

## Modèle de Von Neumann

- Les ordinateurs modernes sont construits autour d'un modèle défini par le mathématicien John Von Neumann dans les années 1940 appelé [Architecture de Von Neumann](#)

## Modèle de Von Neumann

- Les ordinateurs modernes sont construits autour d'un modèle défini par le mathématicien John Von Neumann dans les années 1940 appelé [Architecture de Von Neumann](#)
- Dans ce modèle, l'ordinateur se décompose :

## Modèle de Von Neumann

- Les ordinateurs modernes sont construits autour d'un modèle défini par le mathématicien John Von Neumann dans les années 1940 appelé **Architecture de Von Neumann**
- Dans ce modèle, l'ordinateur se décompose :
  - La **mémoire** qui stocke les données et les programmes (sous forme de 0 et de 1).

## Modèle de Von Neumann

- Les ordinateurs modernes sont construits autour d'un modèle défini par le mathématicien John Von Neumann dans les années 1940 appelé **Architecture de Von Neumann**
- Dans ce modèle, l'ordinateur se décompose :
  - La **mémoire** qui stocke les données et les programmes (sous forme de 0 et de 1).
  - L'**unité arithmétique et logique UAL** qui effectue les opérations arithmétiques (addition, soustraction, ...) et logiques (conjonctions, négations, ...) sur les données.

## Modèle de Von Neumann

- Les ordinateurs modernes sont construits autour d'un modèle défini par le mathématicien John Von Neumann dans les années 1940 appelé **Architecture de Von Neumann**
- Dans ce modèle, l'ordinateur se décompose :
  - La **mémoire** qui stocke les données et les programmes (sous forme de 0 et de 1).
  - L'**unité arithmétique et logique UAL** qui effectue les opérations arithmétiques (addition, soustraction, ...) et logiques (conjonctions, négations, ...) sur les données.
  - L'**unité de contrôle** chargé de l'ordre des opérations et de la récupération des données en mémoire.

## Modèle de Von Neumann

- Les ordinateurs modernes sont construits autour d'un modèle défini par le mathématicien John Von Neumann dans les années 1940 appelé **Architecture de Von Neumann**
- Dans ce modèle, l'ordinateur se décompose :
  - La **mémoire** qui stocke les données et les programmes (sous forme de 0 et de 1).
  - L'**unité arithmétique et logique UAL** qui effectue les opérations arithmétiques (addition, soustraction, ...) et logiques (conjonctions, négations, ...) sur les données.
  - L'**unité de contrôle** chargé de l'ordre des opérations et de la récupération des données en mémoire.
  - Les dispositifs d'**entrée** (ex : clavier, souris, réseau, ...), et de **sortie** (ex : écran, imprimante, ...) des données

Schéma représentant l'architecture de Von Neumann :



Schéma représentant l'architecture de Von Neumann :

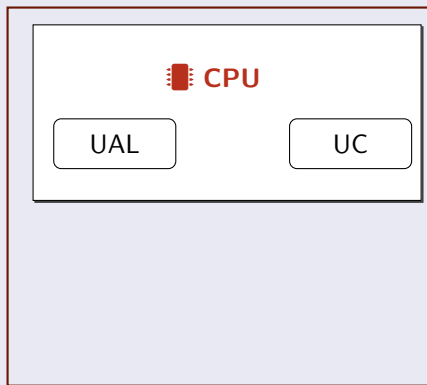




Schéma représentant l'architecture de Von Neumann :

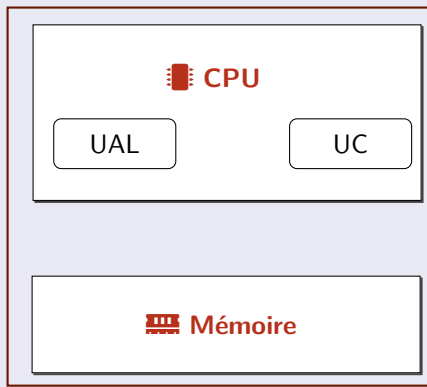


Schéma représentant l'architecture de Von Neumann :

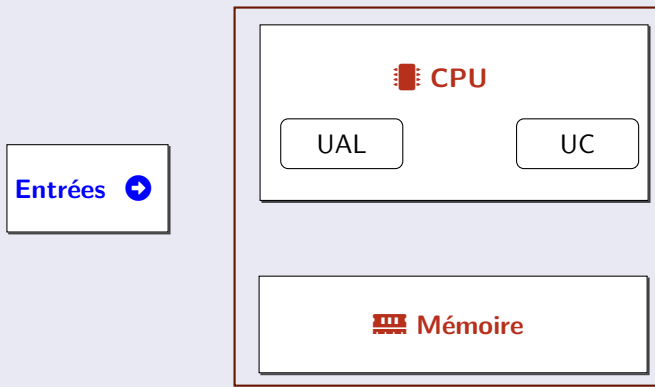


Schéma représentant l'architecture de Von Neumann :

