Introduction aux Systèmes d'Exploitation

Notion d'un Système d'Exploitation

B. Jacob

IC2/LIUM

11 Septembre 2017

Plan

Plan

Définition

Litéralement :
SE = Système d'Exploitation =
OS = Operating System

Les plus connus :

- Windows (dernière version : *Windows NT 10*)
- Unix / Linux (distributions bureau : *Ubuntu*, *Fedora*, . . .)
- Mac OS (dernière version : *macOS Sierra*)

Définition pratique

C'est quoi ? Un ensemble de programmes qui dirigent l'utilisation des ressources d'un ordinateur.

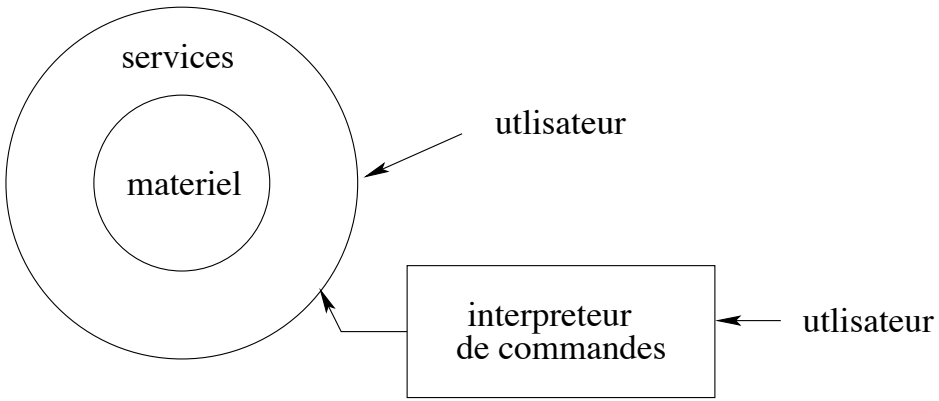
Ca sert à quoi ? A rendre des services à l'utilisateur

- facilite et simplifie l'utilisation d'un ordinateur
- interface entre le matériel et l'utilisateur
- le SE affranchit l'utilisateur des spécificités d'accès au matériel par des ensembles de services.

Définition pratique

C'est fait comment ?

