

# Systeme d'exploitation

## processus, systeme de fichiers, shell

qkzk  
Lycée des Flandres

jan 2020

Systeme d'exploitation

Première

- pratique sous Linux avec la clé
- résumé des commandes de base

Terminale

- **processus**
- fichier et systeme de fichiers (rappel)
- avec le shell (rappel)

## Processus

### Processus & programme

Programme

- description statique
- code, suite d'instructions

Processus

- activité dynamique, temporelle
- vie d'un processus : création d'un processus, exécution, fin d'un processus

...

**Un processus est une instance d'exécution d'un programme**

- plusieurs exécutions de programmes
- plusieurs exécutions d'un même programme
- plusieurs exécutions « simultanées » de programmes différents
- plusieurs exécutions « simultanées » d'un même programme

**P... , p... , processeur**

**Programme, processus... processeur**

- entité matérielle
- désigne l'utilisation du processeur

**Affectation du processeur à un processus**

- pour un temps donné
- permet de faire progresser le processus

## Choix de cette affectation = ordonnancement

- système multiprocessus
- choix à la charge du système d'exploitation

## P... , p... , p... , parallélisme, pseudo-parallélisme

### Ordonnancement

- plusieurs processus, un processeur
- entrelacement des processus

### Deux processus



...

→ exécution séquentielle



...

→ exécutions parallèles (deux processeurs)



...

→ exécutions entrelacées



...

→ autre entrelacement



...

- impression pour chacun de disposer d'un processeur
- le système *virtualise* la ressource processeur

## **Processus = abstraction**

### **Processus = exécution abstraite d'un programme**

- indépendante de l'avancement réel de l'exécution

### **Exécution d'un programme = réunion des instants d'exécution réelle du programme**

- dépend de la disponibilité du processeur

## **Processus = abstraction**

- désigne une entité identifiable
- par exemple : priorité d'un processus
- parallélisme, simultanéité, interaction... de deux processus

## **Compétition (*race condition*)**

- résultats de deux processus dépend de cet entrelacement
- par exemple à cause d'accès partagés à un fichier...
- danger potentiel, à éviter...

## **Processus & ressources**

### **Processus = exécution d'un programme**

- requiert des ressources

## **Ressource**

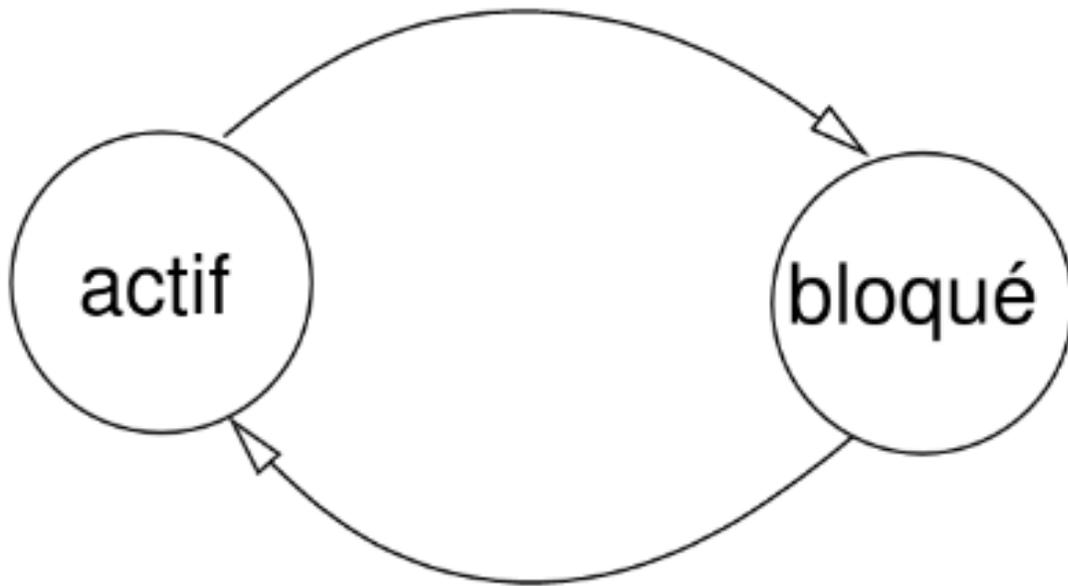
- entité nécessaire à l'exécution d'un processus
- ressources matérielles : processeur, périphérique...
- ressources logicielles : variable...

## **Caractéristiques d'une ressource**

- état : libre, occupée
- nombre de possibles utilisations concurrentes  
(ressource à accès multiples)

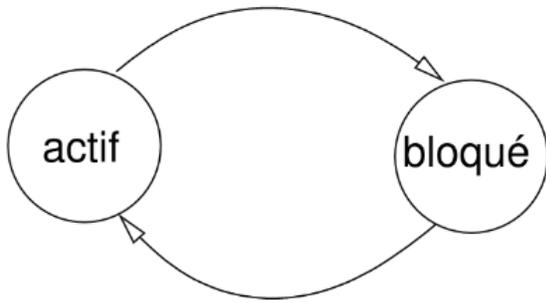
## **Ressources indispensables à un processus**

- mémoire propre (mémoire virtuelle)
- contexte d'exécution (état instantané du processus)
  - pile (en mémoire)
  - registres du processeur



