Initiation à l'environnment Unix
CM3 : fichiers et utilisateurs

Pierre Rousselin

Université Paris 13
L1 informatique
octobre 2021

Utilisateurs et groupes

Fichiers sous Unix, introduction

Utilisateur normal et super-utilisateur

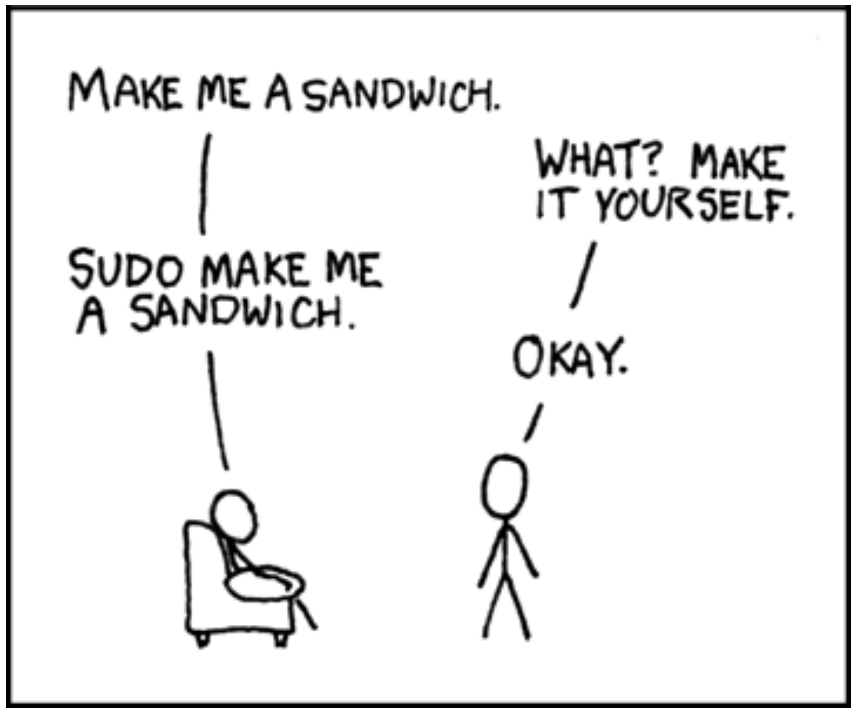
Il y a deux types d'utilisateurs :

- **Un** super-utilisateur, souvent appelé `root` qui peut tout faire sur le système : modifier, supprimer, lire tout ce qu'il veut, lancer ou tuer n'importe quel processus, ...
- Les autres utilisateurs, qui peuvent faire ce que leur permettent leurs permissions sur chaque fichier du système : souvent ils peuvent faire à peu près ce qu'ils veulent dans leur répertoire personnel et pas grand chose en-dehors.
- Certains programmes nécessitent d'être lancés par certains utilisateurs : en particulier, ceux qui modifient le système doivent être lancés par root. Par exemple :
  - ajout d'un utilisateur (`useradd`)
  - installation d'un programme (`make install`) ou utilisation d'un gestionnaire de paquets (`apt` pour Debian, `dnf` pour Fedora, *etc*).

Devenir un autre utilisateur

- Commande `su <user>` pour `switch user`, par exemple `su alice` si `alice` fait partie des utilisateurs. Bien sûr, on vous demande son mot de passe.
- Pour devenir `root` : `su root` ou `su` (sans argument) : parfois impossible, même sur votre propre machine, car le compte `root` n'a pas de mot de passe dans certaines distributions.

La commande `sudo`



Pour ceux qui peuvent être `root`, sur leur machine, ou qui en administrent d'autres, la commande `sudo` (pour *superuser do*) permet de faire quelque chose *en tant que super-utilisateur*, si vous faîtes partie des utilisateurs qui peuvent le faire (*sudoers*, fichier `/etc/sudoers`).

