# Chapitre 3: Fichiers

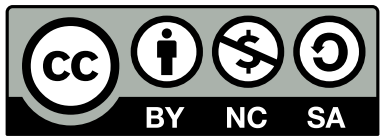
## INF1070

## Utilisation et administration des systèmes informatiques

Jean Privat & Alexandre Blondin Massé

Université du Québec à Montréal

v213



# Plan

# Fichier = information numérique = donnée

## Manipulation des fichiers via des applications

- Ouverture via l'application (menu Fichier > Ouvrir)
- Exécution d'une commande avec le fichier en argument

```
$ vim readme.txt
$ display debian.png
$ evince man.pdf
```

## Application par défaut (par type de fichier)

- Double-clic sur l'icône du fichier lance l'application par défaut
- `xdg-open` ouvre un fichier dans l'application par défaut

```
$ xdg-open debian.png
```

# Problématiques liées aux fichiers

Les données, sont la partie **importante** d'un système informatique

Les fichiers, sont donc liés à de nombreuses problématiques :

- Contenu: type des fichiers
- Organisation: système de fichiers et répertoires
- Manipulation: lecture, écriture, copie, suppression, etc.
- Protection: droits d'accès et utilisateurs
- Fiabilité: sauvegardes

# Répertoires et chemins

# Lister les fichiers

Commande `ls` (*list*)

- Affiche le contenu des répertoires passés en argument
- Ou le répertoire courant par défaut

```
$ ls
hello.txt  lisez-moi.txt  répertoire
```

- `-R` (`--recursive`): parcours les sous-répertoires
- `-d` (`--directory`): affiche les répertoires (et non leur contenu)

`tree` (extra) — affiche l'arborescence

```
$ tree
+-- hello.txt
+-- lisez-moi.txt
\-- répertoire
    \-- document.txt
1 directory, 3 files
```

# Chemins

- Le séparateur des répertoire est « `/` » (*slash*)
- Un nom de fichier peut contenir tous les caractères (sauf `/` et le caractère nul)
- La racine du système de fichier s'appelle `/` (*root*)
- Attention: `/` a donc *deux* rôles

## Chemin absolu

- Un nom de fichier qui part de la racine
- Commence par un `/`
- Exemple: `/home/privat/ens/INF1070/slides/data`

## Chemin relatif

- Un nom de fichier qui part du répertoire courant
- Ne commence pas par un `/`
- Exemple: `slides/data`

# Hiérarchie standard des fichiers

Filesystem Hierarchy Standard (inclus dans la LSB):

- `/bin` commandes minimalement nécessaires
- `/boot` chargeur d'amorçage et noyaux
- `/dev` fichiers de périphériques
- `/etc` fichiers de configuration
- `/home` répertoires des utilisateurs
- `/lib` bibliothèques pour `/bin` et `/sbin`
- `/media` points de montages pour médias amovibles
- `/mnt` points de montages temporaires
- `/opt` logiciels optionnels
- `/proc` interface avec le noyau et les processus (pas FHS)
- `/root` répertoire du super-utilisateur
- `/run` données d'exécution
- `/sbin` commandes minimales pour administrateur

# Hiérarchie standard des fichiers (2)

- `/srv` données des services
- `/sys` autre interface avec le noyau (pas FHS)
- `/tmp` fichiers temporaires
- `/usr` commandes non minimalement nécessaires
- `/usr/bin`, `/usr/lib`, `/usr/sbin` comme pour `/`
- `/usr/include` entête des bibliothèques (pour dev)
- `/usr/share` données indépendantes de l'architecture
- `/usr/local` hiérarchie tertiaire locale
- `/var` fichiers variables divers des services
- `/var/log` fichiers de journalisation

Note: il y a des variations entre les systèmes,
voire pas du tout respecté, ex. macOS

# Fichiers cachés

Un fichier dont le nom commence par . (point)

- N'est pas affiché par `ls` par défaut
- Le . initial n'est pas substitué par un glob (`*`, `?` ou `[]`)
- Ce **n'est pas** un mécanisme de protection

```
$ ls
hello.txt  lisez-moi.txt  répertoire
$ echo *
hello.txt  lisez-moi.txt  répertoire
```

`-a` (`--all`) et `-A` (`--almost-all`) de `ls` affichent les fichiers cachés

```
$ ls -a
.  ..  hello.txt  lisez-moi.txt  répertoire  .secret
$ echo .*
.  ..  .secret
```

# Pourquoi cacher des fichiers ?

Seulement pour alléger la sortie de `ls`

- Fichiers spéciaux . et .. (origine historique)
- Fichiers de configurations (historique aussi)
- Fichiers temporaires et cache

# Répertoires . et ..

- . désigne le répertoire courant
- .. désigne le répertoire parent

Il y a toujours . et .. dans chaque répertoire

- Un répertoire est **vide** s'il contient seulement . et ..

# Question sur les chemins

- Comment afficher le contenu du fichier « - » ?
- Qu'affiche « `cat /home/./privat/../../etc/./passwd` » ?
- Que fait la commande « `cd ../mauvais_nom/..` » ?
- Qu'affiche « `echo ./../.` » ?

# Basename et dirname

`basename` enlève le chemin et le suffixe d'un nom de fichier

`dirname` enlève le dernier composant d'un chemin

- Le suffixe est optionnel
- Ces commandes ne travaillent qu'avec les noms

→ Aucun accès disque n'est réalisé

```
$ basename ce/petit/chemin/noisette.txt
noisette.txt
$ basename ce/petit/chemin/noisette.txt .txt
noisette
$ dirname ce/petit/chemin/noisette.txt
ce/petit/chemin
```

# Question basename et dirname

Qu'affichent les commandes suivantes ?

- `basename ce/petit/chemin/noisette.txt .png`

# Question basename et dirname

Qu'affichent les commandes suivantes ?

- `basename ce/petit/chemin/noisette.txt .png`

→ `noisette.txt`

- `basename ce/petit/chemin/noisette.txt tte.txt`

# Question basename et dirname

Qu'affichent les commandes suivantes ?

- `basename ce/petit/chemin/noisette.txt .png`

→ `noisette.txt`

- `basename ce/petit/chemin/noisette.txt tte.txt`

→ `noise`

- `basename ce/petit/chemin/`

# Question basename et dirname

Qu'affichent les commandes suivantes ?

- `basename ce/petit/chemin/noisette.txt .png`

→ `noisette.txt`

- `basename ce/petit/chemin/noisette.txt tte.txt`

→ `noise`

- `basename ce/petit/chemin/`

→ `chemin`

- `dirname ce/petit/chemin/`

# Question basename et dirname

Qu'affichent les commandes suivantes ?

- `basename ce/petit/chemin/noisette.txt .png`

→ `noisette.txt`

- `basename ce/petit/chemin/noisette.txt tte.txt`

→ `noise`

- `basename ce/petit/chemin/`

→ `chemin`

- `dirname ce/petit/chemin/`

→ `ce/petit`

- `dirname ce`

# Question basename et dirname

Qu'affichent les commandes suivantes ?

- `basename ce/petit/chemin/noisette.txt .png`

→ `noisette.txt`

- `basename ce/petit/chemin/noisette.txt tte.txt`

→ `noise`

- `basename ce/petit/chemin/`

→ `chemin`

- `dirname ce/petit/chemin/`

→ `ce/petit`

- `dirname ce`

→ `.`

# Répertoire courant

Commande `pwd` (*print working directory*)

- Afficher le chemin complet du répertoire courant

Commande interne `cd` (*change directory*)

- Change le répertoire de travail courant
- « `cd toto` » Va au sous-répertoire `toto`
- « `cd ..` » Remonte au répertoire parent
- « `cd` » Retourne au répertoire de connexion
- « `cd -` » Retourne au répertoire précédent (et l'affiche)

Attention à l'espace entre `cd` et `..` (ou `cd` et `-`)

# Répertoire courant

Commande `pwd` (*print working directory*)

- Afficher le chemin complet du répertoire courant

Commande interne `cd` (*change directory*)

- Change le répertoire de travail courant
- « `cd toto` » Va au sous-répertoire `toto`
- « `cd ..` » Remonte au répertoire parent
- « `cd` » Retourne au répertoire de connexion
- « `cd -` » Retourne au répertoire précédent (et l'affiche)

Attention à l'espace entre `cd` et `..` (ou `cd` et `-`)

## Question

- Que fait « `cd .` » ?

# Répertoire personnel (ou maison)

Chaque utilisateur a un répertoire de connexion (*home directory*)

- `cd` sans argument y mène

Caractère spécial « ~ » (tilde) du *shell*

- `~` seul devient le répertoire de connexion
- `~/` en début d'un argument fonctionne aussi
- `~toto` seul devient le répertoire de connexion de l'utilisateur `toto`
- `~toto/` en début d'un argument fonctionne aussi

# Répertoire personnel (ou maison)

Chaque utilisateur a un répertoire de connexion (*home directory*)

- `cd` sans argument y mène

Caractère spécial « `~` » (tilde) du *shell*

- `~` seul devient le répertoire de connexion
- `~/` en début d'un argument fonctionne aussi
- `~toto` seul devient le répertoire de connexion de l'utilisateur `toto`
- `~toto/` en début d'un argument fonctionne aussi

## Questions

- Qu'affiche « `echo ~ "~" ~123 ~/toto toto/~` » ?
- Y a-t-il une différence entre « `cd ~` » et « `cd` » ?