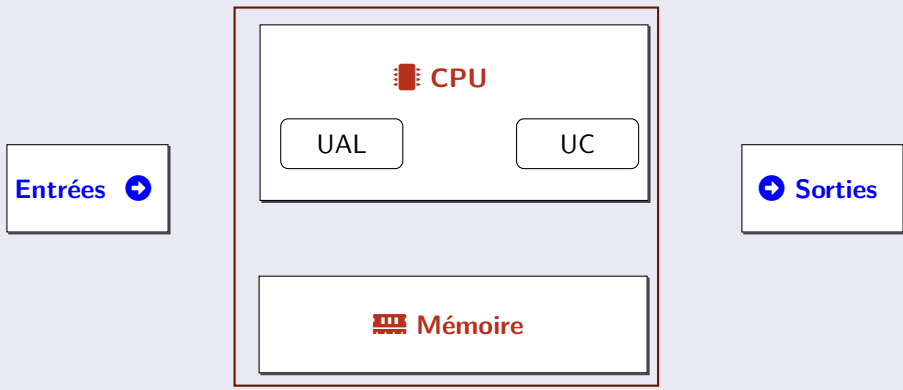


Schéma représentant l'architecture de Von Neumann :

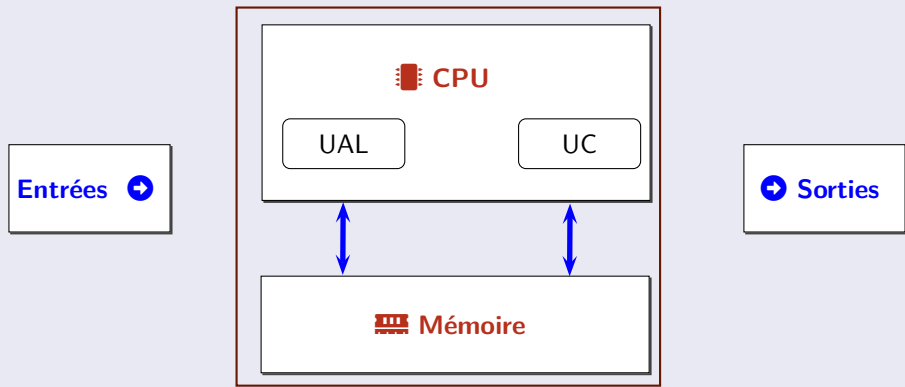
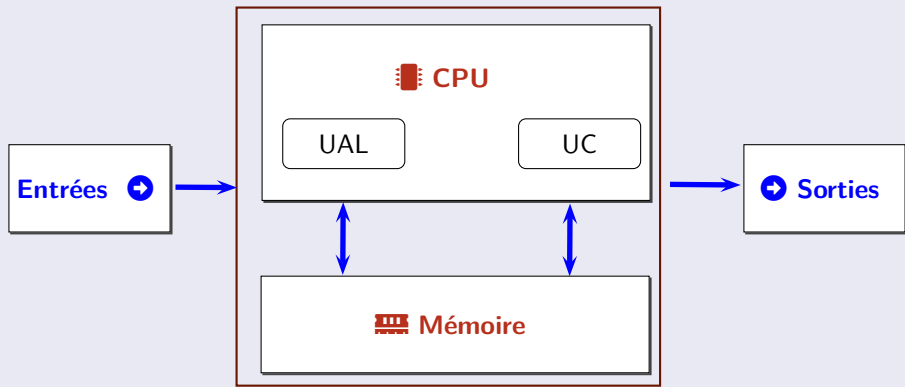


Schéma représentant l'architecture de Von Neumann :



## Définition

Un **système d'exploitation** (en abrégé **OS**, de l'anglais *Operating System*) est un programme (ou ensemble de programme) permettant de

## Définition

Un **système d'exploitation** (en abrégé **OS**, de l'anglais *Operating System*) est un programme (ou ensemble de programme) permettant de gérer les ressources de l'ordinateur (mémoire, fichier, périphériques, ...) sur lequel il s'exécute.

## Définition

Un **système d'exploitation** (en abrégé **OS**, de l'anglais *Operating System*) est un programme (ou ensemble de programme) permettant de gérer les ressources de l'ordinateur (mémoire, fichier, périphériques, ...) sur lequel il s'exécute.

## Exemples

Les systèmes d'exploitation les plus répandus à l'heure actuelle sont :



## Définition

Un **système d'exploitation** (en abrégé **OS**, de l'anglais *Operating System*) est un programme (ou ensemble de programme) permettant de gérer les ressources de l'ordinateur (mémoire, fichier, périphériques, ...) sur lequel il s'exécute.

## Exemples

Les systèmes d'exploitation les plus répandus à l'heure actuelle sont :

-  Windows (différentes versions)

## Définition

Un **système d'exploitation** (en abrégé **OS**, de l'anglais *Operating System*) est un programme (ou ensemble de programme) permettant de gérer les ressources de l'ordinateur (mémoire, fichier, périphériques, ...) sur lequel il s'exécute.

## Exemples

Les systèmes d'exploitation les plus répandus à l'heure actuelle sont :



Windows (différentes versions)






GNU/Linux (plusieurs centaines de distribution différentes, parmi les plus connus : ubuntu, fedora, archlinux)

## Définition

Un **système d'exploitation** (en abrégé **OS**, de l'anglais *Operating System*) est un programme (ou ensemble de programme) permettant de gérer les ressources de l'ordinateur (mémoire, fichier, périphériques, ...) sur lequel il s'exécute.

