

# Hébergement

*survivre dans les nuages*



# État des lieux : Les O.S.

Actuellement on a deux types de **système** d'exploitation (**O**perating **S**ystem) :

- Windows (~80 % des PC)
- Unix(s) : Linux (android), MacOS X (IOS) ...

90 % des **serveurs** tournent sous un système de type **unix**, généralement **Linux**.

# État des lieux : les Clouds\*

Les cloud sont des infrastructures de **serveurs** constitué de machines **virtuelles**.

Ces machines n'ont **pas** d'interface **graphique** utilisateur.

Elle sont accessible et administrable en **ligne de commande**.

\* : par nature cette notion est nébuleuse

# Système de Fichier

Les supports de **stockage** sont gérés par un Système de **fichiers** (File system)

Il en existe de nombreux\*, mais ils ont **tous** en commun une structure **arborescente**

Elle est constitué de

- **nœuds** : les **répertoires** alias **dossiers**
- **feuilles** : les **fichiers**

\*:NTFS, ext(2,3,4), HFS+, BTRFS, VFAT ...

# Système de Fichier

Sous **Windows** il y a une arborescence par support physique (C:\, D:\ ...)

Sous **Unix** il y a une arborescence unique dans laquelle se « branche » (**mount**) les supports physiques dans un dossier appelé « point de montage » qui peut être n'importe où dans l'arbre.

# Système de Fichier : Chemin

Pour retrouver un fichier il faut connaître son « chemin » :

Listes des répertoires constituant la branche ou se trouve le fichier :

- /home/jacquotp
- C:\Utilisateurs\jacquotp

Les répertoires sont séparés par un symbole :

- « / » sous unix, dans les fichiers html ...
- « \ » sous windows

# Système de Fichier :

## Chemin absolu

Un chemin est dit « absolu » quand il commence à la racine de l'arbre :

- « / » : pour unix
- « X:\ » pour windows

Un chemin absolu identifie de façon unique un fichier.

C'est le véritable nom du fichier.

# Système de Fichier : Chemin Relatif

Un chemin est dit « relatif » quand il ne commence pas à la racine de l'arbre :

- BUT/R115/CM2.pdf

Il se construit par rapport à un point de départ qui est variable.

Pour faciliter l'utilisation des chemins relatifs on a défini deux répertoires particuliers :

- « . » : désigne l'endroit où on se trouve (= ici)
- « .. » : désigne le répertoire parent (plus près de la racine d'un cran)



# Système de Fichier : Chemin Relatif

si le fichier « index.html » se trouve  
dans `/web/jacquotp/demo`

Alors la balise

```
<link rel="stylesheet" href=" ../css/monstyle.css">
```

Fait référence au fichier :

- `/web/jacquotp/css/monstyle.css`

«  `../css/monstyle.css`  » est un chemin  
relatif à la position de index.html

«  `/web/jacquotp/css/monstyle.css`  » est le  
chemin absolu correspondant

# Système de Fichier : Absolu ou Relatif ?

Dans les **fichiers** d'un site **web** il est **impératif** d'utiliser des chemins **relatifs** car le répertoire de base **change** d'un **hébergement** à l'autre.

En ligne de commande :

- Les chemins **relatifs** sont souvent plus **courts** mais la commande devient fausse si vous changer de point de départ
- On peut mélanger les deux types de chemin suivant les besoins

# Système de Fichier : les Droits

Les systèmes d'exploitation actuels  
sont tous **multi-utilisateurs**.

Ils disposent d'un mécanisme de  
gestion des accès au fichiers :  
les **droits (modes)** d'accès.

# Système de Fichier : les Droits

Les systèmes d'exploitation  
reconnaissent deux types d'entité :

- Les utilisateurs (users)
- Les groupes d'utilisateurs (groups)

Les droits sont définis par rapport à  
ces deux types d'entité.

# Système de Fichier : les Droits

Les unix utilisent une gestion des droits minimaliste qui se base sur :

3 niveaux d'entité :

- Le propriétaire (**u**ser) **u**
- Le groupe (**g**roup) **g**
- Les autres (**o**ther) **o**

3 droits pour chaque niveau

- Lecture (**r**ead) **r**
- Écriture (**w**rite) **w**
- Exécution (**e**xécute) **x**

# Système de Fichier : les Droits

La notion de **propriétaire** implique le droit de **changer** les droits.