# Manipulation des fichiers

# Déplacer/renommer des fichiers

`mv` (*move*) déplace/renomme des fichiers (ou répertoires)

- `mv ancien nouveau`
- `mv f1 f2 f3 répertoire`

Attention: « `mv foo bar` » peut être ambigu

- Si `bar` n'existe pas: `foo` est **renommé** en `bar`
- Si `bar` est un fichier régulier: `bar` est **écrasé** par `foo`
- Si `bar` est un répertoire: `foo` est **déplacé** dans `bar`
  c'est équivalent à « `mv foo bar/foo` »

# Déplacer/renommer des fichiers

La commande `mv` est **dangereuse**

- Il n'y a pas de corbeille
- Les suppressions sont définitives

L'option `-i` (`--interactive`) permet de confirmer chaque action

L'option `-f` (`--force`) permet de forcer l'écrasement

# Autres commandes de manipulation

`cp` (*copy*) copie (ou écrase) des fichiers

- `-R` (`-r`, `--recursive`) copie récursivement des répertoires
- `-p` (`--preserve`) préserver des attributs (voir plus loin)
- `-a` (`--archive`) récursif et préserve tous les attributs (GNU)
- `-i` ou `-f`: écrasement interactif ou forcé

`rm` (*remove*) supprime des fichiers

- `-R` (`-r`, `--recursive`) supprime récursivement des répertoires
- `-i` ou `-f`: suppression interactive ou forcée

# Manipulations à risque

## Questions

- Que fait « `rm * .txt` » ?
- Que fait « `rm -rf .*` » ?

# Commandes de manipulation des répertoires

`mkdir` (*make directory*) crée des répertoires (vides)

- `-p` (`--parents`) crée les répertoires parents intermédiaires

`rmdir` (*remove directory*) supprime des répertoires vides

- `-p` (`--parents`) supprime les répertoires parents vides intermédiaires

# Types de fichiers

# Fichier = Séquence d'**octets**

- Octet (*byte*) = l'unité d'**information** d'un fichier
- La **signification des octets** est du ressort de l'application

```
$ vim hello.c
```

# Fichier = Séquence d'**octets**

- `vim` interprète `hello.c` comme du texte

$\rightarrow$ le code source C est du texte

# Fichier = Séquence d'**octets**

- `vim` interprète `hello.c` comme du texte

$\rightarrow$ le code source C est du texte

```
$ vim debian.png
```

# Fichier = Séquence d'**octets**

- `vim` interprète `hello.c` comme du texte

→ le code source C est du texte

```
$ vim debian.png
```

- `vim` interprète `debian.png` comme du texte

→ mais ce n'est pas du texte

Quand un fichier n'a pas de sens **pour elle**, une application

- Refuse de travailler (message d'erreur)
- Ou fait n'importe quoi

# Questions des fichiers

## Que fait « `cat debian.png` » ?

Quel est le risque ?

# Questions des fichiers

## Que fait « `cat debian.png` » ?

Quel est le risque ?

- Ça envoie directement les **octets** du fichier au terminal,
- Ils peuvent être **interprétés** comme des caractères de contrôle du terminal,
- Et briser (mal-configurer) le comportement du terminal

`reset` (extra) — réinitialise le terminal

# Questions des fichiers

## Que fait « `cat debian.png` » ?

Quel est le risque ?

- Ça envoie directement les **octets** du fichier au terminal,
- Ils peuvent être **interprétés** comme des caractères de contrôle du terminal,
- Et briser (mal-configurer) le comportement du terminal

`reset` (extra) — réinitialise le terminal

## Que fait « `vim debian.svg` » ?

# Questions des fichiers

## Que fait « `cat debian.png` » ?

Quel est le risque ?

- Ça envoie directement les **octets** du fichier au terminal,
- Ils peuvent être **interprétés** comme des caractères de contrôle du terminal,
- Et briser (mal-configurer) le comportement du terminal

`reset` (extra) — réinitialise le terminal

## Que fait « `vim debian.svg` » ?

Cela affiche le source de l'image (en XML)

On y reviendra plus tard...

# Extra: Voir les octets quand même

`xxd` (extra) (et `hd` (extra) et `od` (POSIX)) affiche la valeur de chaque octet

```
$ xxd debian.png
00000000: 89 50 4e 47 0d 0a 1a 0a  .PNG....
00000008: 00 00 00 0d 49 48 44 52  ....IHDR
00000010: 00 00 01 0a 00 00 01 06  ........
[...]

$ xxd debian.svg
00000000: 3c 3f 78 6d 6c 20 76 65  <?xml ve
00000008: 72 73 69 6f 6e 3d 22 31  rsion="1
00000010: 2e 30 22 20 65 6e 63 6f  .0" enco
[...]
```

Les détails une autre fois...

# Texte et binaire

Fichier **texte**

- Contenu = une séquence de caractères (imprimables)
- Peut se lire et se modifier avec éditeur de texte
- Codages: ASCII, ISO-8859-15, UTF-8, etc.
  man `ascii`
- Représentations des fins de lignes: Unix (`\n`) et DOS (`\r\n`)

Fichier **binaire**

- Fichier non-texte
- Parfois difficile de trancher entre texte/binaire
- Mais généralement non ambigu

# Extensions

Un fichier peut avoir une **extension** (ou non):

- Extension = chaîne après le dernier point
- Exemple avec extensions:

  `readme.txt`, `rapport.pdf`, `Arbre.java`, `logo.png`, `archive.tar.gz`, `diagram.dot.png`
- Exemple sans extension: `Makefile`, `LICENSE`, `ls`

Juste une convention pour aider l'utilisateur (et les logiciels)

# Quiz d'extensions

- **.txt**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**: fichier source Java
- **.zip**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**: fichier source Java
- **.zip**: archive compressée
- **.ini**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**: fichier source Java
- **.zip**: archive compressée
- **.ini**: fichier de configuration
- **.mkv**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**: fichier source Java
- **.zip**: archive compressée
- **.ini**: fichier de configuration
- **.mkv**: fichier multimédia (*matroska*)
- **.class**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**: fichier source Java
- **.zip**: archive compressée
- **.ini**: fichier de configuration
- **.mkv**: fichier multimédia (*matroska*)
- **.class**: fichier compilé Java
- **.deb**

# Quiz d'extensions

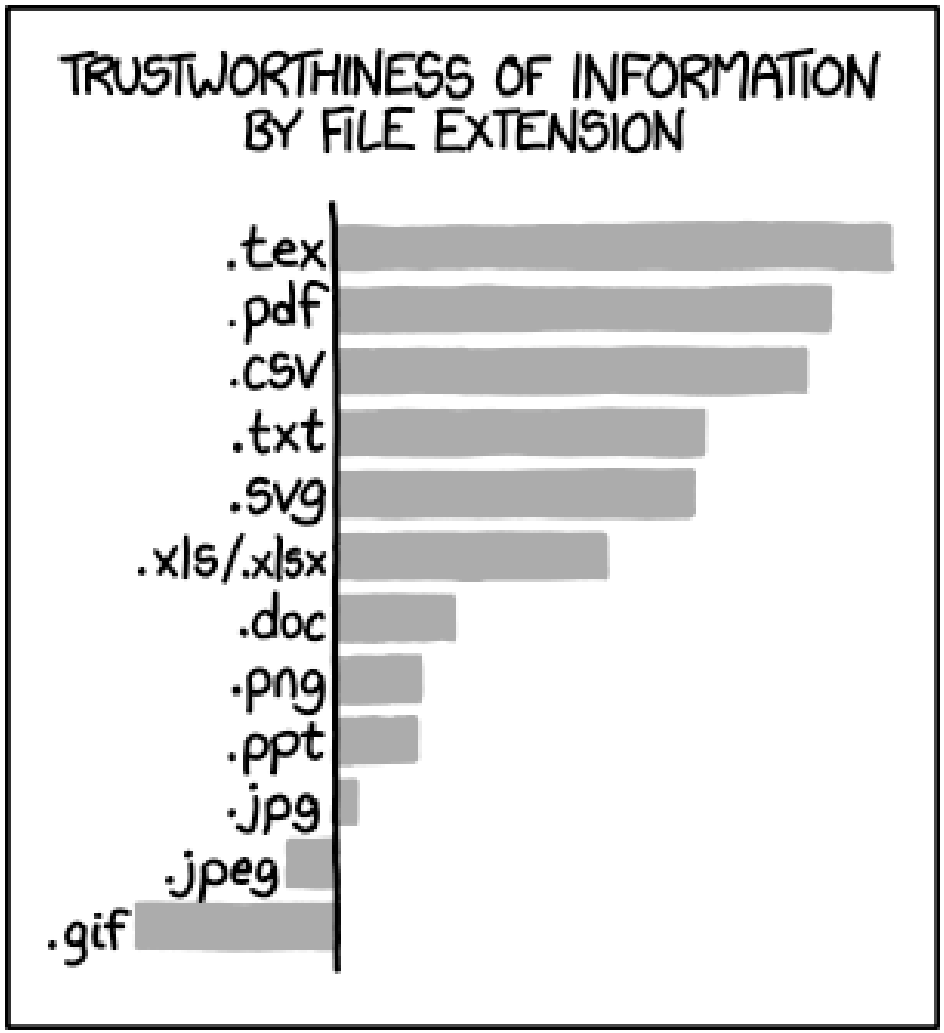
- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**: fichier source Java
- **.zip**: archive compressée
- **.ini**: fichier de configuration
- **.mkv**: fichier multimédia (*matroska*)
- **.class**: fichier compilé Java
- **.deb**: paquet Debian
- **.so**

# Quiz d'extensions

- **.txt**: fichier texte brut
- **.jpg**: image
- **.html**: page web
- **.java**: fichier source Java
- **.zip**: archive compressée
- **.ini**: fichier de configuration
- **.mkv**: fichier multimédia (*matroska*)
- **.class**: fichier compilé Java
- **.deb**: paquet Debian
- **.so**: bibliothèque dynamique (*shared object*)

Rappel: l'extension n'est qu'une indication (pas toujours fiable)

# Extensions de fichier




Source: https://xkcd.com/1301/ (2013)

# Type de média (MIME)

Meilleur identifiant de format de données sur internet

- Courriel: MIME = Multipurpose Internet Mail Extensions
- Web: téléchargement et téléversement
- Bureau: applications associées, icones, etc.