## Exemples

Les systèmes d'exploitation les plus répandus à l'heure actuelle sont :





-  Windows (différentes versions)
-  GNU/Linux (plusieurs centaines de distribution différentes, parmi les plus connus : ubuntu, fedora, archlinux)
-  Android (smartphone)

## Définition

Un **système d'exploitation** (en abrégé **OS**, de l'anglais *Operating System*) est un programme (ou ensemble de programme) permettant de gérer les ressources de l'ordinateur (mémoire, fichier, périphériques, ...) sur lequel il s'exécute.

## Exemples

Les systèmes d'exploitation les plus répandus à l'heure actuelle sont :

-  Windows (différentes versions)
-  GNU/Linux (plusieurs centaines de distribution différentes, parmi les plus connus : ubuntu, fedora, archlinux)
-  Android (smartphone)
-  MacOS (ordinateur) et iOS (smartphone)

## La place du système d'exploitation

## La place du système d'exploitation

 L'utilisateur

## La place du système d'exploitation

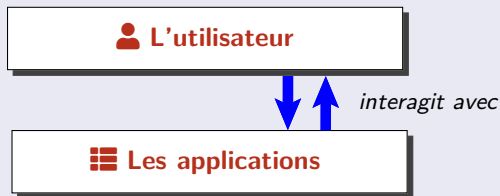
 L'utilisateur Les applications

## La place du système d'exploitation

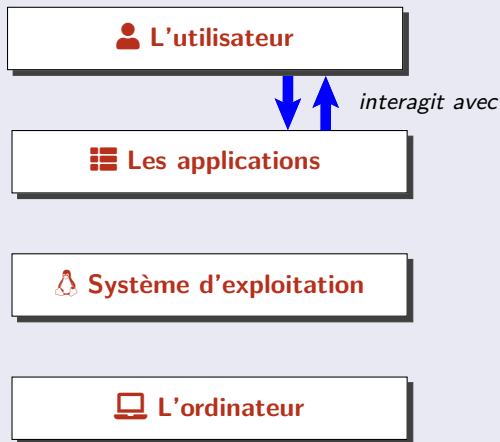




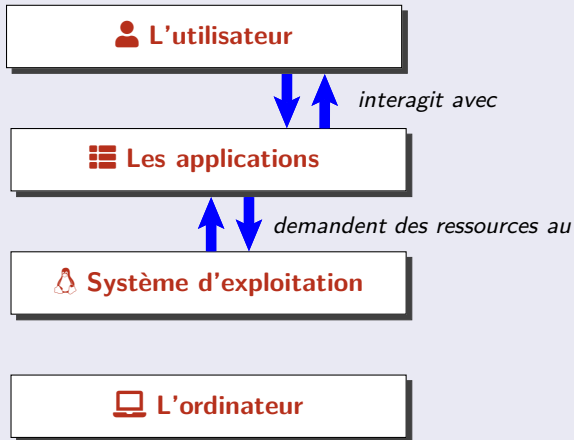
## La place du système d'exploitation



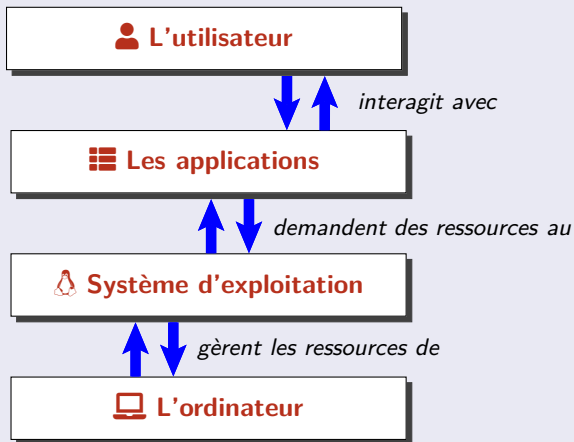
## La place du système d'exploitation



## La place du système d'exploitation



## La place du système d'exploitation



### Fonctionnalités d'un système d'exploitation

- Gestion des périphériques.

### Fonctionnalités d'un système d'exploitation

- Gestion des périphériques.
- Donner l'illusion que l'ordinateur est multitâches.

### Fonctionnalités d'un système d'exploitation

- Gestion des périphériques.
- Donner l'illusion que l'ordinateur est multitâches.
- Gérer les différentes applications utilisées.

## Fonctionnalités d'un système d'exploitation

- Gestion des périphériques.
- Donner l'illusion que l'ordinateur est multitâches.
- Gérer les différentes applications utilisées.
- Identifier les utilisateurs.



### Fonctionnalités d'un système d'exploitation

- Gestion des périphériques.
- Donner l'illusion que l'ordinateur est multitâches.
- Gérer les différentes applications utilisées.
- Identifier les utilisateurs.
- Contrôler et distribuer les accès aux ressources de l'ordinateur.