Un fichier appartient à un utilisateur et un groupe

droit	Fichier	Répertoire
R	Lire le contenu du fichier	Lire le répertoire i.e lister les fichiers
W	Modifier le contenu du fichier	Modifier le répertoire i.e. : Ajouter/supprimer des fichiers
X	Exécuter (programme)	Traverser (cross = croix=X)

# Système de Fichier : les Droits

Notation numérique **octal** :

À chaque droit on associe une valeur :

droit	binaire	octal
<b>r</b>	100	<b>4</b>
<b>w</b>	010	<b>2</b>
<b>x</b>	001	<b>1</b>

- La somme de ces valeurs donne un résultat compris entre 0 et 7 (chiffres de la base **8**)

**rw-** →  $4 + 2 + 0 = 6$       **-w-** →  $0 + 2 + 0 = 2$

**R-x** →  $4 + 0 + 1 = 5$       **rwX** →  $4 + 2 + 1 = 7$

# Système de Fichier : les Droits

Notation numérique **octal** :

Ainsi les droits sont souvent exprimés  
sous la forme d'un nombre à **trois**  
chiffres en base **8** :

- **600** → **u**=rw-    **g**=---    **o**=---
- **640** → **u**=rw-    **g**=r--    **o**=---
- **755** → **u**=rwx    **g**=r-x    **o**=r-x
- **666** → **u**=rw-    **g**=rw-    **o**=rw-
- **777** → **u**=rwx    **g**=rwx    **o**=rwx



# Système de Fichier : les Droits étendus : ACL

Le système de base manque de souplesse  
on peut le compléter avec des  
Access Control List

Elles permettent d'ajouter :

- Des droits pour un/des utilisateurs particuliers
- Des droits pour un/des groupes particuliers
- Des droits par défaut pour les nouveaux fichiers ou répertoires créés.

# Interpréteur de Commande

Interface Utilisateur **fondamentale**

Nommé **shell** (coquille) car se trouve  
« autour du **noyau** (kernel) »

En développement continue depuis  
plus de **50 ans**

Beaucoup de « versions » différentes :

- sh, **bash**(ash, dash), csh, tcsh, ksh,  
zsh, fish, PowerShell

# Interpréteur de Commande

## Syntaxe

Une **commande** se compose de différents éléments **séparés** par des **espaces**

- Le **programme** ou **action** toujours en **premier**
- Les **options** (introduites par **-** ou **--**)
- Les **arguments** (noms de fichiers ou autres)
- **Redirections** (vers fichiers ou autres commandes) toujours en **dernier**

# Interpréteur de Commande

## Syntaxe de syntaxe

La documentation\* utilise une codification pour décrire les commandes :

- `[]` indique des éléments **facultatifs**
- `...` indique une **répétition** ( 1 ou plusieurs)
- `|` ou `{ | }` pour une **alternative** stricte (pas les deux)

ces symboles **ne sont pas** à mettre dans la commande

# Interpréteur de Commande

## Syntaxe de syntaxe

`rm [option...] file...`

- Cette commande peut avoir **0** ou plusieurs **options** et **1** ou plusieurs noms de fichiers
- Exemples :
  - `rm toto`
  - `rm -r -f toto tata titi`
  - `rm -rfv truc`
- Mais pas :
  - `rm -rfv`

\* : **R**ead **T**he **F**ucking **M**anual !

# Interpréteur de Commande

## Syntaxe

Décodez :

```
setfacl [-bkndRLPvh] [{-m|-x}  
acl_spec] [{-M|-X} acl_file] file ...
```

—

# Interpréteur de Commande aide

Les **man** pages

Les unix disposent d'un manuel intégré

Il décrit de façon détaillé les commandes  
usuelles

Usage de base :

man commande

Exemple : man man

Il y a souvent des exemples utiles à la fin  
du manuel.

# Interpréteur de Commande navigation

Pour utiliser les chemins relatifs il faut se positionner dans l'arborescence :

Commande :

**cd** répertoire

(change directory)

Pour savoir où on est :

**pwd**

(print working directory)

Remarque : sous **windows** **cd** sans répertoire fait la même chose que **pwd**, alors que sous **linux** cela vous positionne dans votre répertoire personnel.