Normalisé par l'IANA (*Internet Assigned Numbers Authority*)

## Spécification

`type/[arbre.]sous-type[+suffixe]...[; options]`

- `text/java; charset=UTF-8`: code source Java
- `application/pdf`: fichier PDF
- `text/plain`: fichier texte quelconque
- `application/octet-stream`: fichier binaire quelconque

# Identification des types de fichiers

L'identification des types de fichiers peut se baser sur

- L'extension du nom du fichier
- Des octets spécifiques du fichier (*magic number*)
- Un catalogue d'heuristiques spécifiques à chaque type de fichier

`file` (et `xdg-mime` (LSB)) — déterminer un type de fichier

```
$ file *
bonjour.txt:        UTF-8 Unicode text
cat.pdf:            PDF document, version 1.4
debian.png:         PNG image data, 266 x 262
hello.c:            C source, ASCII text
blob:               data
secret.inconnu:     PNG image data, 266 x 262

$ xdg-mime query filetype secret.inconnu
image/png
```

# Fichier suspect


Source: https://xkcd.com/1247/ (2013)

# Fichiers sous Unix

Un principe Unix de base: tout est fichier

## Fichiers réguliers

- Textes, exécutables, code source, images...
- Contenu décidé par l'utilisateur

## Fichiers spéciaux

- Répertoires, fichiers périphériques (dans `/dev`), liens symboliques, tubes nommés...
- Manipulation par des commandes spéciales
- Règles au cas par cas

# Méta-données des fichiers

# Méta-données des fichiers (attributs)

Autres options de `ls`:

- `-l` format d'affichage long
- `-i` (`--inode`) afficher le numéro d'index

```
$ ls -li /etc/passwd
30169 -rw-r--r-- 1 root root 2652 nov 16 2017 /etc/passwd
```

- « `30169` » numéro d'index (inœud, *inode*)
- « `-` » type de fichier (fichier régulier, répertoire…)
- « `rw-r--r--` » droits (utilisateur, groupe, autre)
- « `1` » nombre de liens durs
- « `root` » utilisateur propriétaire
- « `root` » groupe propriétaire
- « `2652` » taille du fichier (en octets)
- « `nov 16 2017` » date de dernière modification

# Trier les fichiers

Autres options de `ls`:

- `-S` trier par taille
- `-t` trier par date de modification
- `-r` (`--reverse`) trier à l'envers

```
$ ls -lSr
-rw-r--r-- 1          72 sep 12 20:49 hello.c
-rw-r--r-- 1         460 sep 12 20:49 bonjour.txt
-rw-r--r-- 1        6863 sep 12 20:49 debian.svg
-rw-r--r-- 1       15197 sep 11 18:50 debian.png
-rw-r--r-- 1       41493 sep  6 20:30 linux.au
-rw-r--r-- 1 2729345024 sep 10 16:01 debian.iso
```

# État complet d'un fichier

`stat` — affiche l'état complet d'un fichier (GNU)

- `-c` (`--format`) utilise le format indiqué

```
$ stat /etc/passwd
Fichier : /etc/passwd
Taille : 2652  Blocs : 8  Blocs d'E/S : 4096  fichier
Périphérique : 807h/2055d  Inoeud : 30169  Liens : 1
Accès : (0644/-rw-r--r--)  UID : (0/root)  GID : (0/root)
Accès  : 2018-06-05 09:39:01.288330072 -0400
Modif. : 2017-11-16 09:58:36.656200344 -0500
Changt : 2017-11-16 09:58:36.680200043 -0500
Créé   : -

$ stat -c '%n %U %s' /etc/passwd
/etc/passwd root 2652
```

# Taille des fichiers

Unité de base: l'**octet** (**byte**), qui contient 8 **bits**

- 1 octet = 1o =
- 1 byte = 1B =
- 8 bits = 8b

→ Attention à la **confusion** byte et bit

# Multiples d'octets

Système international d'unités (SI)

- Préfixes habituels: kilo, méga, giga, téra, péta, etc.
- En puissance de 10
- 1km $= 10^3$ mètres $=$ 1000 mètres

Commission électrotechnique internationale (CEI) 1998

- Préfixes: kibi (kilo binaire), mébi (méga binaire), etc.
- En puissance de 2
- 1kio $= 2^{10}$ octets $=$ 1024 octets

Traditionellement, pour les informaticiens et les applications

- Préfixes habituels: kilo, méga, giga, etc.
- En puissance de 2
- 1ko $= 2^{10}$ octets $=$ 1024 octets

$\rightarrow$ Encore plus de **confusion**

# Quelques multiples

- 1 kibioctet (kio) $= 2^{10}$ octets $=$ 1 024 octets
  Code source de `cowsay` $\approx$ 4 kio
- 1 mébioctet (Mio) $= 2^{20}$ octets $=$ 1 024 kio
  $=$ 1 048 576 octets
  Photo JPG de téléphone $\approx$ 4 Mio
- 1 gibioctet (Gio) $= 2^{30}$ octets $=$ 1 024 Mio
  $=$ 1 073 741 824 octets
  Disque Blu-ray $\approx$ 25 Gio
- 1 tébioctet (Tio) $= 2^{40}$ octets $=$ 1 024 Gio
  $=$ 1 099 511 627 776 octets
  Disque dur moderne $\approx$ 1 Tio
- 1 pébioctet (Pio) $= 2^{50}$ octets $=$ 1 024 Tio
  $=$ 1 125 899 906 842 624 octets
  Les archives du CERN contiennent plus de 200 Pio

# Lister la taille des fichiers

Autres options de `ls` (GNU):

- `-h` (`--human-readable`) afficher les tailles en multiples de 1024
- `--si` afficher les tailles en multiples de 1000

```
$ ls -lh
-rw-r--r-- 1  460 sep 12 13:31 bonjour.txt
-rw-r--r-- 1 2,6G sep 10 16:01 debian.iso
-rw-r--r-- 1  15K sep  7 11:36 debian.png
-rw-r--r-- 1 6,8K jun  1 00:50 debian.svg
-rw-r--r-- 1  41K fév 15  2000 linux.au

$ ls -l --si
-rw-r--r-- 1  460 sep 12 13:31 bonjour.txt
-rw-r--r-- 2 2,8G sep 10 16:01 debian.iso
-rw-r--r-- 1  16k sep  7 11:36 debian.png
-rw-r--r-- 1 6,9k jun  1 00:50 debian.svg
-rw-r--r-- 1  42k fév 15  2000 linux.au
```

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$ 130ko
- Le jeu vidéo *Zelda a link to the past* (1991) $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$ 130ko
- Le jeu vidéo *Zelda a link to the past* (1991) $\approx$ 1Mo
- Le jeu vidéo *Zelda: breath of the wild* (2017) $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$ 130ko
- Le jeu vidéo *Zelda a link to the past* (1991) $\approx$ 1Mo
- Le jeu vidéo *Zelda: breath of the wild* (2017) $\approx$ 13Go
- Le binaire exécutable de `bash` $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$ 130ko
- Le jeu vidéo *Zelda a link to the past* (1991) $\approx$ 1Mo
- Le jeu vidéo *Zelda: breath of the wild* (2017) $\approx$ 13Go
- Le binaire exécutable de `bash` $\approx$ 1Mo
- Le binaire exécutable de `dash` $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$ 130ko
- Le jeu vidéo *Zelda a link to the past* (1991) $\approx$ 1Mo
- Le jeu vidéo *Zelda: breath of the wild* (2017) $\approx$ 13Go
- Le binaire exécutable de `bash` $\approx$ 1Mo
- Le binaire exécutable de `dash` $\approx$ 100ko
- Un film HD de 2h $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$ 130ko
- Le jeu vidéo *Zelda a link to the past* (1991) $\approx$ 1Mo
- Le jeu vidéo *Zelda: breath of the wild* (2017) $\approx$ 13Go
- Le binaire exécutable de `bash` $\approx$ 1Mo
- Le binaire exécutable de `dash` $\approx$ 100ko
- Un film HD de 2h $\approx$ 3 ou 4 Go
- Debian (17 architectures et 4 versions actives) $\approx$

# Quiz de taille

Quelle est la taille approximative, en octet de

- Le fichier PDF de cette présentation $\approx$ 400ko
- Les articles du Wikipédia anglophone $\approx$ 60Go
- Le jeu vidéo *Zelda* original (1986) $\approx$ 130ko
- Le jeu vidéo *Zelda a link to the past* (1991) $\approx$ 1Mo
- Le jeu vidéo *Zelda: breath of the wild* (2017) $\approx$ 13Go
- Le binaire exécutable de `bash` $\approx$ 1Mo
- Le binaire exécutable de `dash` $\approx$ 100ko
- Un film HD de 2h $\approx$ 3 ou 4 Go
- Debian (17 architectures et 4 versions actives) $\approx$ 2.8To

# Questions de confusion

## Combien d'octets dans...

...un disque dur d'« un téra » ?

# Questions de confusion

## Combien d'octets dans...

...un disque dur d'« un téra » ?

- 1 téraoctet (To) $= 10^{12}$ octets $=$ 1 000 Go
  $=$ 1 000 000 000 000 octets $\approx$ **0.91Tio**
- 1 tébioctet (Tio) $= 2^{40}$ octets $=$ 1 024 Gio
  $=$ 1 099 511 627 776 octets $\approx$ 1.10To

# Questions de confusion

## Combien d'octets dans...

...un disque dur d'« un téra » ?

- 1 téraoctet (To) $= 10^{12}$ octets $=$ 1 000 Go $=$ 1 000 000 000 000 octets $\approx$ **0.91Tio**
- 1 tébioctet (Tio) $= 2^{40}$ octets $=$ 1 024 Gio $=$ 1 099 511 627 776 octets $\approx$ 1.10To

## Combien d'octets par secondes avec...

...une connexion internet de « 50 méga » ?

# Questions de confusion

## Combien d'octets dans...

...un disque dur d'« un téra » ?

- 1 téraoctet (To) $= 10^{12}$ octets $=$ 1 000 Go
  $=$ 1 000 000 000 000 octets $\approx$ **0.91Tio**
- 1 tébioctet (Tio) $= 2^{40}$ octets $=$ 1 024 Gio
  $=$ 1 099 511 627 776 octets $\approx$ 1.10To

## Combien d'octets par secondes avec...

...une connexion internet de « 50 méga » ?