### Fonctionnalités d'un système d'exploitation

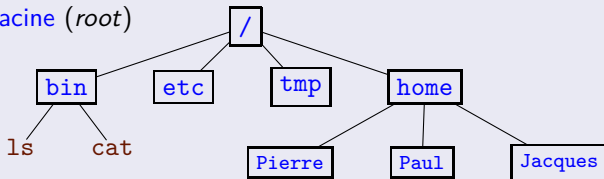
- Gestion des périphériques.
- Donner l'illusion que l'ordinateur est multitâches.
- Gérer les différentes applications utilisées.
- Identifier les utilisateurs.
- Contrôler et distribuer les accès aux ressources de l'ordinateur.
- Gérer le système de fichier.

## Le système de gestion de fichiers

- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)

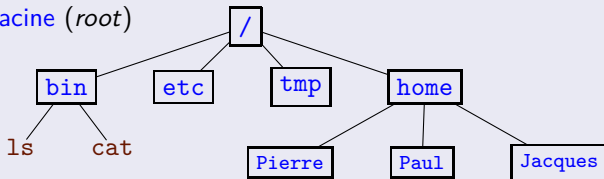
## Le système de gestion de fichiers

- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)



## Le système de gestion de fichiers

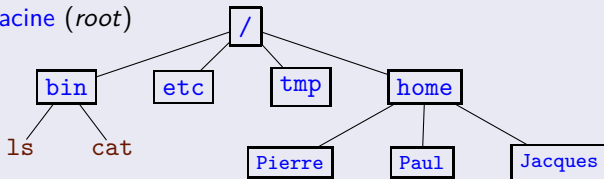
- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)



- Les fichiers où les répertoires sont spécifiés :

## Le système de gestion de fichiers

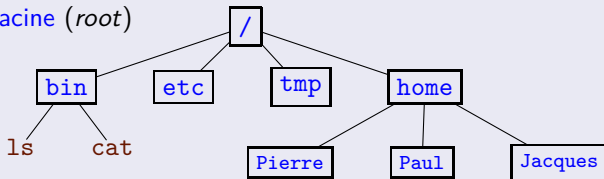
- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)



- Les fichiers où les répertoires sont spécifiés :
  - s'ils commencent par / en chemin **absolu** c'est à dire depuis la racine.

## Le système de gestion de fichiers

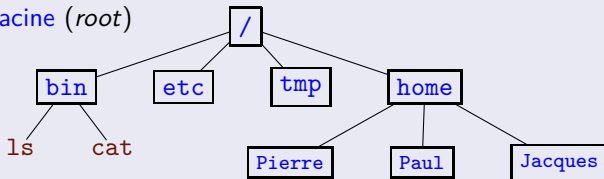
- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)



- Les fichiers où les répertoires sont spécifiés :
  - s'ils commencent par / en chemin **absolu** c'est à dire depuis la racine.
  - sinon en chemin **relatif** c'est à dire depuis le répertoire courant (le répertoire parent se note alors **..**).

## Le système de gestion de fichiers

- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)

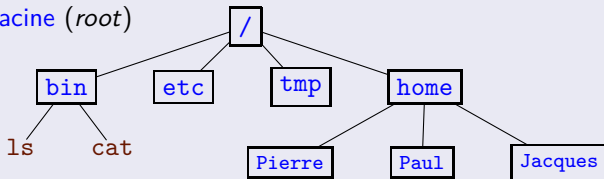


- Les fichiers où les répertoires sont spécifiés :
  - s'ils commencent par / en chemin **absolu** c'est à dire depuis la racine.
  - sinon en chemin **relatif** c'est à dire depuis le répertoire courant (le répertoire parent se note alors ..).
- Trois type de droits sont définis sur les fichiers et dossiers :
  - r** droit de lecture du fichier



## Le système de gestion de fichiers

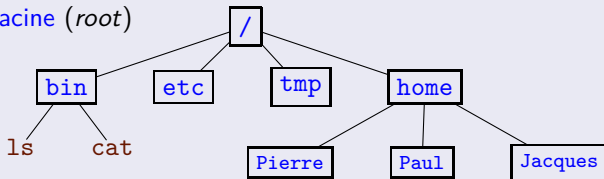
- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)



- Les fichiers où les répertoires sont spécifiés :
  - s'ils commencent par / en chemin **absolu** c'est à dire depuis la racine.
  - sinon en chemin **relatif** c'est à dire depuis le répertoire courant (le répertoire parent se note alors **..**).
- Trois type de droits sont définis sur les fichiers et dossiers :
  - r** droit de lecture du fichier
  - w** droit d'écriture dans le fichier

## Le système de gestion de fichiers

- L'ensemble des fichiers forme une arborescence démarrant au répertoire / appelé **racine** (*root*)



- Les fichiers où les répertoires sont spécifiés :
  - s'ils commencent par / en chemin **absolu** c'est à dire depuis la racine.
  - sinon en chemin **relatif** c'est à dire depuis le répertoire courant (le répertoire parent se note alors ..).
- Trois type de droits sont définis sur les fichiers et dossiers :
  - r droit de lecture du fichier
  - w droit d'écriture dans le fichier
  - x droit d'exécution du fichier

## Interface système : *shell*

- Avant l'avènement des interfaces graphiques (et de la souris), l'utilisateur communiquait avec le système d'exploitation via un programme appelé *shell* par l'intermédiaire d'un simple clavier et d'une interface en ligne de commande (CLI en anglais pour *Command Line Interface*).