La commande sudo

```
$ ls /etc/nftables
ls: impossible d'ouvrir le répertoire '/etc/nftables/':
    Permission non accordée
$ sudo ls /etc/nftables
[sudo] Mot de passe de pierre :
all-in-one.nft...
$ useradd alice # créer un nouvel utilisateur appelé alice
$ sudo useradd alice
$ sudo userdel alice # supprimer le compte d'alice
$ sudo su # devenir root
# id -un # mon identifiant
root
# echo $HOME
# /root
```

La commande `sudo`

- Évidemment, vous ne pouvez pas tester `sudo` en salle de TP (d'ailleurs moi non plus et c'est très bien comme ça !)
- En pratique, sur votre machine, pour mettre à jour ou installer des paquets. Par exemple sur Debian et dérivés (Ubuntu, Mint, ...)

```
$ sudo apt update # mettre à jour la base de donnée
$ sudo apt upgrade # mettre à jour les paquets
$ sudo apt install coqide # installer le paquet coqide
```

Être ou ne pas être `root`

- « De grands pouvoirs entraînent de grandes responsabilités. »
- `root` c'est que dans les grandes occasions !
- Métaphore : quand vous êtes `root`, c'est comme quand vous portez une perceuse branchée, vous risquez de faire des bêtises !
- `Permission denied :` la tentation est grande d'ajouter `sudo` en début de commande, mais on a vu (en vrai) des gens effacer leur table de partition avec ça.
- Si un utilisateur normal n'a pas le droit de le faire, il y a sans doute une raison !
- Les réseaux importants sont gérés par des *administrateurs systèmes* qui sont les seuls à avoir accès au compte `root` : c'est un métier assez recherché qui demande des compétences techniques et une grande droiture.

Utilisateurs : être

- Chaque utilisateur a un nom (`id -nu`) et un identifiant numérique : l'`uid` (pour *user identity*, `id -u`).
- La liste de tous les utilisateurs est accessible dans le fichier `/etc/passwd`.
- Il y a des utilisateurs qui ne correspondent à aucun être humain : pour des raisons de sécurité, certains programmes sont lancés en faisant « comme si » ils étaient lancés par un certain utilisateur (évite qu'ils soient lancés par `root` et aient tous les drois).

Utilisateurs : avoir

Un utilisateur a :

- un `SHELL` par défaut, souvent `bash` sous Linux ou `zsh` sous MacOS mais peut être configuré (commande `chsh`) ;
- un `HOME`, répertoire personnel, aussi noté `~` dans le shell ;
- ses propres fichiers de configuration (`.bashrc`, `.zshrc`, `.vimrc`, ...) souvent dans son répertoire personnel ;
- un mot de passe, des clés publiques et privées (répertoire `.ssh`), etc.

Utilisateurs et groupes

En outre, chaque utilisateur a

- un groupe courant (`id -g` pour le numéro (*gid*) et `id -ng` pour le nom) ;
- éventuellement d'autres groupes auquel il appartient (`id -G` pour les numéros et `id -Gn` pour les noms).

L'idée de groupe d'utilisateurs illustre parfaitement quelques idées importantes du système Unix :

- partage et importance de travailler ensemble ;
- simplicité : il suffit de régler certaines permissions pour que les utilisateurs du même groupe puissent accéder aux mêmes ressources.

Utilisateurs et groupes

- C'est `root` qui a la charge de donner au moins un groupe à chaque utilisateur, pour les enthousiastes :
  - `groupadd` et `groupdel` pour créer et supprimer un groupe
  - `usermod` pour modifier les attributs d'un utilisateur, en particulier lui ajouter ou supprimer des groupes.
- La commande `newgrp <groupe>` ouvre un nouveau shell avec `<groupe>` comme groupe courant.

```
$ id -Gn
pierre wheel dialout
$ id -gn
pierre
$ newgrp wheel
$ id -Gn
wheel dialout pierre
$ id -gn
wheel
```

Fichiers sous Unix, introduction

Tout est fichier

> *Everything in the* Unix *system is a file. This is less an oversimplification than you might think. It is one of the best examples of the "keep it simple" philosophy, showing the power achieved by careful implementation of a few well-chosen ideas.*

Kernighan et Pike, *The Unix Programming Environment (1984)*.

Tout est fichier

En effet, tout ce dans quoi on peut *lire* (appel système `read`) et/ou *écrire* (appel système `write`) des données est un fichier : ceci inclut

- les fichiers contenus sur un périphérique de stockage (disque dur...) qui peuvent contenir des données (dans tel ou tel format) ou des programmes qui seront exécutés par le noyau ;
- les terminaux, physiques (`/dev/tty`) ou virtuels (`/dev/pts`) ;
- les périphériques d'entrée (clavier, souris, ...)
- les périphériques de sortie (écran, enceintes, ...)
- les périphériques de stockage (disques durs, ...)