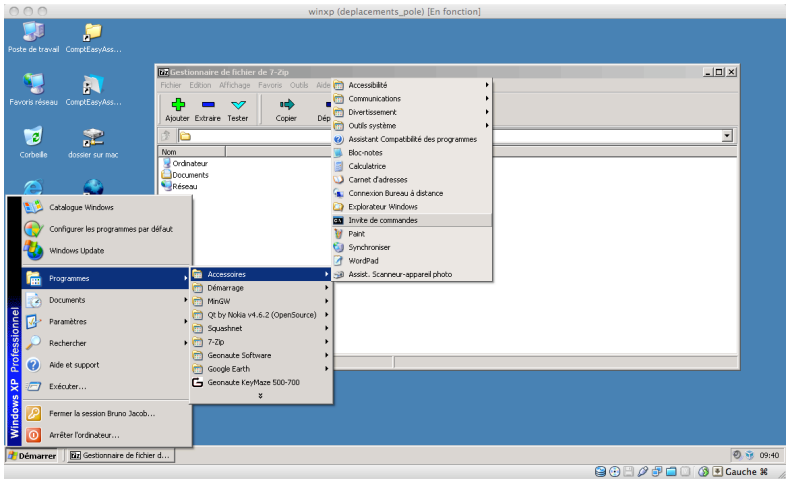
En pelures d'oignon



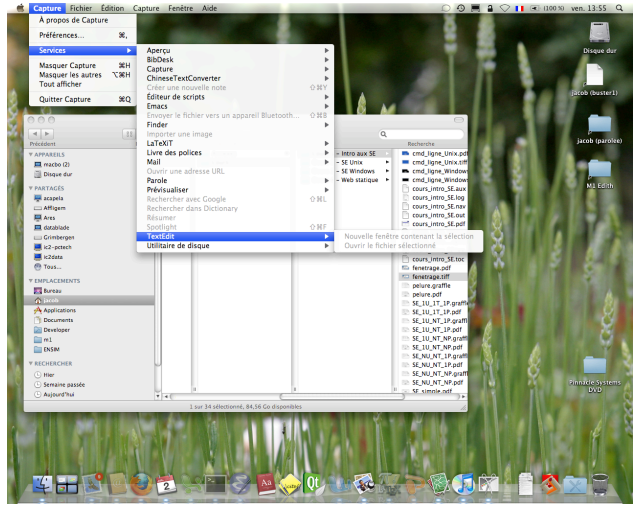
Analogie : restaurant

Graphismes

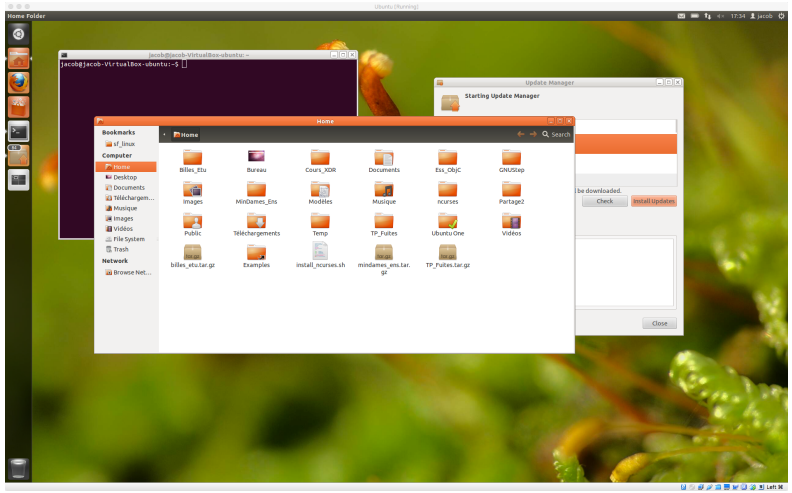
Windows



MacOs

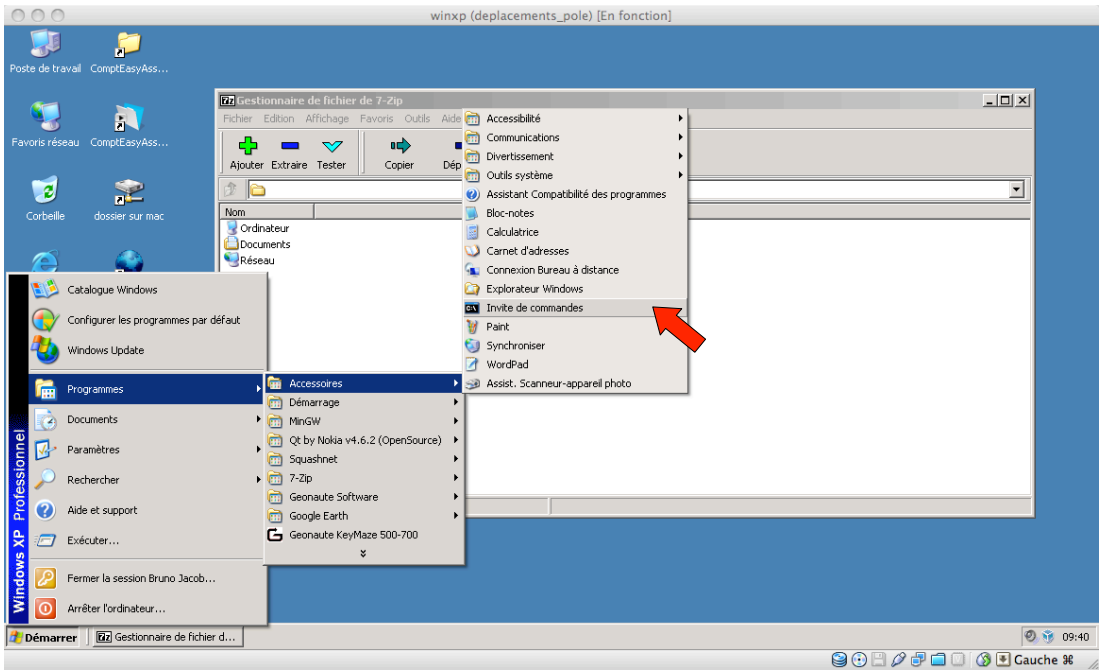


Linux



Interaction avec le SE

En général en cliquant dans des fenêtres



Problème fenêtrage

Inconvénients de demander un service (ou exécuter une commande) par la méthode souri/fenêtre

- il doit y avoir un clic par commande possible
  $\rightarrow$ Si nb commandes $\nearrow$ alors nb de fenêtres/boutons $\nearrow$
- pas personnalisable
  $\rightarrow$ le bouton qui fait la commande que vous voulez existe t il ?
- pas facilement automatisable
  $\rightarrow$ on ne peut pas programmer facilement le clic de la souri dans une fenêtre

Principe Commandes en Ligne

Pas de clic dans une fenêtre

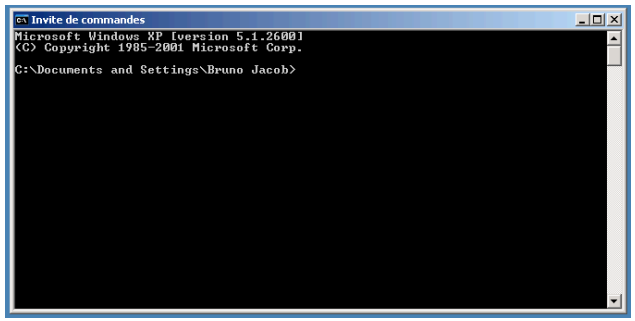
Exécution d'une commande

On écrit le nom de ma commande à exécuter dans un environnement capable de l'interpréter (un shell)

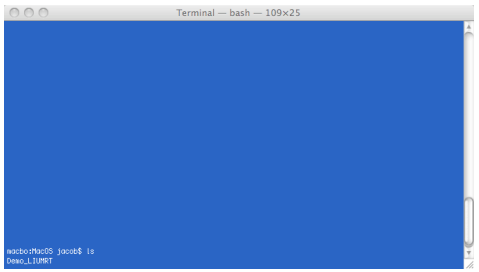
Principe Commandes en Ligne

Cet environnement se présente sous la forme d'une fenêtre "spéciale"