- 50Mbit/s $=$ 50 000 000bit/s $=$ 6 250 000o/s $\approx$ **5.96Mio/s**

# Espace disque libre

`df` (*disk free*) indique l'espace libre et occupé sur un disque

- `-k` la taille est en bloc de 1024 octets

→ La taille par défaut des blocs n'est pas stable: 512 chez POSIX, 1024 chez GNU, non-spécifiée chez LSB. Utilisez `-k` pour être portable.

- `-h` (`--human-readable`) utilise des multiples de 1024 (GNU)

```
$ df -h .
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
/dev/sda7          162G    126G   28G  82% /
```

## Abus de langage

On parle abusivement de **disque** même s'il n'y a pas de **disque physique**

# Points de montage et système de fichiers

- Les fichiers sont dans une **hiérarchie de fichiers** unique
- « / » est la **racine** de cette hiérarchie de fichiers
- La hiérarchie peut être composée de plusieurs **systèmes de fichiers**
- Les *systèmes de fichiers* sont **autonomes** et peuvent être sur des disques, partitions, clés USB, etc. (plus tard...)

`df`, par défaut, affiche la liste des systèmes de fichiers

```
$ df -h
Sys.fich.  Taille Utilisé Dispo Uti% Monté sur
udev         7,8G       0  7,8G   0% /dev
tmpfs        1,6G     75M  1,5G   5% /run
/dev/sda7    162G    126G   28G  82% /
/dev/sdb1    917G    657G  214G  76% /media/privat/shark
```

# Taille occupée

`du` (*disk usage*) estime l'espace disque occupé

- `-k` la taille est en bloc de 1024 octets
- `-a` (`--all`) affiche l'occupation des fichiers individuels
- `-s` (`--summarize`) n'affiche que la taille totale
- `-x` (`--one-file-system`) ne change pas de système de fichiers
- `-h` (`--human-readable`) utilise des multiples de 1024 (GNU)

```
$ du -k
2665512 ./media
40  ./special
2665552 .
```

# Question: les plus gros

Quels sont les 5 plus gros répertoires de `/usr/share` ?

# Question: les plus gros

## Quels sont les 5 plus gros répertoires de `/usr/share` ?

```
$ du /usr/share | sort -n | tail -n 5
392116  /usr/share/doc/texlive-doc
394140  /usr/share/help
748756  /usr/share/locale
820048  /usr/share/doc
3993332 /usr/share
```

## Même question mais avec des multiples de 1024

# Question: les plus gros

## Quels sont les 5 plus gros répertoires de `/usr/share` ?

```
$ du /usr/share | sort -n | tail -n 5
392116  /usr/share/doc/texlive-doc
394140  /usr/share/help
748756  /usr/share/locale
820048  /usr/share/doc
3993332 /usr/share
```

## Même question mais avec des multiples de 1024

```
$ du -h /usr/share | sort -h | tail -n 5
383M    /usr/share/doc/texlive-doc
385M    /usr/share/help
732M    /usr/share/locale
801M    /usr/share/doc
3,9G    /usr/share
```

# Taille disponible

`quota` indique les limites des utilisateurs et leur utilisation de l'espace disque (extra)

- `-s` (`--human-readable`) utilise des multiples de 1024

Les quotas sont :

- définis par l'administrateur
- spécifiques à chaque système de fichiers
- comptabilisés pour chaque utilisateur

```
$ quota -s
Disk quotas for user privat_j (uid 442541020):
Système fichiers   blocs   quota  limite
192.168.10.227:/usagers/gf391175
                   2432M   7813M   8204M
```

# Droits et utilisateurs

# Droits et utilisateurs

Les fichiers et répertoires ont des **propriétaires** et des **droits** qui limitent/autorisent l'accès à certains **utilisateurs**.

- Chaque utilisateur est associé à une identité et à des groupes
- Chaque fichier du système possède un utilisateur propriétaire et un groupe propriétaire

Par défaut et en général:

- Un utilisateur peut lire les fichiers des autres utilisateurs
- Un utilisateur ne peut modifier les fichiers des autres utilisateurs
- Un utilisateur peut créer des fichiers dans son répertoire personnel

# Utilisateurs et groupes

Commande `id` — affiche les identifiants d'utilisateur et de de groupes

```
$ id
uid=1000(privat) gid=1000(privat)
groupes=1000(privat),4(adm),6(disk)
```

Un utilisateur peut appartenir à plusieurs groupes

À l'interne, les utilisateurs et groupes sont identifiés par des numéros.

- `/etc/passwd` info sur les utilisateurs
- `/etc/group` info sur les groupes

```
$ cat /etc/passwd
privat:x:1000:1000:Jean Privat,,,:/home/privat:/bin/bash
```

# Super-utilisateur

Le super utilisateur (*super-user*)

- S'appelle **root** (traditionnellement)
- Est le compte de l'administrateur
- Son uid est 0

Il a de grands pouvoirs: contrôle **complet** du système

- Ne respecte pas les droits des fichiers
- Peut contrôler les programmes des utilisateurs
- Peut configurer le système et le réseau
- Peut installer de nouveaux programmes et pilotes

Et de grandes responsabilités

- L'administrateur a aussi un compte utilisateur normal
- Ne passe en **root** que lorsque c'est nécessaire

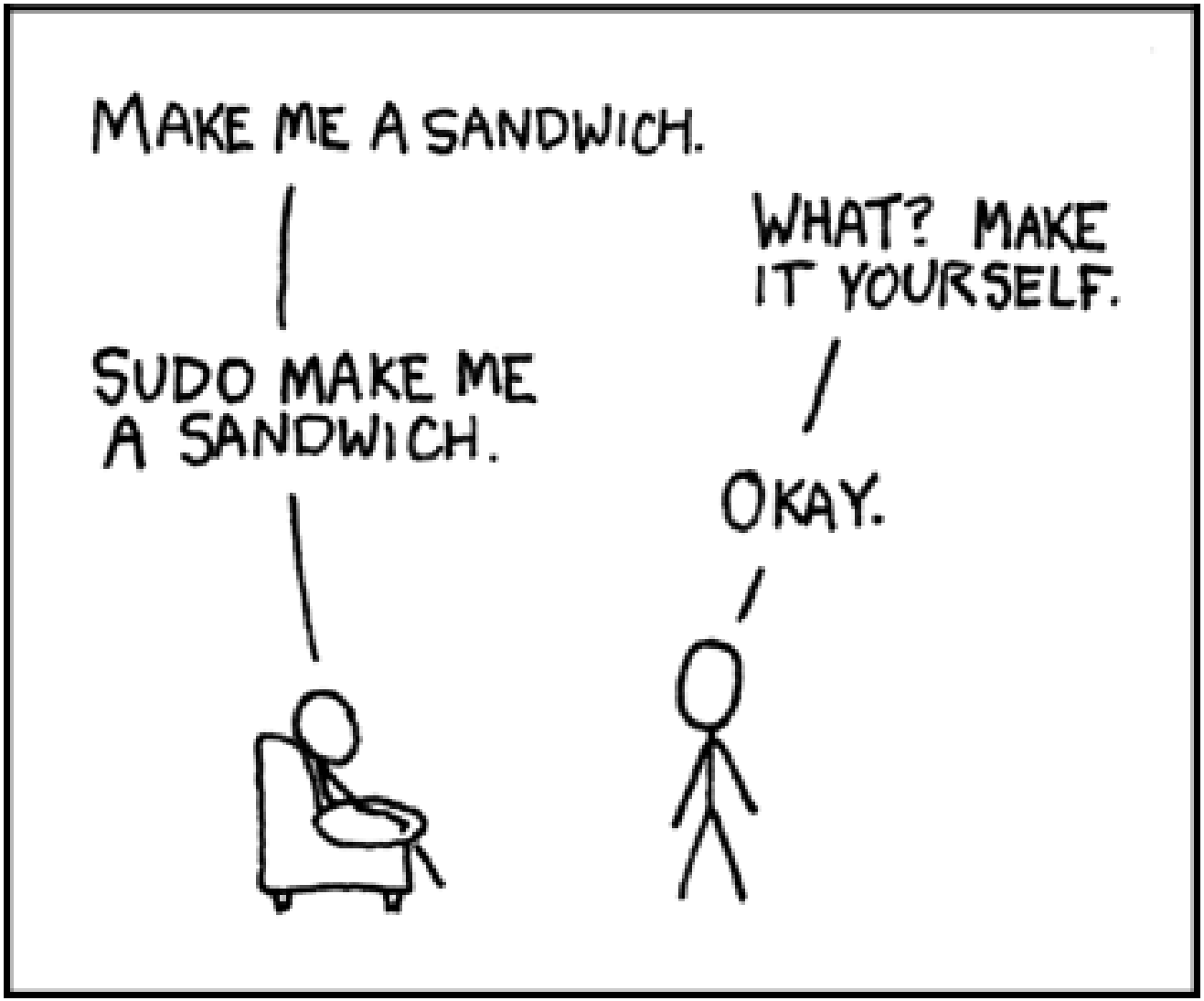
# Passer en root

`sudo` (extra) – exécute une commande sous un autre utilisateur

- Moderne (et le défaut de nos jours)
- Change temporairement d'utilisateur (root par défaut)
- Exécute une commande (défaut)
- Nécessite une configuration préalable
- Nécessite le mot de passe de l'**utilisateur courant**
- `-s` pour lancer un shell
- *Logue* (enregistre dans un journal) les utilisations

```
$ sudo apt install cowsay
[...]
$ grep sudo /var/log/auth.log
Sep 12 22:38:43 lama sudo:   privat
  PWD=/home/privat/ens/INF1070/slides
  COMMAND=/usr/bin/apt install cowsay
```

# Sandwich




Source: https://xkcd.com/149/ (2006)

# Passer en root (version alternative)

`su` (*superuser* ou *switch user*) (LSB)

- Moins recommandé que sudo (historique)
- Change temporairement d'utilisateur (root par défaut)
- Lance un shell (par défaut)
- Nécessite le mot de passe de l'**utilisateur demandé**
- `-c` pour exécuter une commande