## Interface système : *shell*

- Avant l'avènement des interfaces graphiques (et de la souris), l'utilisateur communiquait avec le système d'exploitation via un programme appelé *shell* par l'intermédiaire d'un simple clavier et d'une interface en ligne de commande (CLI en anglais pour *Command Line Interface*).
- Aujourd'hui encore et pour diverses raisons (rapidité, contrôle plus fin de l'ordinateur, récupération d'erreurs, ...) la ligne de commande reste très utilisée.

Interface système : *shell*

- Avant l'avènement des interfaces graphiques (et de la souris), l'utilisateur communiquait avec le système d'exploitation via un programme appelé *shell* par l'intermédiaire d'un simple clavier et d'une interface en ligne de commande (CLI en anglais pour *Command Line Interface*).
- Aujourd'hui encore et pour diverses raisons (rapidité, contrôle plus fin de l'ordinateur, récupération d'erreurs, ...) la ligne de commande reste très utilisée.
- Voici un exemple d'invite de commande :

```
neo@matrix:~/Morpheus$
```

On y trouve :

Interface système : *shell*

- Avant l'avènement des interfaces graphiques (et de la souris), l'utilisateur communiquait avec le système d'exploitation via un programme appelé *shell* par l'intermédiaire d'un simple clavier et d'une interface en ligne de commande (CLI en anglais pour *Command Line Interface*).
- Aujourd'hui encore et pour diverses raisons (rapidité, contrôle plus fin de l'ordinateur, récupération d'erreurs, ...) la ligne de commande reste très utilisée.
- Voici un exemple d'invite de commande :

```
neo@matrix:~/Morpheus$
```

On y trouve :

- Le nom de l'utilisateur ici **neo** suivi de **@**

Interface système : *shell*

- Avant l'avènement des interfaces graphiques (et de la souris), l'utilisateur communiquait avec le système d'exploitation via un programme appelé *shell* par l'intermédiaire d'un simple clavier et d'une interface en ligne de commande (CLI en anglais pour *Command Line Interface*).
- Aujourd'hui encore et pour diverses raisons (rapidité, contrôle plus fin de l'ordinateur, récupération d'erreurs, ...) la ligne de commande reste très utilisée.
- Voici un exemple d'invite de commande :

```
neo@matrix:~/Morpheus$
```

On y trouve :

- Le nom de l'utilisateur ici **neo** suivi de **@**
- Le nom de l'ordinateur ici **matrix** suivi de **:**

## Interface système : *shell*

- Avant l'avènement des interfaces graphiques (et de la souris), l'utilisateur communiquait avec le système d'exploitation via un programme appelé *shell* par l'intermédiaire d'un simple clavier et d'une interface en ligne de commande (CLI en anglais pour *Command Line Interface*).
- Aujourd'hui encore et pour diverses raisons (rapidité, contrôle plus fin de l'ordinateur, récupération d'erreurs, ...) la ligne de commande reste très utilisée.
- Voici un exemple d'invite de commande :

```
neo@matrix:~/Morpheus$
```

On y trouve :

- Le nom de l'utilisateur ici **neo** suivi de **@**
- Le nom de l'ordinateur ici **matrix** suivi de **:**
- Le chemin du dossier de travail ici **~/Morpheus** suivi de **:** et du curseur



## Généralités sur le shell

- Les commandes ont généralement le format suivant :  
`<nom commande> <option> <arguments>`  
où les options sont précédées d'un tiret simple `-` ou double `--`

## Généralités sur le shell

- Les commandes ont généralement le format suivant :

`<nom commande> <option> <arguments>`

où les options sont précédées d'un tiret simple `-` ou double `--`

- certains caractères spéciaux permettent d'agir sur un ensemble d'arguments :

<code>?</code>	correspond à n'importe quel caractère
<code>*</code>	correspond à n'importe quel suite de caractères
<code>[...]</code>	correspond aux caractères entre crochets

## Généralités sur le `shell`

- Les commandes ont généralement le format suivant :

`<nom commande> <option> <arguments>`

où les options sont précédées d'un tiret simple `-` ou double `--`

- certains caractères spéciaux permettent d'agir sur un ensemble d'arguments :

<code>?</code>	correspond à n'importe quel caractère
<code>*</code>	correspond à n'importe quel suite de caractères
<code>[...]</code>	correspond aux caractères entre crochets

- Le résultat d'une commande peut-être dirigé :

## Généralités sur le `shell`

- Les commandes ont généralement le format suivant :  
`<nom commande> <option> <arguments>`  
où les options sont précédées d'un tiret simple `-` ou double `--`
- certains caractères spéciaux permettent d'agir sur un ensemble d'arguments :

<code>?</code>	correspond à n'importe quel caractère
<code>*</code>	correspond à n'importe quel suite de caractères
<code>[...]</code>	correspond aux caractères entre crochets

- Le résultat d'une commande peut-être dirigé :
  - Vers un fichier avec `>` (avec écrasement si le fichier existe)