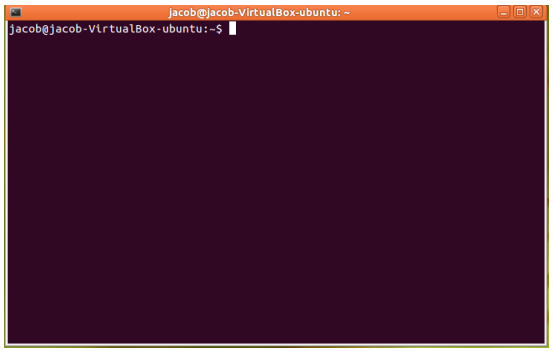
Fenêtre "Invite de cmd" Windows



Fenêtre "Terminal" Mac



Fenêtre "Terminal" Linux



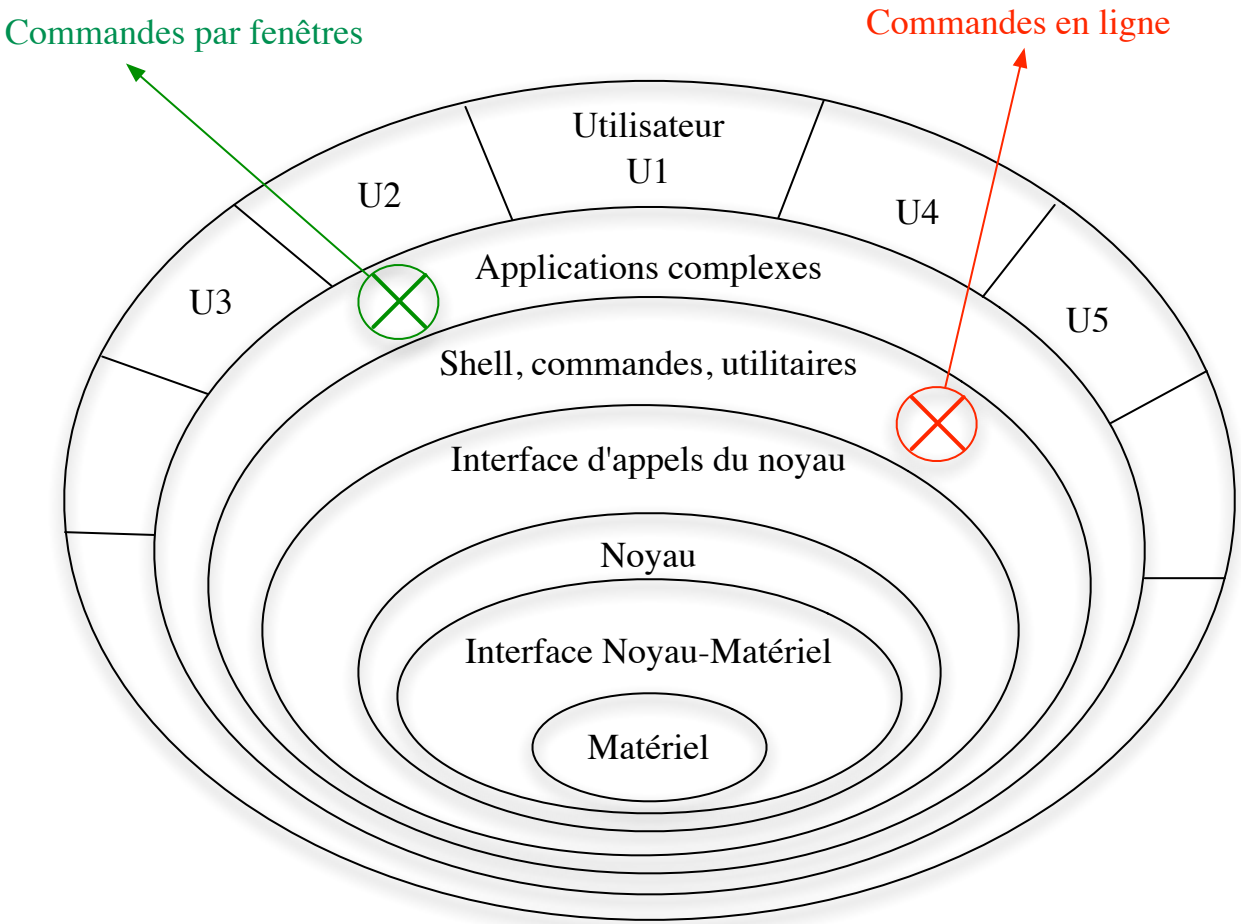
Intérêts

Pourquoi exécuter les commandes en tapant leur nom plutôt que par fenêtrage ?

- on peut réutiliser les noms des commandes dans un programme/script
- nb opérations possibles en ligne $\gg$ nb opérations possibles par fenêtrage
- utilisation en général plus simple
  (peu de commandes à mémoriser)

Intérêts

Les commandes en lignes sont de plus bas niveau que les fenêtres dans les pelures d'oignon :



But du cours

$\Rightarrow$ Utilisation des commandes en ligne

Remarques :

- Commandes en ligne peu pratiques ou peu utilisées sous Windows
- Objets les + utilisés : fichiers

$\Rightarrow$ Contenu première partie (S1) de ce cours :
Commandes en ligne sur la gestion des fichiers sous Linux/MacOs :

Suite :

- Organisation des fichiers sous Linux
- Commandes de gestion des fichiers

Plan

Plan

Définition

- Un fichier informatique = collection d'informations numériques (données)
- les fichiers sont gérés par le SGF (Système de Gestion de Fichiers)
- le SGF identifie un fichier par son nom (unique)
- les données sont stockées de manière permanente sur un support (disque dur, CD...)

Type de fichier

On peut parler de fichier

- texte : fichiers, qui comme on l'aura deviné, contiennent du texte.
- exécutable : que l'on peut exécuter ou lancer, autrement dit des programmes.
- compressé : archives, fichiers Zip, Sit, Rar et autres formats compactés.
- graphique, audio, vidéo : qui contiennent des images, des sons codés en différents formats.
- de données : tout autre type de fichiers.

Identification des fichiers

Avec un système de fenêtrage / icônes

Identification des fichiers

Avec les commandes en ligne = avec les noms de fichiers

Nom d'un fichier

préfixe.suffixe ou *base.extension*

- Base/Préfixe = Nom
- Extension/Suffixe = Type

Exemples types fichiers :

- .txt fichiers texte
- .sh scripts shell
- .c, .h programmes C

pas d'extension → programme exécutable

Commandes sur les fichiers

- `basename` – le nom sans le préfixe
- `dirname` – le préfixe
- `file` → type du fichier
- `cat` → affiche le contenu sans pause
- `head` → premières lignes
- `tail` → dernières lignes
- `touch`
  - si le fichier $\exists$: maj date dernière modif
  - si le fichier $\nexists$, créer un fichier vide
- `more, less` → affiche le contenu avec pause

Droits d'accès aux fichiers

Il faudrait définir autant de droits d'accès qu'il y a de

- façons d'utiliser un fichier (lecture, écriture, modificiations. . .)
- d'utilisateurs (souvent des centaines)

$\Rightarrow$ impossible, nombre de combinaisons trop grand
Définition "arbitraire" de

- 3 façons d'utilisation (appelés droits)
  - droit de lecture (read)
  - droit d'ecriture (write)
  - droit d'exécution (ou de traverser un répertoire)
- 3 classes d'utilisateurs :
  - le propriétaire du fichier
  - le groupe auquel $\in$ le propriétaire
  - tous les autres

