice




```
$ pwd
/data/exp1
$ cd ../../doc
$ pwd
# ?
```

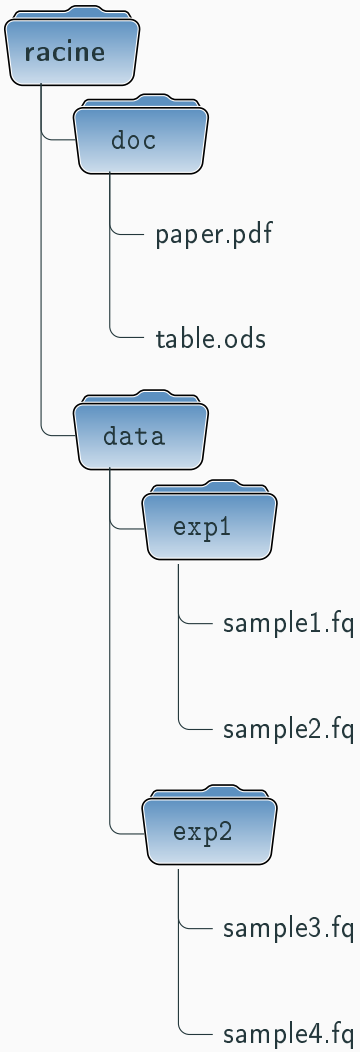
Exercice




```
$ pwd
/data/exp1
$ cd ../../doc
$ pwd
# ?
/doc
```

```
$ cd ../../data/./exp1/./..
$ pwd
# ?
```

Exercice




```
$ pwd
/data/exp1
$ cd ../../doc
$ pwd
# ?
/doc
```

```
$ cd ../../data/./exp1/./..
$ pwd
# ?
/data
```

Quelques outils pour inspecter fichiers et répertoires

Nouvelles options pour `ls` (1/2)

ls

`ls [OPTIONS] CHEMIN*`

Affiche le nom des fichiers ou le contenu des répertoires donnés en arguments

-l	affiche les détails du fichier
-a	affiche les fichiers débutant par . (fichiers cachés)
-h	affiche des tailles de fichiers lisibles par un humain
-R	lister récursivement les sous-répertoires
-t	trier du plus récent au plus ancien
-r	inverser le tri

Nouvelles options pour `ls` (2/2)

Note 1 : les options de la forme `-x` sont appelés interrupteurs (*switch*), ils permettent d'activer un comportement.

Note 2 : les interrupteurs peuvent être combinés :

`ls -l -a -t -h -r` ≡ `ls -lathr`

Exercice

Tester chaque interrupteur de la commande `ls` et quelques combinaisons

Visualiser une arborescence (en mode texte !)

tree

`tree CHEMIN?`

Affiche un rendu texte d'une arborescence

```
$ tree formation-unix
formation-unix/supports
|-- formation-unix.cls
|-- img
|   |-- files.png
|   |-- IBM_card_storage_NARA.jpg
|   |-- logo-lbmc.png
|   |-- logo-prabi.png
|   |-- openssh.png
|   `-- os-stacks.png
|-- Makefile
[...]
|-- presentation1A_accueil.pdf
`-- tp1B_systeme_de_fichiers.pdf
```

Format d'un fichier

- on appelle extension d'un fichier le morceau de son nom à droite du dernier point
- l'extension indique conventionnellement le format utilisé pour encoder le contenu
- les fichiers n'ont pas nécessairement d'extension et rien ne dit que l'extension est bien cohérente avec le contenu réel du fichier
- pour obtenir de l'information sur le type de contenu, on peut utiliser `file`

file

`file CHEMIN`

« devine » le type de contenu et d'encodage d'un fichier.

Inspecter le contenu d'un fichier (1/2)

cat

```
cat CHEMIN
```

Affiche le contenu d'un fichier à l'écran

Attention à ne pas utiliser cat sur de gros fichiers.

Inspecter le contenu d'un fichier (2/2)

less

`less CHEMIN`

Affiche de manière interactive le contenu d'un fichier à l'écran

Commandes :

q	quitter `less`
↑	se déplacer d'une ligne vers le haut
↓	se déplacer d'une ligne vers le bas
espace	se déplacer d'une page vers le bas
g	aller au début du fichier
G	aller à la fin du fichier
/	menu interactif de recherche
n	occurrence suivante du motif recherché
N	occurrence précédente

Afficher le début d'un fichier

head

`head [OPTIONS] CHEMIN`

Affiche les premières lignes d'un fichier

-n ENTIER nombre de lignes

Note : on voit que les options d'une commande (ici `-n`) peuvent avoir une valeur associée (ici un entier)