## Généralités sur le shell

- Les commandes ont généralement le format suivant :  
`<nom commande> <option> <arguments>`  
où les options sont précédées d'un tiret simple `-` ou double `--`
- certains caractères spéciaux permettent d'agir sur un ensemble d'arguments :

<code>?</code>	correspond à n'importe quel caractère
<code>*</code>	correspond à n'importe quel suite de caractères
<code>[...]</code>	correspond aux caractères entre crochets

- Le résultat d'une commande peut-être dirigé :
  - Vers un fichier avec `>` (avec écrasement si le fichier existe)
  - Vers un fichier avec `>>` (avec ajout en fin si le fichier existe)

## Généralités sur le shell

- Les commandes ont généralement le format suivant :  
`<nom commande> <option> <arguments>`  
où les options sont précédées d'un tiret simple `-` ou double `--`
- certains caractères spéciaux permettent d'agir sur un ensemble d'arguments :

<code>?</code>	correspond à n'importe quel caractère
<code>*</code>	correspond à n'importe quel suite de caractères
<code>[...]</code>	correspond aux caractères entre crochets

- Le résultat d'une commande peut-être dirigé :
  - Vers un fichier avec `>` (avec écrasement si le fichier existe)
  - Vers un fichier avec `>>` (avec ajout en fin si le fichier existe)
  - Vers une autre commande avec `|` (*pipe*)

## Quelques commandes

`man` : affiche l'aide sur une commande

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant



## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

`rmdir` : supprime un dossier

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

`rmdir` : supprime un dossier

`mv` : déplace un dossier ou un fichier

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

`rmdir` : supprime un dossier

`mv` : déplace un dossier ou un fichier

`ls` : liste le contenu d'un dossier

- a affiche les fichiers cachés (dont le nom commence par un point)

- l permet de voir les droits sur les fichiers

- i permet de voir les inodes

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

`rmdir` : supprime un dossier

`mv` : déplace un dossier ou un fichier

`ls` : liste le contenu d'un dossier

- a affiche les fichiers cachés (dont le nom commence par un point)

- l permet de voir les droits sur les fichiers

- i permet de voir les inodes

`cat` : écrit le contenu d'un fichier dans le terminal

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

`rmdir` : supprime un dossier

`mv` : déplace un dossier ou un fichier

`ls` : liste le contenu d'un dossier

`-a` affiche les fichiers cachés (dont le nom commence par un point)

`-l` permet de voir les droits sur les fichiers

`-i` permet de voir les inodes

`cat` : écrit le contenu d'un fichier dans le terminal

`touch` : crée un fichier vide

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

`rmdir` : supprime un dossier

`mv` : déplace un dossier ou un fichier

`ls` : liste le contenu d'un dossier

`-a` affiche les fichiers cachés (dont le nom commence par un point)

`-l` permet de voir les droits sur les fichiers

`-i` permet de voir les inodes

`cat` : écrit le contenu d'un fichier dans le terminal

`touch` : crée un fichier vide

`echo` : écrit dans le terminal

## Quelques commandes

`man` : affiche l'aide sur une commande

`pwd` : affiche le répertoire courant

`mkdir` : crée un ou plusieurs dossiers

`rmdir` : supprime un dossier

`mv` : déplace un dossier ou un fichier

`ls` : liste le contenu d'un dossier

`-a` affiche les fichiers cachés (dont le nom commence par un point)

`-l` permet de voir les droits sur les fichiers

`-i` permet de voir les inodes

`cat` : écrit le contenu d'un fichier dans le terminal

`touch` : crée un fichier vide

`echo` : écrit dans le terminal

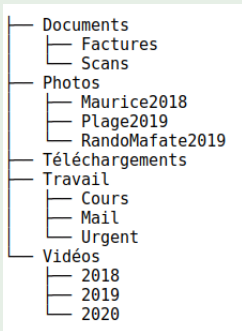
`cp` : copie un fichier



## Exemples

A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`

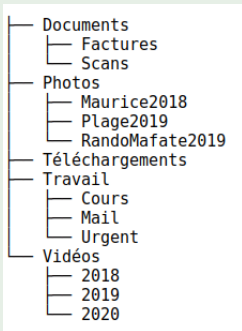


## Exemples

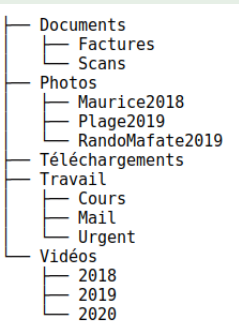
A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`

```
touch to_do
```



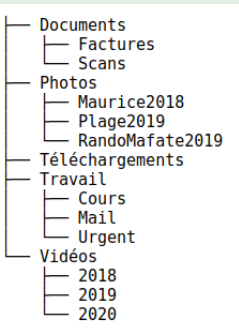
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`  
`touch to_do`
- 2 Ecrire dans ce fichier : "réviser le chapitre 0"

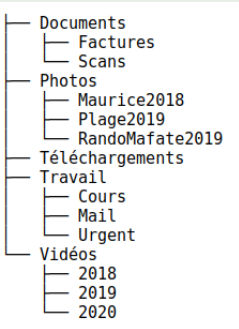
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`  
`touch to_do`
- 2 Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`