Affichage des droits

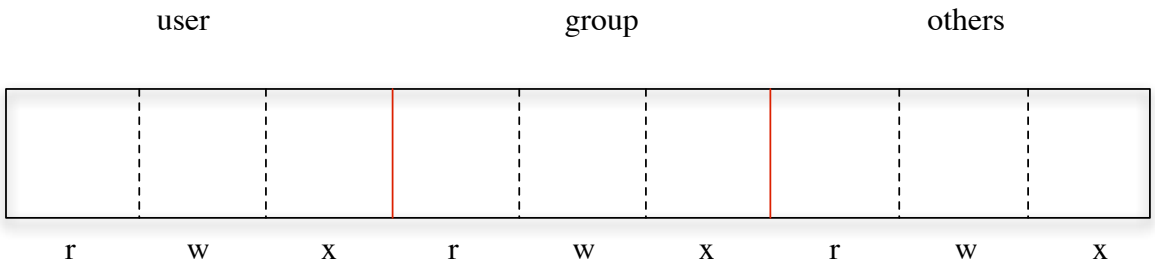
Avec la commande `ls -l`

Affichés sur 10 bits : `-rwxwxrwx`

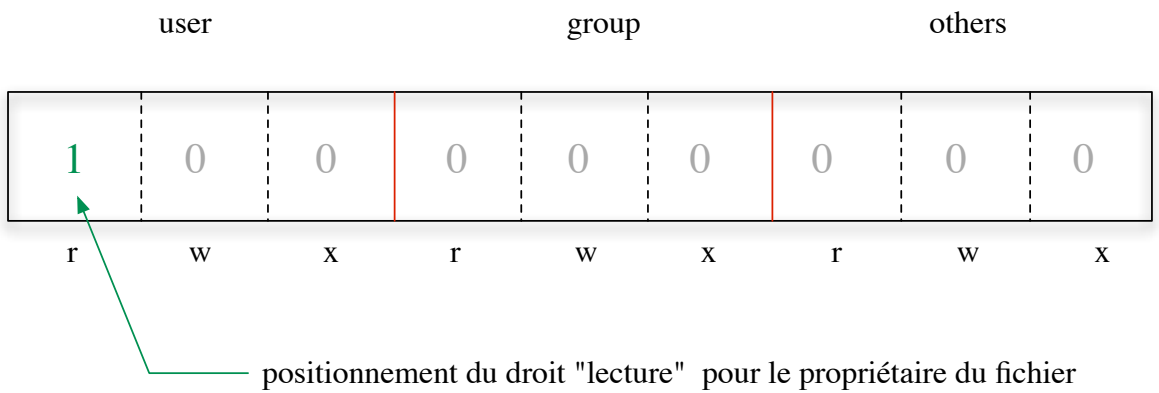
- 1 : Type du fichier
  - `-` : ordinaire
  - `d` : répertoire (directory)
- 2 à 10 : Droits d'accès / d'utilisation
  - 3 droits
    - `r` (Read)
    - `w` (Write)
    - `x` (eXecution)
  - pour les 3 classes d'utilisateurs :
    - `u` (User)
    - `g` (Group)
    - `o` (Others)

$\Rightarrow 3 \times 3$ combinaisons possibles

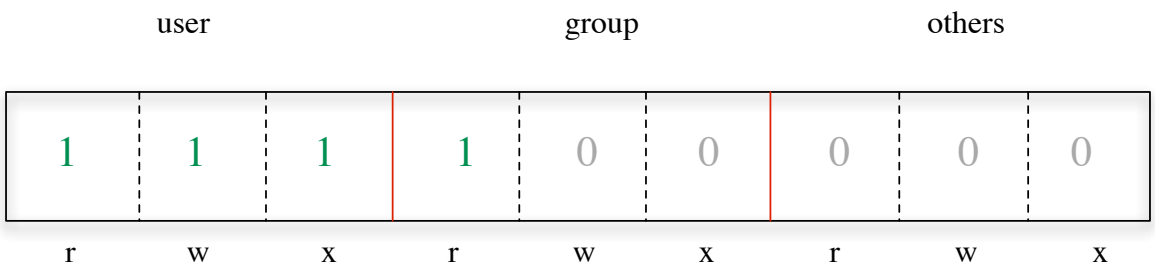
Exemples droits de fichiers



Exemples droits de fichiers



Exemples droits de fichiers



Explications : Fichier que

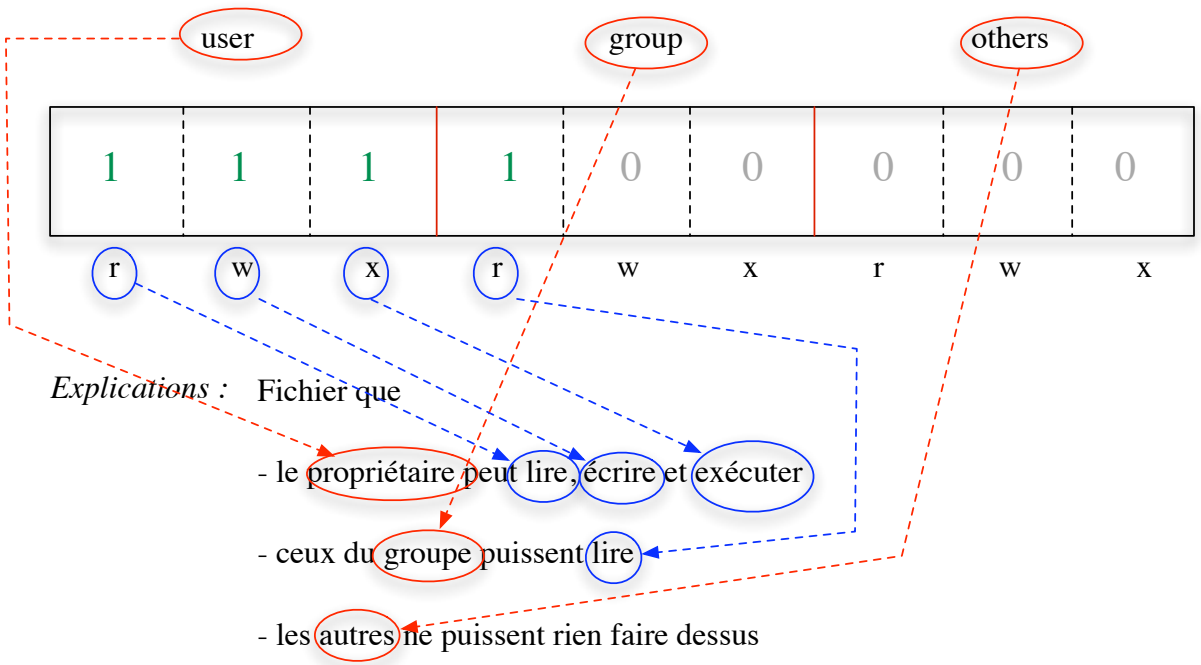
- le propriétaire peut lire, écrire et exécuter