Types de fichiers

Il y a 7 types de fichiers dans l'arborescence, et c'est tout !

- `-` pour les fichiers normaux (*regular files*) ;
- `d` pour *directory*, répertoire ;
- `l` pour *link*, les liens symboliques (des « raccourcis » sous windows) ;
- `b` pour *block device* et `c` pour *character device* : périphériques de type bloc ou caractère (pas ou peu abordés dans ce cours) ;
- `p` pour *pipe* ou tube nommé (assez rare) ;
- `s` pour *socket* (en français « prise »comme une prise de courant) permettant les échanges de données entre processus sur le même système ou via le réseau (internet par exemple).

Premier caractère de la sortie de `ls -l`.

Types de fichiers

```
$ ls -ld /etc/passwd /run/ /bin /dev/sda /dev/input/mouse0
lrwxrwxrwx.  1 root root       7 26 janv.  2021 /bin -> usr/bin
crw-rw----.  1 root input 13, 32  5 oct.  07:28 /dev/input/mouse
brw-rw----.  1 root disk   8,  0  5 oct.  07:28 /dev/sda
-rw-r--r--.  1 root root    2690  5 oct.  10:58 /etc/passwd
drwxr-xr-x. 51 root root    1420  5 oct.  07:49 /run/
$ ls -l /run/cups/cups.sock
srw-rw-rw-. 1 root root 0  5 oct.  07:28 /run/cups/cups.sock
$ ls -l /run/initctl
prw-------. 1 root root 0  5 oct.  07:28 /run/initctl
```

Fichiers réguliers et données

- Uniformité des fichiers réguliers : le système ne traite pas différemment un fichier source C ou un fichier pdf ou un tableau libreoffice.
- Tous sont des fichiers qui ne sont que des suites d'octets : ce sont les processus qui sont chargés de les interpréter et leur donner un sens.
- Il n'y a donc pas différents types de fichiers.
- Conventions pour le nommage : une extension, pour aider à s'y retrouver, par exemple `.c` pour du code en C, `.pdf` pour un document pdf (*portable document file*), etc.
- Mais ce n'est qu'une convention entre humains. Le système s'en fiche (toutefois certains utilitaires s'en servent pour « deviner » à quoi est destiné ce fichier).

La commande `file`

La commande `file` essaie de deviner à quoi sert un fichier en se servant de son contenu.

```
$ file unix_cm3.tex
unix_cm3.tex: LaTeX document, UTF-8 Unicode text
$ file sandwich.png
sandwich.png: PNG image data, 360 x 299, 8-bit grayscale, non-in
$ file unix_cm3.pdf
unix_cm3.pdf: PDF document, version 1.5
```

La commande `file`

La commande `file` essaie de deviner à quoi sert un fichier en se servant de son contenu.

```
$ file unix_cm3.tex
unix_cm3.tex: LaTeX document, UTF-8 Unicode text
$ file sandwich.png
sandwich.png: PNG image data, 360 x 299, 8-bit grayscale, non-in
$ file unix_cm3.pdf
unix_cm3.pdf: PDF document, version 1.5
```

Mais peut toujours se tromper.

```
$ cat a.txt
Pour coder en C, on doit souvent taper
#include <stdio.h>
$ file a.txt
a.txt: C source, ASCII text
```

Contenu d'un fichier

Il n'y a jamais rien d'autre dans un fichier que ce qu'on y écrit.
La commande **od** (comme **octal dump**, déverser en octal) permet de voir les données contenues dans le fichier, sous différentes formes, ici octal et hexadécimal.

```
$ cat >comptine.txt
Mon petit lapin
a bien du chagrin.
$ ls -l comptine.txt
-rw-r--r--. 1 pierre pierre 35  5 oct.  13:34 comptine.txt
$ od -cx comptine.txt
0000000   M   o   n       p   e   t   i   t       l   a   p   i
           6f4d    206e    6570    6974    2074    616c    6970
0000020   a       b   i   e   n       d   u       c   h   a   g
           2061    6962    6e65    6420    2075    6863    6761
0000040   n   .  \n
           2e6e    000a
0000043
```