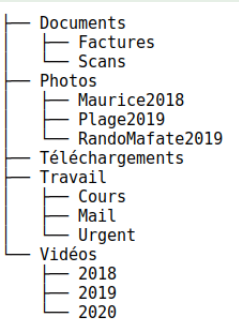
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`  
`touch to_do`
- 2 Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- 3 se déplacer vers le dossier Factures (chemin relatif)

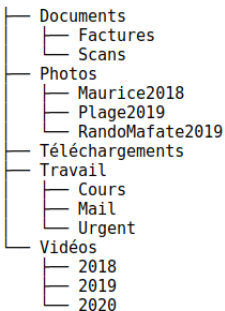
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`  
`touch to_do`
- 2 Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- 3 se déplacer vers le dossier Factures (chemin relatif)  
`cd ../../Documents/Factures`

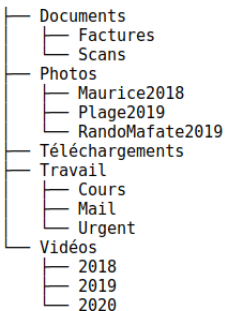
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- ❶ Créer le fichier vide `to_do`  
`touch to_do`
- ❷ Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- ❸ se déplacer vers le dossier Factures (chemin relatif)  
`cd ../../Documents/Factures`
- ❹ y créer les dossiers Eau, Electricité et Téléphone

## Exemples

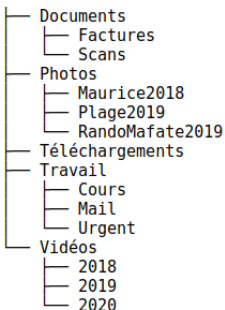


A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`  
`touch to_do`
- 2 Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- 3 se déplacer vers le dossier Factures (chemin relatif)  
`cd ../../Documents/Factures`
- 4 y créer les dossiers Eau, Electricité et Téléphone  
`mkdir Eau Electricité Téléphone`



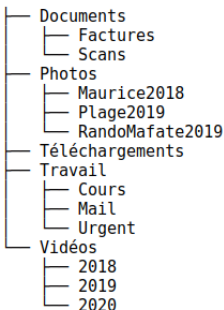
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- ❶ Créer le fichier vide `to_do`  
`touch to_do`
- ❷ Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- ❸ se déplacer vers le dossier Factures (chemin relatif)  
`cd ../../Documents/Factures`
- ❹ y créer les dossiers Eau, Electricité et Téléphone  
`mkdir Eau Electricité Téléphone`
- ❺ lister tous le fichiers contenant `edf`

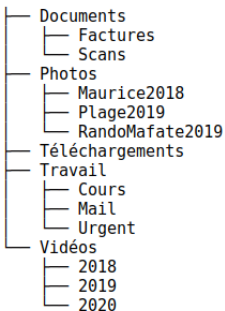
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- ❶ Créer le fichier vide `to_do`  
`touch to_do`
- ❷ Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- ❸ se déplacer vers le dossier Factures (chemin relatif)  
`cd ../../Documents/Factures`
- ❹ y créer les dossiers Eau, Electricité et Téléphone  
`mkdir Eau Electricité Téléphone`
- ❺ lister tous le fichiers contenant edf  
`ls *edf*`

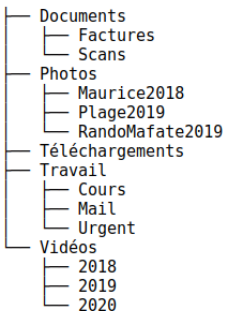
## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- 1 Créer le fichier vide `to_do`  
`touch to_do`
- 2 Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- 3 se déplacer vers le dossier Factures (chemin relatif)  
`cd ../../Documents/Factures`
- 4 y créer les dossiers Eau, Electricité et Téléphone  
`mkdir Eau Electricité Téléphone`
- 5 lister tous les fichiers contenant `edf`  
`ls *edf*`
- 6 Les déplacer dans le dossier Electricité

## Exemples



A partir du répertoire Cours, écrire les commandes pour :

- ❶ Créer le fichier vide `to_do`  
`touch to_do`
- ❷ Ecrire dans ce fichier : "réviser le chapitre 0"  
`echo "réviser le chapitre 0" > to_do`
- ❸ se déplacer vers le dossier Factures (chemin relatif)  
`cd ../../Documents/Factures`
- ❹ y créer les dossiers Eau, Electricité et Téléphone  
`mkdir Eau Electricité Téléphone`
- ❺ lister tous le fichiers contenant `edf`  
`ls *edf*`
- ❻ Les déplacer dans le dossier Electricité  
`mv *edf* Electricité`

Droits sur un fichier, `chmod`

Exemples

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).

## Exemples

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.