- ceux du groupe puissent lire

- les autres ne puissent rien faire dessus

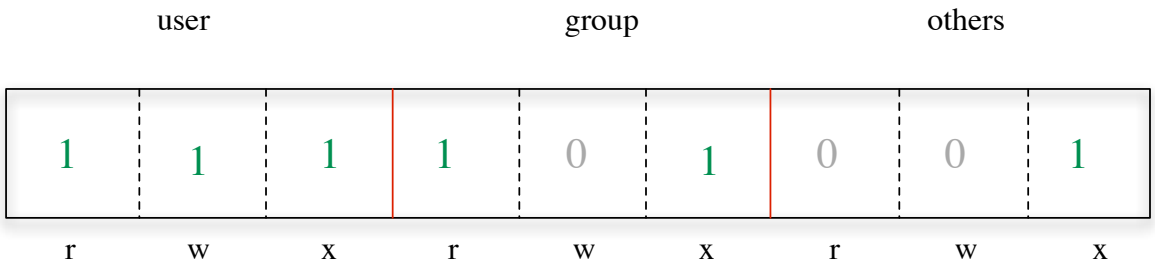
Affichage : `-rwxr-----`

Exemples droits de fichiers



Affichage : `-rwxr-----`

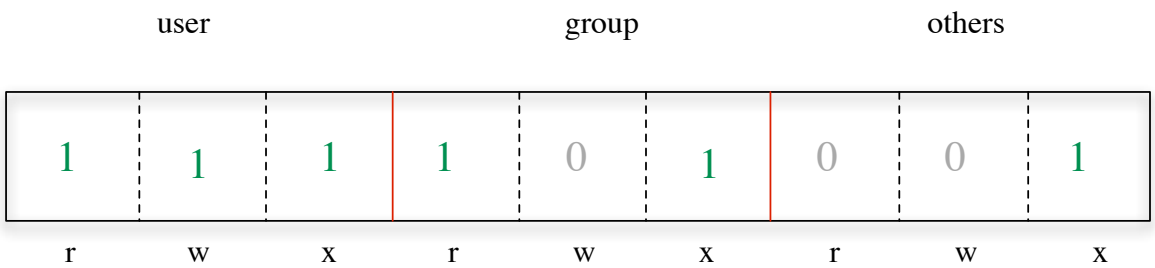
Exemples droits de fichiers



Affichage :

Explications :

Exemples droits de fichiers



Affichage : `-rwxr-x--x`

Explications :

Changement droits d'un fichier

Commande `chmod`

Utilisation 2 façons

- `chmod [ugoa] [+-=] [rwx] filename`
- `chmod` *`OctalMode`* `filename`

Changement droits d'un fichier

`chmod [ugoa] [+-=] [rwx] filename`

Classe

- `a` appliqué à tous (défaut)
- `u` appliqué au propriétaire (user)
- `g` appliqué au groupe
- `o` appliqué aux autres (others)

Opérations

- `+` ajout de droits
- `-` retrait de droits
- `=` affectation de droits

Droits

- `r` droit de lecture (read)
- `w` droit d'ecriture (write)
- `x` droit d'exécution (ou de traverser un répertoire)

Changement droits d'un fichier

`chmod` *`OctalMode`* `filename`

On affecte directement le mode souhaité en octal (base 8)
Exemples :

- 644 : Lecture/écriture pour le propriétaire, lecture seulement pour le groupe et les autres

user	group	other
rw	r	r
110	100	100
6	4	4

- 766 : `rwx` user , `rw` group , `rw` other
- 740 : `rwx` user , `r` group , aucun droit other

Exemple

```
$ ls -l toto.sh
-rw--w---- 1 jacob enseign 433 sep 20 15:28 toto.sh

$ chmod a+x,g-w+r toto.sh
$ ls -l toto.sh
-rwxr-x--x 1 jacob enseign 433 sep 20 15:30 toto.sh
```

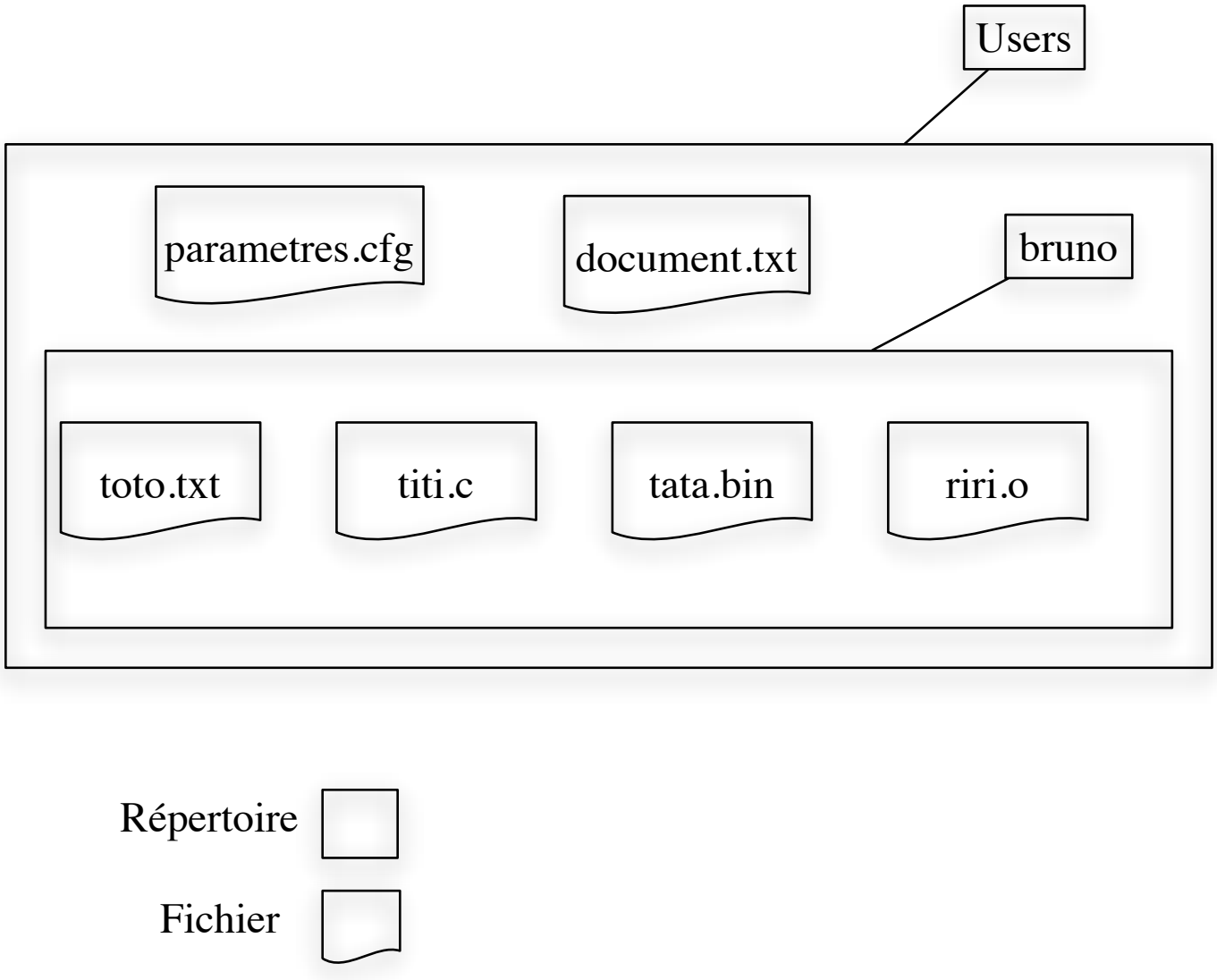
- Fichier exécutable par tous
- Avec les droits du groupe enseign
- Lisible par le groupe
- Modifiable par le propriétaire