## Inconvénients

- Nécessite l'existence d'un mot de passe root
- Ne logue pas les commandes
- Ne permet pas un contrôle fin par utilisateur

# Modèle de sécurité UNIX

- Les protections Unix isole et protège les utilisateurs **entre eux**
- Mais ne protège pas l'utilisateur de **lui-même**

## Ne fonctionne pas

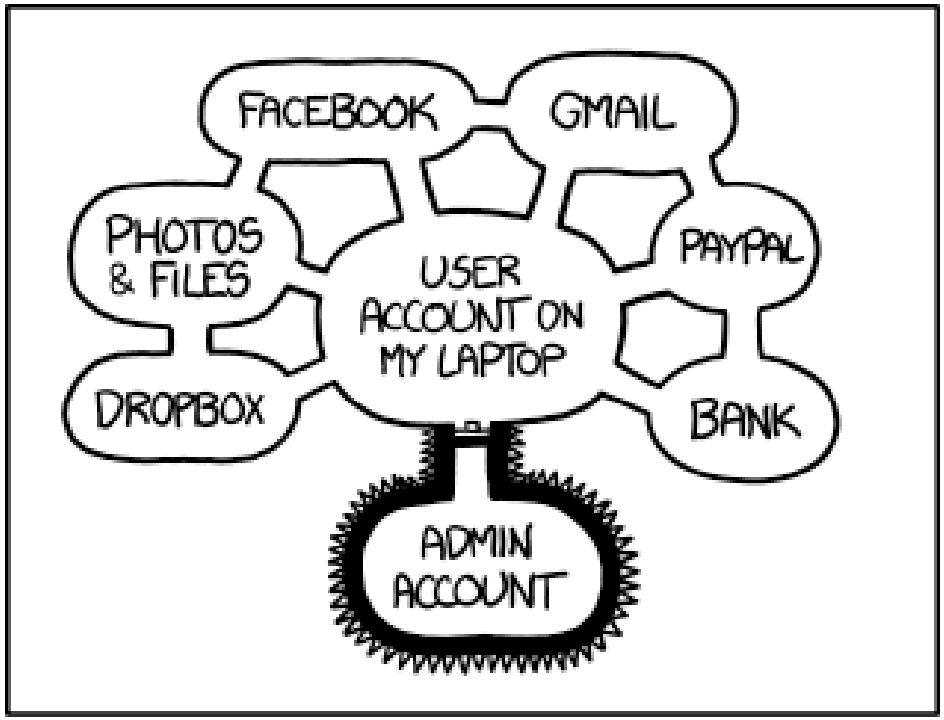
```
$ rm -rf /usr
```

## Fonctionne

```
$ rm -rf ~
```

on y reviendra...

# Autorisation




IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

Source: https://xkcd.com/1200/ (2013)

# Droits des fichiers traditionnels Unix

## 3 catégories d'accès (ugo)

- u (*user*/utilisateur) l'utilisateur propriétaire
- g (*group*/groupe) le groupe propriétaire
- o (*other*/autre) les autres

## 3 permissions par catégorie (rwx)

Pour un fichier

- r (*read*/lecture) on peut lire le contenu
- w (*write*/écriture) on peut modifier le contenu
- x (*execute*/exécution) on peut exécuter le programme

Pour un répertoire

- r (*read*/lecture) on peut lister les entrées
- w (*write*/écriture) on peut ajouter/supprimer des entrées
- x (*execute*/exécution) on peut accéder aux entrées

# Exercice

```
drwxr-xr-x 6 root root 4096 jui  5 10:05 .
-rw-rw-r-- 1 root root   10 jui  5 10:40 f1
-rw-rw-rw- 1 root root   13 jui  5 10:40 f2
-rwxrwx--x 1 root root 6304 jui  5 10:40 f3
drwxrwx--x 2 root root 4096 jui  5 10:40 d1
-rw-rw-rwx 1 root root   13 jui  5 10:40 d1/f
drwxrwxrw- 2 root root 4096 jui  5 10:40 d2
-rw-rw-rwx 1 root root   16 jui  5 10:40 d2/f
drwxrwx-wx 2 root root 4096 jui  5 10:40 d3
-rw-rw-r-x 1 root root   16 jui  5 10:40 d3/f
```

Pour chaque fichier, est-ce qu'on peut, sans être root:

- Afficher le contenu ?
- Afficher les méta-données ?
- Modifier le contenu ?
- Renommer le fichier ?
- Supprimer le fichier ?

# Solution (1/4)

## Afficher le contenu

- Avec `cat` et `ls` par exemple
- Il faut le droit `r` du fichier ou répertoire
- Et `x` des répertoires intermédiaires

→ `f1`, `f2`, `d2`, `d1/f`, `d3/f`: OK

→ `f3`, `d1`, `d3`: manque le `r` du fichier ou répertoire

→ `d2/f`: manque `x` sur `d2`

## Afficher les méta-données

- Avec `stat` ou `ls -ld` par exemple
- Il faut le droit `x` des répertoires intermédiaires

→ `f1`, `f2`, `f3`, `d1`, `d1/f`, `d2`, `d3`, `d3/f`: OK

→ `d2/f`: manque `x` sur `d2`

# Solution (2/4)

## Modifier le contenu du fichier

- Avec `echo toto >` par exemple
- Il faut le droit `w` du fichier
- Et `x` des répertoires intermédiaires

→ `f2`, `d1/f`: OK

→ `d2/f`: manque `x` sur `d2`

→ `f1`, `f3` et `d3/f`: manque `w` du fichier

# Solution (3/4)

## Renommer (et/ou déplacer)

- Avec `mv` par exemple
- Il faut le droit `w` des répertoires parents
- Et `x` des répertoires intermédiaires

→ `d3/f`: OK

→ `f1`, `f2`, `f3`, `d1`, `d2`, `d3`: manque `w` sur `.`

→ `d1/f`: manque `w` sur `d1`

→ `d2/f`: manque `x` sur `d2`

# Solution (4/4)

## Supprimer

- Avec `rm` et `rmdir`par exemple
- Comme pour `renommer` en fait
- Pour un répertoire, il faut aussi qu'il soit vide

## Créer un nouveau

- Avec `touch` et `mkdir` par exemple
- Comme pour `renommer` en fait

## Copier

- Avec `cp` et `cp -r` par exemple
- Il faut pouvoir lire le contenu de chaque source
- Et créer les fichiers et répertoires dans le répertoire cible

# Octal

**Octal** = représentation des nombres en base 8 (8 chiffres, de 0 à 7)

- r = 4
- w = 2
- x = 1

Les droits d'un fichiers sous forme octale

- minimum = `---` = 0 + 0 + 0 = 0
- maximum = `rwx` = 4 + 2 + 1 = 7
- `rw-r-----` = 0640

# Octal

**Octal** = représentation des nombres en base 8 (8 chiffres, de 0 à 7)

- r = 4
- w = 2
- x = 1

Les droits d'un fichiers sous forme octale

- minimum = `---` = 0 + 0 + 0 = 0
- maximum = `rwx` = 4 + 2 + 1 = 7
- `rw-r-----` = 0640

## Question

- Quel sont les droits de `/etc/passwd` et `/etc/shadow` en octal ?

# Modifier les droits

Commande `chmod` (*change file mode*)

- `chmod octal fichier`
- `chmod mode fichier`

où `mode` contient

- les catégories d'utilisateurs: **u**, **g**, **o** ou **a** (*all*)
- une opération: **+** (ajouter), **-** (enlever), **=** (mettre exactement)
- les droits: **r**, **w**, **x**

Options utiles

- `-R` récursif
- `-v` verbeux

# Exemple

```
$ ls -l fichier
-rw-r--r-- 1 privat privat fichier
$ chmod 640 fichier ; ls -l fichier
-rw-r----- 1 privat privat fichier
$ chmod +x fichier ; ls -l fichier
-rwxr-x--x 1 privat privat fichier
$ chmod o-x,g+w fichier ; ls -l fichier
-rwxrwx--- 1 privat privat fichier
$ chmod a-x fichier ; ls -l fichier
-rw-rw---- 1 privat privat fichier
```

# Exemple

```
$ ls -l fichier
-rw-r--r-- 1 privat privat fichier
$ chmod 640 fichier ; ls -l fichier
-rw-r----- 1 privat privat fichier
$ chmod +x fichier ; ls -l fichier
-rwxr-x--x 1 privat privat fichier
$ chmod o-x,g+w fichier ; ls -l fichier
-rwxrwx--- 1 privat privat fichier
$ chmod a-x fichier ; ls -l fichier
-rw-rw---- 1 privat privat fichier
```

## Questions

- Quels sont les droits utiles d'un répertoire personnel ?
- Peut-on faire `chmod 000` d'un fichier ?
- Quel est le risque de `chmod 777` d'un répertoire ?