## Exemples

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples



## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w :`

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w` : Ajoute (+) le droit d'écriture (w) au propriétaire (g)

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w` : Ajoute (+) le droit d'écriture (`w`) au propriétaire (`g`)
- `chmod 700` :

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w` : Ajoute (+) le droit d'écriture (`w`) au propriétaire (`g`)
- `chmod 700` : Le propriétaire a tous les droits, les autres et le groupe aucun

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w` : Ajoute (+) le droit d'écriture (`w`) au propriétaire (`g`)
- `chmod 700` : Le propriétaire a tous les droits, les autres et le groupe aucun
- `chmod og-r` :

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w` : Ajoute (+) le droit d'écriture (`w`) au propriétaire (`g`)
- `chmod 700` : Le propriétaire a tous les droits, les autres et le groupe aucun
- `chmod og-r` : Enlève (-) le droit de lecture (`r`) au groupe et aux autres (`og`)

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w` : Ajoute (+) le droit d'écriture (`w`) au propriétaire (`g`)
- `chmod 700` : Le propriétaire a tous les droits, les autres et le groupe aucun
- `chmod og-r` : Enlève (-) le droit de lecture (`r`) au groupe et aux autres (`og`)
- `chmod 544` :

## Droits sur un fichier, `chmod`

- Les droits sur un fichier (**r** : lecture, **w** : écriture, **x** exécution) sont définis pour le propriétaire : **u** (*user*), le groupe : **g** (*groupe*) et les autres : **o** (*others*).
- La commande `ls -l` permet d'afficher les droits, par exemple :  
`rwxr-x--` indique que l'utilisateur a tous les droits, le groupe peut lire et écrire et les autres n'ont aucun droit.
- La commande `chmod` permet de modifier les droits, à l'aide de + (ajout), - (retrait), = (attribution) ou en notation octale (`r=4`, `w=2`, `x=1`).

## Exemples

- `chmod u+w` : Ajoute (+) le droit d'écriture (`w`) au propriétaire (`g`)
- `chmod 700` : Le propriétaire a tous les droits, les autres et le groupe aucun
- `chmod og-r` : Enlève (-) le droit de lecture (`r`) au groupe et aux autres (`og`)
- `chmod 544` : Le propriétaire peut lire et exécuter, le groupe et les autres peuvent lire



## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).

## Exemples

## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).
- Deux noms de fichiers différents peuvent référencer les mêmes données, ils partagent alors le même inode, on dit que c'est un lien physique (*hardlink*).

## Exemples

## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).
- Deux noms de fichiers différents peuvent référencer les mêmes données, ils partagent alors le même inode, on dit que c'est un lien physique (*hardlink*).
- Un lien symbolique (*softlink*) est un fichier indiquant un chemin vers un autre fichier (équivalent d'un raccourci de *Windows*).

## Exemples

## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).
- Deux noms de fichiers différents peuvent référencer les mêmes données, ils partagent alors le même inode, on dit que c'est un lien physique (*hardlink*).
- Un lien symbolique (*softlink*) est un fichier indiquant un chemin vers un autre fichier (équivalent d'un raccourci de *Windows*).
- La commande **ln** (resp. **ln -s**) permet de créer un lien physique (resp. symbolique).

## Exemples

## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).
- Deux noms de fichiers différents peuvent référencer les mêmes données, ils partagent alors le même inode, on dit que c'est un lien physique (*hardlink*).
- Un lien symbolique (*softlink*) est un fichier indiquant un chemin vers un autre fichier (équivalent d'un raccourci de *Windows*).
- La commande **ln** (resp. **ln -s**) permet de créer un lien physique (resp. symbolique).

## Exemples

- `ln important.txt ../Sauvegarde/important.sav`

## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).
- Deux noms de fichiers différents peuvent référencer les mêmes données, ils partagent alors le même inode, on dit que c'est un lien physique (*hardlink*).
- Un lien symbolique (*softlink*) est un fichier indiquant un chemin vers un autre fichier (équivalent d'un raccourci de *Windows*).
- La commande **ln** (resp. **ln -s**) permet de créer un lien physique (resp. symbolique).

## Exemples

- `ln important.txt ../Sauvegarde/important.sav`  
Les deux noms de fichiers font référence aux mêmes données

## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).
- Deux noms de fichiers différents peuvent référencer les mêmes données, ils partagent alors le même inode, on dit que c'est un lien physique (*hardlink*).
- Un lien symbolique (*softlink*) est un fichier indiquant un chemin vers un autre fichier (équivalent d'un raccourci de *Windows*).
- La commande **ln** (resp. **ln -s**) permet de créer un lien physique (resp. symbolique).

## Exemples

- `ln important.txt ../Sauvegarde/important.sav`  
Les deux noms de fichiers font référence aux mêmes données
- `ln -s important.txt ../Sauvegarde/important.sav`

## Liens physiques ou symboliques

- Les fichiers sont stockés sur le support physique par blocs et retrouvés grâce à leur **inode** (*index node* ou noeud d'index).
- Deux noms de fichiers différents peuvent référencer les mêmes données, ils partagent alors le même inode, on dit que c'est un lien physique (*hardlink*).
- Un lien symbolique (*softlink*) est un fichier indiquant un chemin vers un autre fichier (équivalent d'un raccourci de *Windows*).
- La commande **ln** (resp. **ln -s**) permet de créer un lien physique (resp. symbolique).

## Exemples

- `ln important.txt ../Sauvegarde/important.sav`  
Les deux noms de fichiers font référence aux mêmes données
- `ln -s important.txt ../Sauvegarde/important.sav`  
Création d'une simple redirection