Plan

Définition

- répertoire = directory
- En informatique, un répertoire = liste de fichiers.
- le SGF gère un répertoire comme un fichier
  $\rightarrow$ 1 répertoire = liste de fichiers + répertoires
- gestion des droits d'un répertoire = idem que fichier
- manipulation des répertoires = idem que fichiers.

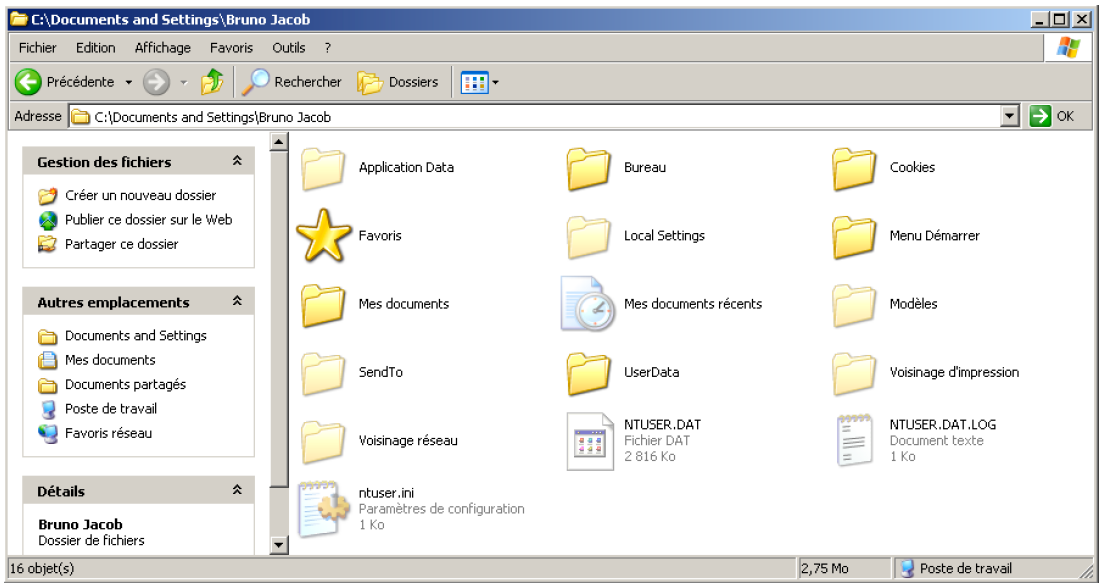
Exemple de répertoire



Visualisation d'un répertoire

Sous Windows répertoire $\approx$ dossier 

Affichage d'un répertoire $\rightarrow$ clic sur une icône qui affiche une fenêtre