# Changer les propriétaires

`chown` (*change owner*) et `chgrp` (*change group*)

```
$ ls -l fichier
-rwxrwx--- 1 privat privat fichier
$ chgrp adm fichier
-rwxrwx--- 1 privat adm fichier
$ chown test fichier
chgrp: modification ...: Opération non permise
$ sudo chown test fichier ; ls -l fichier
-rwxrwx--- 1 test adm 0 jui  5 09:09 fichier
$ sudo chown privat:privat fichier ; ls -l fichier
-rwxrwx--- 1 privat privat fichier
```

# Masque de création de fichiers

Quels droits pour les fichiers créés

- C'est l'utilisateur qui choisit ?
- C'est le programme qui choisit ?
- C'est l'administrateur qui choisit ?
- C'est la combinaison des trois ?

umask (*user mask*) — masque de création de fichiers

```
$ umask
0022
```

- Les droits de l'umask sont éliminés des fichiers créés
- Un nouvel umask ne s'applique que pour les **nouveaux** fichiers et pour la session shell **courante**

# Exemple de umask

```
$ umask
0022

$ touch f1
$ ls -l f1
-rw-r--r-- f1

$ umask 000
$ touch f2
$ ls -l f2
-rw-rw-rw- f2

$ umask 077
$ touch f3
$ ls -l f3
-rw------- f3
```

# *setuid, setgid*

Marqueurs (bits) supplémentaires

## ***setuid* (4000, u+s)**

- **Fichier exécutable** le fichier est lancé avec les droits de l'utilisateur propriétaire

## ***setgid* (2000, g+s)**

- **Fichier exécutable** Le fichier est lancé avec les droits du groupe propriétaire
- **Répertoires** un nouveau fichier héritera le groupe du répertoire (et non celui de l'utilisateur)

Utilisé pour augmenter automatiquement les privilèges

```
$ ls -l /usr/bin/sudo
-rwsr-xr-x 1 root root 140944 jun  5  2017 /usr/bin/sudo
```

## *sticky* (1000, +t) (extra)

- **Répertoires** un utilisateur ne peut supprimer un fichier du répertoire sans en être propriétaire du fichier ou du répertoire

Utilisé pour faire des répertoires temporaires communs

```
$ ls -ld /tmp
drwxrwxrwt 14 root root 12288 sep 24 21:43 /tmp
```

# Liste de contrôle d'accès des fichiers (ACL)

les ACL offrent des droits plus fins

- Liste de groupes et d'utilisateurs
- Droits par défaut des nouveaux fichiers

Commandes `getfacl` et `setfacl` (extra) ; voir man `acl`

## Avantages

- Meilleures répertoires partagés entre utilisateurs
- Compatibilité avec Windows

## Inconvénients

- 3 normes: POSIX, NFS et Windows
- Complexe
- Utilisation de niche

# Contrôle d'accès obligatoire

ou MAC (*mandatory access control*)

L'**administrateur** impose des protections

- En fonction des fichiers
- En fonction des utilisateurs
- En fonction des applications

Exemples

- SELinux: défaut chez Red Hat et Android
- AppArmor: défaut chez Debian et Ubuntu

## Contrôle d'accès discrétionnaire

ou DAC (*Discretionary access control*)
Le propriétaire d'un fichier décide des protections
Exemple: les droits rwx classiques

# Autres attributs

## Attributs de fichiers Linux

Marqueurs (bits) supplémentaires

- Spécifiques à des systèmes de fichiers
- Voir `lsattr` et `chattr` (extra)

## Attributs étendus

Paires nom:valeur associées aux fichiers

- Peuvent servir aux MAC
- Peuvent servir à implanter les ACL
- Peuvent être utilisés librement
- Voir `getfattr`, `setfattr` et `xattr` (extra)

# Dates des fichiers

# Dates des fichiers (Unix)

Trois types de dates

- atime : date de dernier accès au fichier (lecture)
- mtime : date de dernière modification du fichier
- ctime : date de dernière modification des métadonnées

```
$ stat /etc/passwd
  Fichier : /etc/passwd
   Taille : 2720   Blocs : 8  Blocs d'E/S : 4096
fichier
Périphérique : 807h/2055d  Inoeud : 3020392  Liens : 1
Accès : (0644/-rw-r--r--)  UID : (0/root)   GID : (0/root)
 Accès : 2018-07-08 14:29:13.536125040 -0400
Modif. : 2018-06-20 09:20:56.939040056 -0400
Changt : 2018-06-20 09:20:56.963040057 -0400
  Créé : -
```

# Heure de la machine

`date` – affiche et configure la date

- `+` utilise un format différent
- `-r` (`--reference`) utilise la date de dernière modification d'un fichier (GNU)

```
$ date
jeudi 13 septembre 2018, 15:32:11 (UTC-0400)
$ date +%H:%M
15:32
$ date -r debian.png
vendredi 7 septembre 2018, 11:36:44 (UTC-0400)
```

# Dates (Unix)

Représentation: temps Unix

- Temps écoulé depuis *epoch*, le premier janvier 1970 UTC.
- Stocké en nanosecondes (historiquement en secondes)

```
$ stat -c '%.Y = %y' /etc/passwd
1529500856,939040056 = 2018-06-20 09:20:56.939040056 -0400

$ date +%s.%N
1536867622.139983577
```

# Modifier l'horodatage d'un fichier

Commande `touch`

- Met à jour la date de dernière modification
- Optionnellement crée le fichier à vide s'il n'existe pas

Options utiles

- `-d` spécifie la date (par défaut maintenant)
  GNU permissif sur la forme
- `-a` ne change que la date de dernier accès

# Modifier l'horodatage d'un fichier

```
$ touch fichier
$ stat -c %y fichier
2018-07-09 08:39:57.149794939 -0400
$ touch -d 2000-01-02T03:04:05.6666 fichier
$ stat  -c %y fichier
2000-01-02 03:04:05.666600000 -0500
$ touch -d '10 years' fichier
$ stat  -c %y fichier
2028-07-09 08:40:10.677100049 -0400
```

# Liens durs et liens symboliques

# Liens symboliques

Nouveau type de fichier « `l` » (L)

- Contenu : un chemin (relatif ou absolu)
- Même vers un fichier spécial : répertoire, fichier périphérique, un autre lien symbolique
- Même vers un autre système de fichiers

```
$ ls -lh /bin/sh
lrwxrwxrwx 1 root root    4 jan 24 2017 sh -> dash
$ ls -lh /bin/dash
-rwxr-xr-x 1 root root 115K jan 24 2017 dash
```

`readlink` — affiche la valeur d'un lien symbolique (GNU)

```
$ readlink /bin/sh
dash
```

# Utilisation des liens symboliques

- **Ouvrir** un lien symbolique c'est ouvrir le fichier lié
  ouvrir = lire, écrire ou exécuter
- Le système d'exploitation suit les liens **automatiquement**
- Les programmes n'ont pas à les gérer **spécifiquement**
  (mais peuvent s'ils le veulent)

```
$ ls -l
lrwxrwxrwx 1 privat privat   16 lien.txt->les_effarés.txt
-rw-r--r-- 1 privat privat 1155 les_effarés.txt
$ wc lien.txt les_effarés.txt
  51  191 1155 lien.txt
  51  191 1155 les_effarés.txt
 102  382 2310 total
$ echo toto >> lien.txt
$ tail -n 2 les_effarés.txt
Au vent d'hiver.
toto
```

# Création de liens symboliques

Commande `ln -s` (*link –symbolic*)

```
$ ln -s fichier lien
$ ls -il fichier lien
188577 -rw-rw-r-- 1 privat 0 jui  9 08:39 fichier
188786 lrwxrwxrwx 1 privat 7 jui  9 09:31 lien -> fichier
$ echo 'Bonjour' > lien
$ cat fichier
Bonjour
```

# Manipulation de liens symboliques

Un lien symbolique est un fichier autonome

- Numéro d'index (inœud ou *inode*)
- Dates
- Propriétaires
- Pas de droits (toujours 0777)
- Indépendant du fichier lié éventuel

On a donc 2 fichiers

- le fichier lié
- le lien lui-même

# Cibles des opérations : lien ou lié ?

Supprimer (`rm`), déplacer/renommer (`mv`)

- Le lien

Copier (`cp`)

- Le lié par défaut
- `-P` (`--no-dereference`) pour copier le lien

Modifier (`touch`, `chown`, `chgrp`)

- Le lié (par défaut)
- `-h` pour ne pas suivre le lien

Ouvrir, lire, écrire, exécuter

- Le lié, toujours

# Questions

Peut-on **créer** et **utiliser** un lien symbolique

- vers un fichier inexistant ?
- vers un fichier d'un autre utilisateur ?
- vers un fichier sur un chemin qu'on ne peut pas traverser ?
- vers lui-même (boucle de liens symboliques) ?
- vers le répertoire . ?

# Liens durs

Un lien dur (ou direct, ou physique, ou *hard link*) est

- Une entrée **supplémentaire** vers un même fichier
- Dans un **répertoire**
- Qui désigne un **même** fichier (même numéro d'index)

`ln` — crée des liens entre des fichiers

## Piège

Les liens durs ne sont pas des liens

# Liens durs: exemple

```
$ ls -li fichier
1885770 -rw-rw-r-- 1 privat 8 jui  9 09:32 fichier
$ ln fichier liendur
$ ls -li fichier liendur
1885770 -rw-rw-r-- 2 privat 8 jui  9 09:32 fichier
1885770 -rw-rw-r-- 2 privat 8 jui  9 09:32 liendur
```

`ln` incrémente le **nombre des liens durs**

# Liens durs: exemple

```
$ ls -li fichier
1885770 -rw-rw-r-- 1 privat 8 jui  9 09:32 fichier
$ ln fichier liendur
$ ls -li fichier liendur
1885770 -rw-rw-r-- 2 privat 8 jui  9 09:32 fichier
1885770 -rw-rw-r-- 2 privat 8 jui  9 09:32 liendur
```

`ln` incrémente le **nombre des liens durs**

```
$ rm fichier
$ ls -li liendur
1885770 -rw-rw-r-- 1 privat 8 jui  9 09:32 liendur
```

`rm` décrémente le **nombre des liens durs**

Le fichier n'est supprimé que si le compteur atteint 0

# Limitations des liens durs

- Limité à une même partition
- Ne fonctionne pas sur les répertoires (en général)
- Dans le doute utilisez les liens symboliques

## Pourquoi les liens durs alors ?

- Existaient avant les liens symboliques
- Certains usages spécifiques (déduplication)
- Utilisés à l'interne pour gérer les hiérarchies de répertoires (. et ..)