Afficher la fin d'un fichier

tail

`tail [OPTIONS] CHEMIN`

Affiche les dernières lignes d'un fichier

-n ENTIER nombre de lignes

Comparer deux fichiers

diff

```
diff -u FICHIER1 FICHIER2
```

Affiche un alignement de deux fichiers pour repérer les différences

```
$ diff -u v1.txt v2.txt
--- v1.txt 2017-09-10 12:41:39.250264747 +0200
+++ v2.txt 2017-09-10 12:41:43.526093497 +0200
@@ -1,4 +1,5 @@
 1 l de lait
-120 g de farine
+100 g de farine
 200 g de sucre
 4  oeufs
+1 gousse de vanille
```

Exercice

- faites une copie du fichier `~/.bash_history`
- à l'aide de `cd`, `ls` et `file`, trouvez un maximum de formats de fichiers présents sur votre machine
- ouvrez le fichier `~/.bash_history` avec `less` et essayez chacune des commandes présentées
- essayez les commandes `head` et `tail` sur le fichier `~/.bash_history`
- à l'aide de `diff`, comparer votre copie de `~/.bash_history` et sa version actuelle

Quelques outils pour manipuler fichiers et répertoires

Créer

touch — créer un fichier vide

```
touch CHEMIN
```

mkdir — créer un répertoire vide

```
mkdir [OPTIONS] CHEMIN
```

`-p` crée répertoires parents si nécessaire

Utilisation de `mkdir`

```
$ tree --charset=ascii
.
$ mkdir foo/bar
mkdir: impossible de créer le répertoire « foo/bar »: Aucun fichier ou dossier de ce type
$ mkdir -p foo/bar
$ tree --charset=ascii
.
`-- foo
    `-- bar

2 directories, 0 files
$ mkdir foo/bar
mkdir: impossible de créer le répertoire « foo/bar »: Le fichier existe
$ mkdir -p foo/bar
```

Détruire

Sous UNIX la suppression est « définitive »

rmdir **supprime un répertoire vide**

```
rmdir CHEMIN+
```

rm **supprime un fichier**

```
rm [OPTIONS] CHEMIN+
```

-r efface récursivement les fichiers d'un répertoire

Exemple d'utilisation de `rm`

```
$ mkdir -p foo/bar
$ mkdir -p foo/baz
$ tree --charset=ascii
.
`-- foo
    |-- bar
    `-- baz

3 directories, 0 files
$ rmdir foo/bar
$ rm foo/baz/
rm: impossible de supprimer 'foo/baz/': est un dossier
$ rm foo
rm: impossible de supprimer 'foo': est un dossier
$ rm -rf foo
$ tree --charset=ascii
.

0 directories, 0 files
```

Copier

cp **copie de fichiers/répertoires**

```
cp [OPTIONS] SOURCE DESTINATION
```

-r déplace aussi les répertoires

-i demande confirmation avant d'écraser

Déplacer/renommer

Sous UNIX, il n'y a pas de différence entre déplacer et renommer un fichier.

mv **renommer un fichier ou un répertoire**

```
mv [OPTIONS] SOURCE DESTINATION
```

-i demande confirmation avant d'écraser

Note :

- si la destination existe déjà et est un répertoire, la source y est déplacée.

Compléments

Ergonomie

Épargnez vos doigts !

- rappel des commandes (↑ et ↓)
  - notion d'historique des commandes
- complétion des commandes et de leurs arguments (TAB)
- raccourcis clavier

Ctrl-A	aller en début de ligne
Ctrl-E	aller en fin de ligne
Ctrl-K	couper la commande à droite du curseur
Ctrl-Y	coller
Ctrl-_	défaire

- recherche dans l'historique (Ctrl-R)

Joker

Le symbole * est inerprété par le terminal pour désigner un ensemble de chemins.

Exemple :

`ls doc.pdf` seulement `doc.pdf`

Joker

Le symbole `*` est inerprété par le terminal pour désigner un ensemble de chemins.

Exemple :

`ls doc.pdf`	seulement `doc.pdf`
`ls *`	tous les fichiers du répertoire courant

Joker

Le symbole `*` est inerprété par le terminal pour désigner un ensemble de chemins.

Exemple :

`ls doc.pdf`	seulement `doc.pdf`
`ls *`	tous les fichiers du répertoire courant
`ls *.pdf`	ceux dont le nom termine par `.pdf`

Joker

Le symbole * est inerprété par le terminal pour désigner un ensemble de chemins.

Exemple :

`ls doc.pdf`	seulement `doc.pdf`
`ls *`	tous les fichiers du répertoire courant
`ls *.pdf`	ceux dont le nom termine par `.pdf`
`ls d*.pdf`	idem + le nom commence par `d`

Cas typique d'usage : déplacer un ensemble de fichiers

```
mv *.pdf destination/plus/appropriée
```