Visualisation d'un répertoire

Avec commande en ligne Linux → commande `ls`

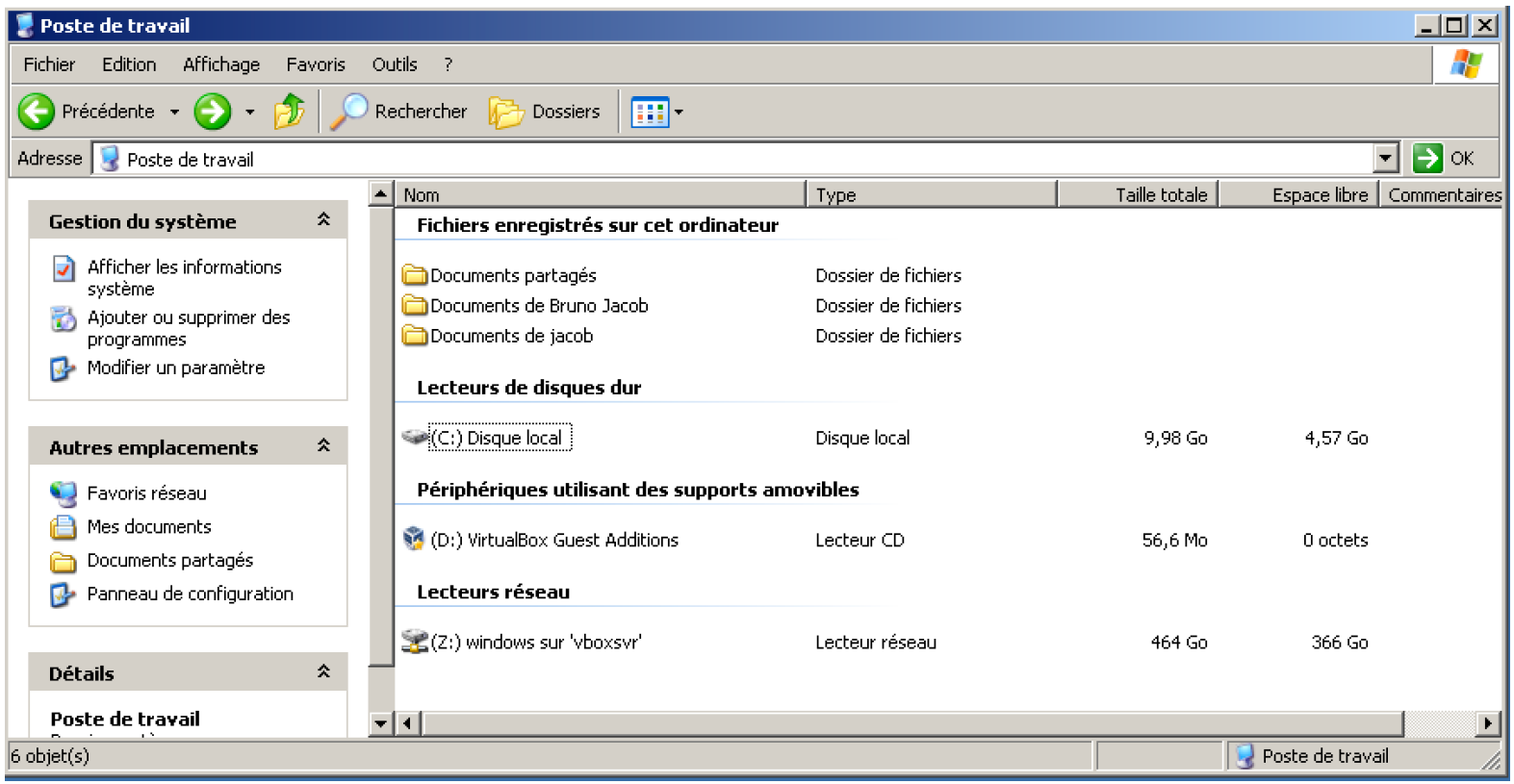
Tapez la commande `ls`
produit l'affichage à l'écran de son contenu

```
Application Data
Bureau
Cookies
Favoris
Local Settings
MenuDémarrer
Mes documents ...
```

Plan

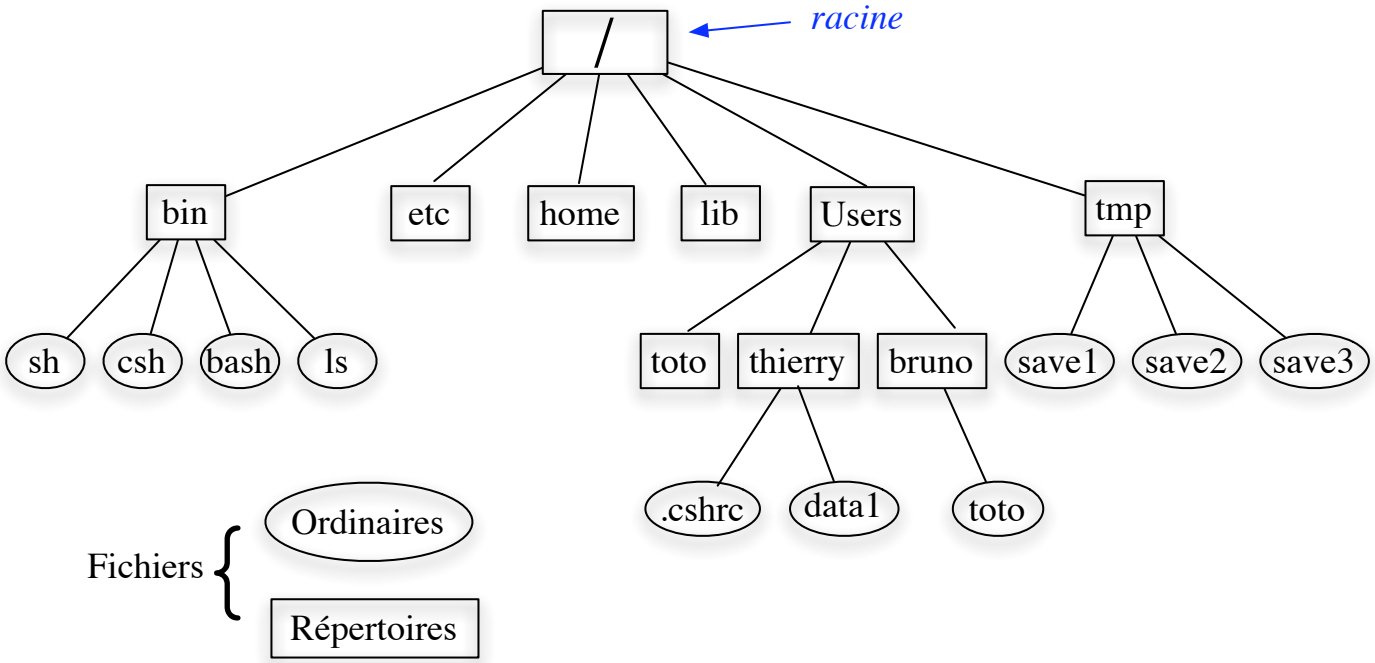
Arborescence

- Modèle hiérarchique en "arbre inversé"
- "arborescence Unix" : 1 seule racine notée "/"
  $\neq$ Windows plusieurs racines (`C:` `D:` `A:` ...)