# Chercher et filtrer

# Chercher des lignes

`grep` — cherche les lignes correspondant à un motif

- `-i` ignorer la casse (majuscule/minuscule)
- `-n` affiche le numéro de ligne
- `-v` affiche les lignes qui ne correspondent pas
- `-x` chercher la ligne entière
- `--color` colorie le motif (GNU)

```
$ grep jamb /usr/share/dict/french
croc-en-jambe
[...]
unijambistes
```

`grep` utilise des **expressions régulières** (les détails une autre fois)

# Chercher des fichiers

`find` — rechercher des fichiers dans une hiérarchie de répertoires

```
$ find .
.
./.secret
./hello.txt
./répertoire
./répertoire/document.txt
./lisez-moi.txt

$ find /
/
/sys
/sys/kernel
[...]
^C
```

# Opérandes de recherche

Les opérandes de `find` permettent de contrôler la recherche

- `-name` fichiers avec un nom de base (*glob*)
- `-type f` fichier régulier (`d` pour répertoire)

```
$ find . -name '*.txt'
./lisez-moi.txt
./répertoire/document.txt
./hello.txt

$ find . -name '*oi*' -type d
./répertoire/document.txt
```

## Attention

- Les opérandes commencent par un « - » **simple**
- Échapper **correctement** les opérandes (`shell` vs. `find`)
- Le répertoire de recherche est **avant** les opérandes

# Plus d'opérandes

≈ 20 opérandes POSIX (et plus de 50 chez GNU):

- `-empty` fichiers et répertoires vides
- `-mtime` +**n** fichiers modifiés depuis **plus** de **n** jours
- `-path` chemin des fichiers (incluant les répertoires)
- `-perm` permissions des fichiers
- `-size` -**n**`c` fichiers de **moins** de **n** octets
  GNU permet aussi `k`, `M` et `G` comme multiple de taille. Sans multiple (ni `c`), des blocs de 512o sont considérés
- `-user` fichiers appartenant à un utilisateur

```
$ find . -size +500c -type f
./répertoire/document.txt
./.secret
```

# Faire des actions

`find` peut exécuter des opérations en masse

- `-print` affiche le nom du fichier (par défaut)
- `-printf` affiche avec un format (GNU)
- `-delete` supprime le fichier (GNU)

```
$ find -name "*.txt" -printf "%p: taille=%s date=%AD\n"
./lisez-moi.txt: taille=491 date=09/17/18
./répertoire/document.txt: taille=910 date=09/17/18
./hello.txt: taille=460 date=09/17/18
```

# Actions arbitraires

- `-exec` exécute une commande
- `-ok` demande une confirmation avant d'exécuter

```
$ find -name "*.txt" -print -exec head -n 1 {} \;
./lisez-moi.txt
Bash est un interpréteur de commandes (shell)
./répertoire/document.txt
INF1070 - Utilisation et administration des
./hello.txt
Bonjour
```

## Attention à l'opération `-exec`

- Chaque argument est **séparé** (par des espaces)
- Échapper **correctement** les arguments
- « `{}` » (seul) est remplacé par le **chemin** du fichier
- « `;` » (seul) **termine** le `-exec` mais doit être échappé

# Compression, archivage et sauvegarde

# Compresser des fichiers

**Compresser** = stocker autant de données dans moins d'espace

## Comment c'est possible ?

- Éliminer la redondance dans l'information
- Théorie de l'information, dans le nouveau cours d'algorithmique (INF5130)

# GNU zip

`gzip` compresse et décompresse des fichiers (LSB)

- `-d` (`-decompress`) décompresse
- `-l` (`--list`) affiche les informations sur le fichier compressé
- `-v` (`--verbose`) mode verbeux

`gunzip` équivalent à `gzip -d` (LSB)

```
$ ls -l hello.*
-rw-r--r-- 1 460 jui 13 15:02 hello.txt
$ gzip -v hello.txt
hello.txt:    34.1%
$ ls -l hello.*
-rw-r--r-- 1 331 jui 13 15:02 hello.txt.gz
$ gunzip hello.txt.gz
-rw-r--r-- 1 460 jui 13 15:02 hello.txt
$ ls -l hello.*
```

# Compresser des flux

`gzip` (et `gunzip`) peuvent s'utiliser comme des **filtres**

- `-c` (`--stdout`) écrit sur la sortie standard

`zcat` équivalent à `gunzip -c` (LSB)

```
$ find /usr/share/doc | gzip -vc > list.gz
 87.9%
$ ls -l list.gz
-rw-r--r-- 1 privat privat 288766 sep 25 15:35 list.gz
$ zcat list.gz | wc
  48235   48240 2395728
```

# Question: compresser un fichier compressé ?

Compressons `hello.txt.gz` !

# Question: compresser un fichier compressé ?

Compressons `hello.txt.gz` !

```
$ gzip hello.txt.gz
gzip: hello.txt.gz already has .gz suffix -- unchanged
$ gzip -vkf hello.txt.gz
hello.txt.gz:    -1.5%
$ ls -l hello.txt*
-rw-r--r-- 1 460 sep 24 14:48 hello.txt
-rw-r--r-- 1 331 sep 24 14:48 hello.txt.gz
-rw-r--r-- 1 367 sep 24 14:48 hello.txt.gz.gz
```

La redondance que `gzip` sait éliminer a **déjà** été éliminée

# Compression et formats de fichiers

## De combien compresse-t-on ?

```
$ ls -l debian.png mots.txt
-rw-r--r-- 1 10676 sep 24 13:44 debian.png
-rw-r--r-- 1 10676 sep 24 13:50 mots.txt
```

# Compression et formats de fichiers

## De combien compresse-t-on ?

```
$ ls -l debian.png mots.txt
-rw-r--r-- 1 10676 sep 24 13:44 debian.png
-rw-r--r-- 1 10676 sep 24 13:50 mots.txt

$ gzip -vk debian.png mots.txt
debian.png:    0.6% -- replaced with debian.png.gz
mots.txt:     77.7% -- replaced with mots.txt.gz
$ ls -l debian.png* mots.txt*
-rw-r--r-- 1 10676 sep 24 13:44 debian.png
-rw-r--r-- 1 10639 sep 24 13:44 debian.png.gz
-rw-r--r-- 1 10676 sep 24 13:50 txt.mots
-rw-r--r-- 1  2404 sep 24 13:50 mots.txt.gz
```

- Le format `png` est **déjà** compressé par défaut

$\rightarrow$ Il est **superflu** de le sur-compresser

# Compresser des images

**Image** matricielle (*bitmap*) = grille de **pixels** (*picture elements*)

**BMP (Windows bitmap)**

- Le format par défaut des vieux Windows
- Pas compressé (par défaut)

**PNG (*portable network graphic*)**

- Compression sans perte d'information
- Efficace pour compresser les aplats et les dégradés

# Compresser des images

**JPEG (*joint photographic experts group*)**

- Compression avec perte d'information
- Efficace pour compresser les photos

**SVG (*scalable vector graphics*)**

- Format vectoriel (pas matriciel)
- Sous forme textuelle (XML)

# Outils et formats de compression

Différents outils, formats et algorithmes de compression

- `gzip` (`.gz`) → algo DEFLATE (aussi utilisé par `zip` et `png`)
- `bzip2` (`.bz2`) → algo BWT
- `xz` (`.xz`) → algo LZMA (aussi utilisé par `7z`)

Wikipédia liste plus de 60 formats

## L'art du compromis

- Taux de compression
- Compression: vitesse, mémoire, etc.
- Décompression: vitesse, mémoire, etc.

# Archivage

**Archiver** regrouper fichiers et répertoires dans un seul gros fichier

## Pourquoi ?

- Sauvegarder les données
- Simplifier la distribution d'un tas de fichiers
- Pouvoir compresser ensemble un tas de fichiers

## Difficultés

- Gérer les types de fichiers (liens durs et symboliques)
- Stocker les métadonnées (propriétaires, protection, dates, etc.)
- Déterminer les fichiers à archiver

$\rightarrow$ Tous les problèmes de `find` et des systèmes de fichiers

# Archiver avec tar

`tar` archive des fichiers (LSB)

## Modes

- `-c` créer une nouvelle archive
- `-x` extraire des fichiers
- `-r` ajoute des fichiers à une archive
- `-t` liste les fichiers

## Options

- `-f` travailler avec un fichier (et non l'entrée et sortie standard)
- `-v` verbeux

```
$ tar -cf base.tar base
$ ls base.tar
-rw-r--r-- 1 privat privat 10240 sep 21 15:34 base.tar
$ tar -xf base.tar
```

# Tarballs

Autres options de `tar`

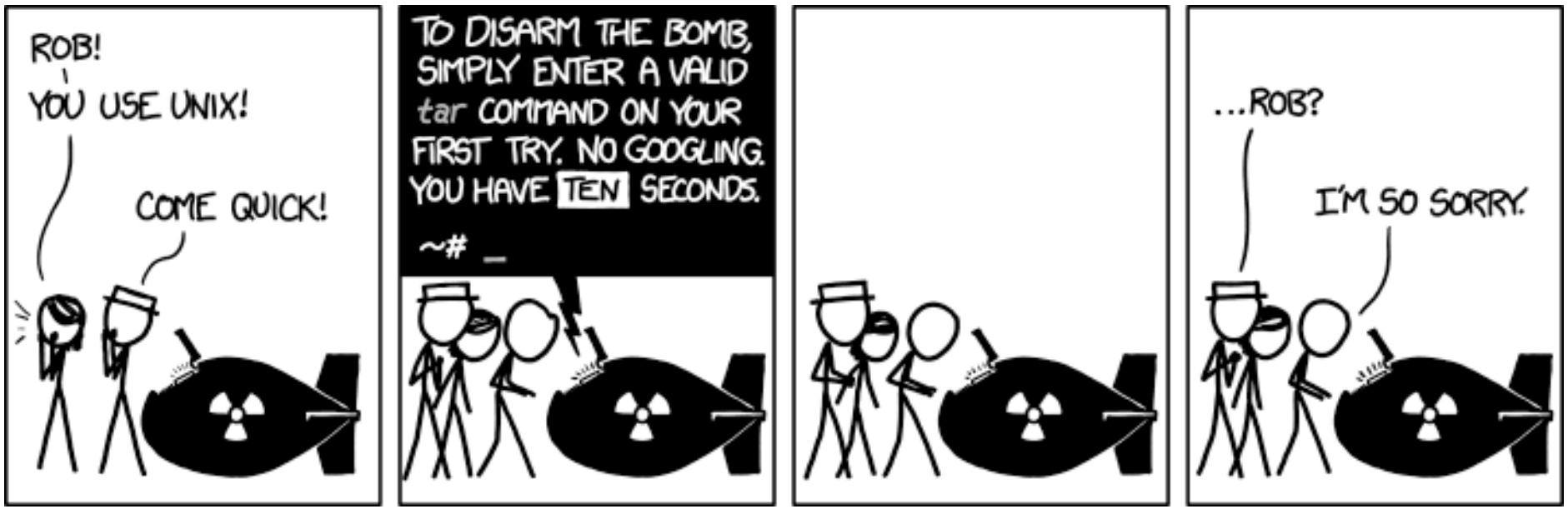
- `-z` (dé)compresse avec gzip (`.tar.gz` ou `.tgz`)
- `-j` (dé)compresse avec bzip2 (`.tar.bz2`)
- `-J` (dé)compresse avec xz (`.tar.xz`)

```
$ wget https://cdn.kernel.org/pub/\
> linux/kernel/v4.x/linux-4.18.9.tar.xz

$ xz -l linux-4.18.9.tar.xz
Compressé  Décompressé  Ratio  Nom de fichier
 97,1 MiB    791,6 MiB  0,123  linux-4.18.9.tar.xz

$ tar -tJf linux-4.18.9.tar.xz | wc -l
65053

$ tar -xJf linux-4.18.9.tar.xz
```

# Mythologie de tar

- Un vieil outil: 1979 (*tape archive*)
- Pas standardisé (tentative POSIX avortée)
- Variations d'implémentation (GNU, BSD, autres UNIX®)
- Formats incompatibles (attributs de fichiers étendus)
- Beaucoup de fonctionnalités
- Beaucoup d'options (plus de 100 chez GNU)

# Mythologie de tar: XKCD




Source: https://xkcd.com/1168/ (2013)

# Autres outils d'archivage

Des outils qui archivent et compressent

- `zip` (winzip)
- `7z` (7zip)
- `rar` (winrar)

Origine windowsienne:

- Gestions insuffisante des noms, droits et types de fichiers Unix
- Algorithmes parfois propriétaires
- Incapables de compresser des flux (entrée/sortie standard)

À éviter dans les environnements Unix

# Besoin de fiabilité

## Rappel

- Les données sont la partie essentielle d'un système informatique
- « Système d'information » : la valeur est dans les informations, pas dans le système

## Administrateur système

Une de ses tâches principales

- Garantir que les données ne sont ni perdues ni corrompues

## Données et fichiers

- Dans quelles conditions peut-il y avoir perte ou corruption ?
- Quelles sont les autres tâches d'un administrateur système ?

# Sauvegarde (backup)

- Dupliquer les données
- Conserver un historique
- Stocker le double ailleurs

## Deux approches

- Sauvegarde physique vs. sauvegarde logique

# Outils de sauvegarde

Très nombreux outils qui varient surtout

- Les fonctionnalités
- Les performances
- La simplicité
- La configurabilité

## Exemples

- déjà-dup (graphique) système très simple mais qui fait la bonne chose (régulier, distant, chiffré) par défaut chez Ubuntu. Frontale à `duplicity`
- bacula suite complète pour gros et moyens systèmes
- duplicity (console)
- rdiff-backup (console) simple

# Sauvegardes plus manuelles

- Unison synchronise des fichiers entre machines.
  Attention: pas d'historique
- Git système de contrôle de version distribué.
  Attention: pas toujours adapté, pas de nettoyage d'historique
- `rsync` copie de fichiers efficace, distante et versatile
- `tar` archiveur standard relativement portable
- `dd` convertit et copie (POSIX). S'utilise surtout pour copier des disques et partitions.

# Fichiers spéciaux, partitions et systèmes de fichiers

# Fichiers périphériques

- Fichiers spéciaux
- Traditionnellement dans `/dev` (*device*)

Deux types

- caractères (c) : envoie et/ou reçoit des séquences d'octets
- blocs (b) : écrit et/ou lit dans un bloc d'octets

Un peu bizarres

- Pas de taille mais deux numéros: majeur, mineur

```
$ ls -l /dev/sda1 /dev/null
crw-rw-rw- 1 root root 1, 3 sep 14 15:11 /dev/null
brw-rw---- 1 root disk 8, 1 sep 14 15:11 /dev/sda1
```

# Périphériques virtuels

- `/dev/null` : la poubelle (POSIX)
- `/dev/zero` : une infinité d'octets nuls (extra)
- `/dev/urandom` : générateur aléatoire (extra)

```
$ cat /dev/urandom | tr -cd a-z | head -c 20
yvpbfowxckmqbmvrouvc
```

# Terminaux

- `/dev/tty3` : un terminal (*teletypewriter*) (extra)
- `/dev/pts/2` : un pseudo-terminal esclave (*slave*) (extra)

```
$ echo "hello" > /dev/pts/3
```

- `tty` affiche le terminal courant (POSIX)
- `/dev/tty` alias vers le terminal courant (POSIX)

# Périphériques caractères

Tout ça dépend du **modèle** et du **pilote** du périphérique

- `/dev/audio1` : une carte son (extra)

```
$ cat linux.au > /dev/audio1
```

- `/dev/input/mice` : l'ensemble des souris (extra)

```
$ xxd /dev/input/mice
```

# Disques et périphériques bloc

- `/dev/sda` : le premier disque dur (*SCSI drive*) (extra)
- `/dev/sda1` : la première partition de `sda` (extra)

```
$ /sbin/fdisk -l /dev/sda
```

# Stockage indépendant de données

- Un système de fichiers n'a pas à connaître les autres
- Le système d'exploitation offre une vue **unique** et **cohérente**
- La racine de la hiérarchie unique est « / »

## Windows

- Historiquement (DOS): chacun est associé a une lettre
  A:, C:, etc.
- Windows modernes: l'approche Unix est également possible

**Question** Quel est l'avantage d'un système sur l'autre?

# Type de système de fichiers

- ext2, ext3, ext4: les types natifs de Linux
- NTFS: celui de Windows
- HFS+: celui de macOS
- FAT32: celui de Windows95 (fréquent sur les clés USB)

## Les différences ?

- Chaque système d'exploitation est plus ou moins capable de lire les types des autres
- Chaque type de système de fichier a plus ou moins de fonctionnalités

$\rightarrow$ Wikipédia: Comparison of file systems

# Montage

`mount` monte un système de fichiers (LSB)

- Un système de fichier est monté sur un répertoire
- Ce répertoire est le **point de montage**
- Seul l'administrateur peut monter un système de fichiers

`umount` démonte un système de fichiers (LSB)

`findmnt` affiche la hiérarchie des systèmes de fichiers (extra)

`/etc/fstab` informations statiques sur les systèmes de fichiers

# Montage automatique

Dans les environnements Linux graphiques

- Les périphériques amovible (clés USB, DVD, etc.)
- Sont montés automatiquement
- Exemple: `/media/privat/usb1/`

L'utilisateur doit **démonter** le périphérique avant de le retirer

- Avec l'interface graphique (« *éjecter* » ou « *démonter* »)
- `udisksctl` outil en ligne de commande analogue à l'interface graphique (extra)

# Démontage

Quel est le risque de retirer un périphérique sans le démonter ?

# Démontage

Quel est le risque de retirer un périphérique sans le démonter ?

- Des pertes de données

Pour des raisons de **performances**

- Les écritures sur disques passent par des **tampons** (*buffers*)
- Les tampons sont vidés régulièrement

`sync` force le vidage des tampons du système de fichiers (LSB)

# Pseudosystèmes de fichiers

Des fichiers qui ne sont pas en vrai sur un disque

## tmpfs – système de fichiers temporaires (extra)

Les fichiers sont en mémoire

- Rapide d'accès
- Mais pas de persistance
- Limite en taille

$\rightarrow$ Pratique pour la communication inter-processus

## proc et sysfs (extra) – interfaces avec le noyau

```
$ cat /proc/version
$ cat /proc/cpuinfo
$ cat /proc/meminfo
```

Beaucoup d'entrées avancées et spécifiques

# Quoi monter ?

Un système de fichier régulier (non pseudo-) réside sur un périphérique de type bloc

- `lsblk` affiche tous les périphériques blocs (extra)
- `blkid` affiche les attributs des périphériques blocs

Exemples de périphérique de type bloc

- Les disques
- Les partitions

# Partitions et formatage

- Partitionner: découper un périphérique en morceaux
- Formater: créer un système de fichiers vide

## Quand partitionner et formater?

- Lors de l'installation d'un OS
- Lors de l'ajout d'un nouveau disque

## Risque

- Perte de données
- Même sur le disque d'à coté
- Due à une erreur humaine

## Mitigation

- Sauvegarder (faire des *backups*)

# Outils de partition et formatage

La gestion du stockage est propre à chaque système

Aucun des outils n'est POSIX ni mème LSB

- `gparted` et `gnome-disks` éditeurs de partition graphique
- `parted`, `fdisk` et `cfdisk` éditeurs de partition console
- `mkfs` crée des systèmes de fichiers

# Partition d'échange (*swap*)

Simuler et étendre de la mémoire (*swap*)

- Une partition entière
- Ou un fichier sur une partition

`free` affiche la mémoire libre et utilisée

## Hibernation

Les partitions d'échange servent aussi à l'hibernation (*suspend to disk*)

- L'état de la mémoire est **sauvegardé** dans la swap
- L'ordinateur s'**éteint** (la mémoire est perdue)
- À l'allumage, la swap est **chargée** en mémoire

$\rightarrow$ L'ordinateur revient à un état **identique**

# Aller plus loin

La bonne gestion des données est fondamentale

Il n'y a pas de limite aux besoins et à la complexité des solutions

- Système de fichier réseau: NFS
- Volumes logiques: LVM
- Redondance: RAID
- Chiffrement: LUKS
- Sauvegardes instantanés (*snapshots*)

## Systèmes de fichiers **de pointe**

- Offrent le passage à l'échelle
- Intègre les fonctionnalités avancés
- ZFS (Sun/Oracle) et Btrfs (Linux)