Arborescence

Exemple d'arborescence Unix :



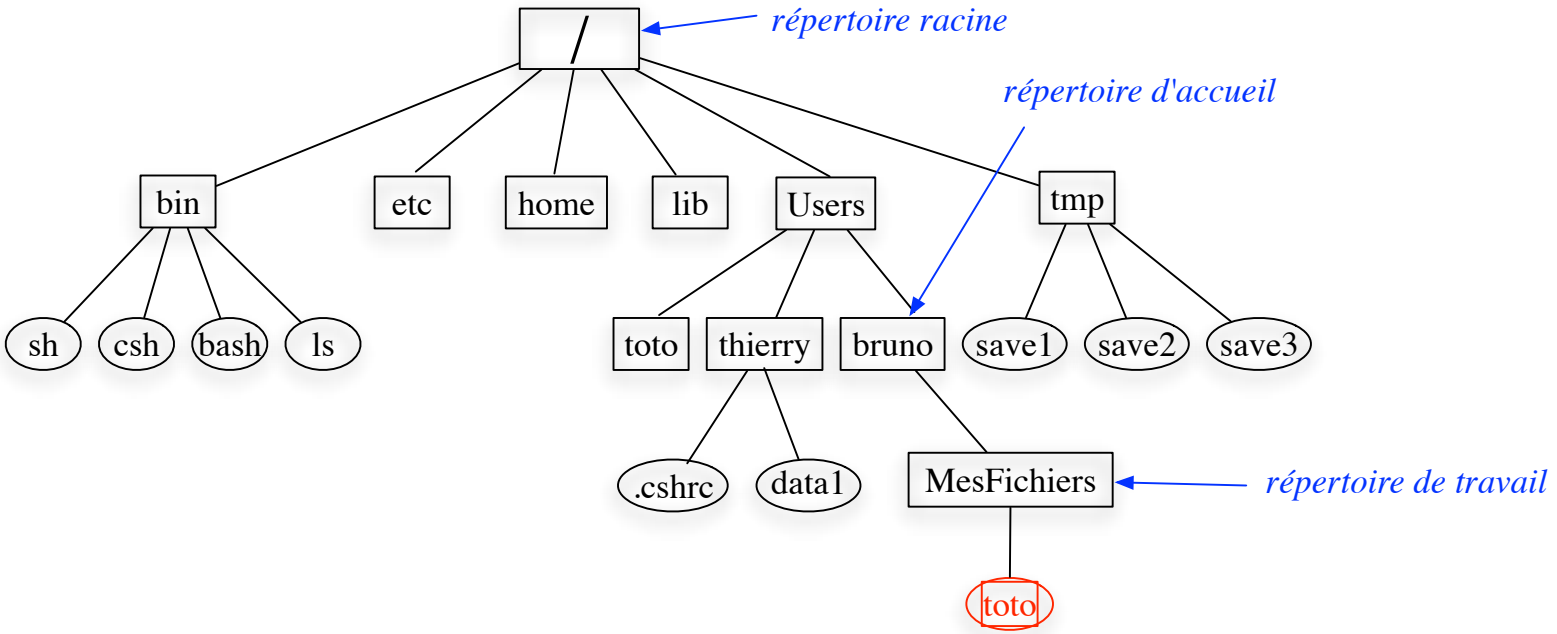
Noms des répertoires spéciaux

Vocabulaire relatif aux répertoires :

- répertoire d'accueil ou *home* :
  → répertoire sur lequel on est positionné à la connexion
  → symbole ‘‘∼’’
- répertoire courant ou *working directory* :
  → répertoire sur lequel on se trouve à tout moment → symbole ‘‘.’’
- répertoire père :
  → répertoire/noeud au dessus du répertoire courant dans l'arborescence Unix
  → symbole ‘‘..’’

Plan

Chemins d'accès à un fichier



- par chemin absolu : `/Users/bruno/MesFichiers/toto`
- par chemin relatif au répertoire d'accueil `~/MesFichiers/toto`
- par chemin relatif au répertoire courant `./toto` ou `toto`

Commandes sur les répertoires

Uniquement sur les répertoires :

- `pwd` (path working directory)
- `cd` (change directory)
- `mkdir` (make directory)
- `rmdir` (remove directory)

Sur les répertoires et fichiers :

- `ls` (list)
- `cp` (copy)
- `mv` (move)
- `rm` (remove)

Retour sur l'affichage d'un répertoire

```
ls [-algiARF...] [name]...
```

Options

- `-a` : all (même commençant par un .)
- `-l` : format long
- `-c` ou `-t` : tri par dernière date de modification
- `-R` : récursif → permet d'afficher toute une arborescence

Exemples

```
homel% ls /bin
X11     grep    roffbib   ...

homel% ls -l
drwxr-xr-x   2 jacob ens    512 mar 26  2003 TclTk
drwxr-xr-x  21 jacob ens   9216 sep 18 17:29 Temporaire
drwxr-xr-x   3 jacob ens    512 jan 10  2002 tst
-rw-r--r--   1 jacob ens    148 jui  7 16:41 uhb.fr

homel% ls -a
.                bin              kadb
..               cdrom            lib
.cshrc           dev              mnt
.login           etc              net

homel% ls -R
 ...
```

Copie fichiers/réperoires

`cp` Copy, 3 utilisations

- Recopie de fichiers
  `cp filename1 filename2` : Recopie du fichier filename1 dans le fichier filename2
- Copie de répertoire
  `cp -rR [-ip] dirname1 dirname2` : Copie récursive de dirname1 dans dirname2
- Copie de fichiers dans un répertoire
  `cp [-iprR] filename... dirname` : Copie des fichiers filename dans dirname
  (dirname doit déjà exister)

Déplacement de fichiers/répertoires

`mv` Déplace ou renomme un fichier/un répertoire

- `mv filename1 filename2` : Renomme filename1 en filename2
- `mv dirname1 dirname2` Si dirname2 n'existe pas, renomme dirname1 en dirname2
- `mv filename... dirname` : Place les fichiers filename dans le répertoire dirname

```
home% mv /home/deust/isr1/  /home/deust/isr2
```

Destructions de fichiers

`rm` Remove

```
rm [-] [-fir] filename...
```

Options :

- -r Destruction récursive
- -i Mode interactif
- -f Force

Suite ...

Suite des aventures en TP ... où vous aurez besoin :

- commande `tar`
  - "mise à plat" d'une arborescence de fichiers dans un seul fichier
  - convention : suffixe .tar
- commande `env` / `printenv`
  - affichage des variables utilisateur
  - correspond à votre environnement
- commande `echo`
  - affichage à l'écran d'une chaine de caractères
  - du contenu d'une variable avec $

Exemples ...