États logiques

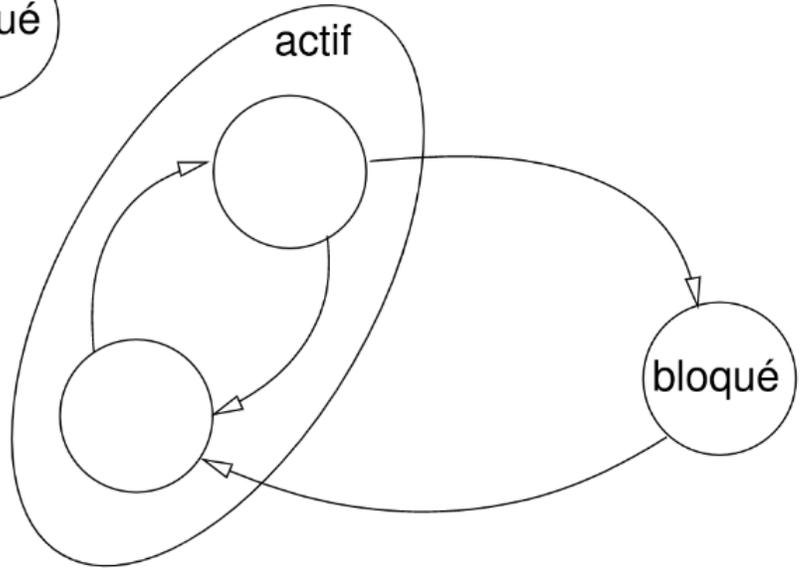
indisponibilité  
d'une ressource



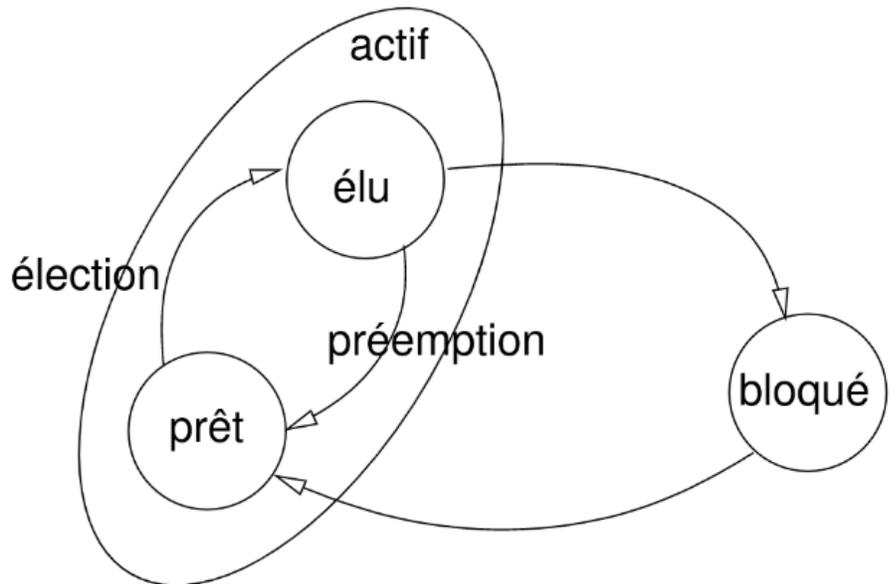
États logiques

particularisation  
de la ressource  
processeur

États effectifs

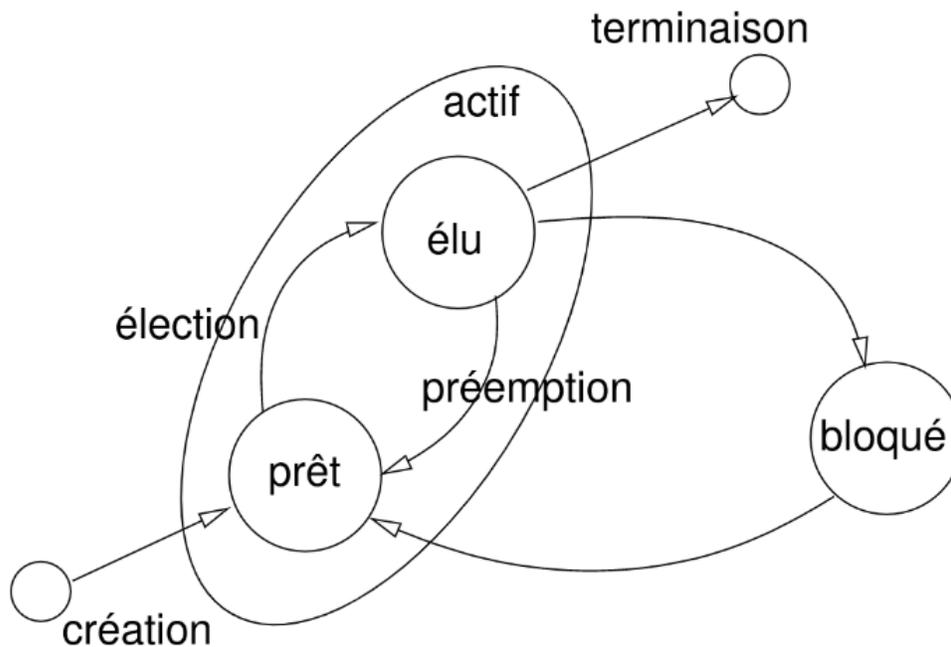


États effectifs



particularisation  
de la ressource  
processeur

## États effectifs



## Attributs d'un processus

### Identification univoque

- PID *process ID*
- numéro entier *pid\_t*

### Propriétaire

- utilisateur qui a lancé le processus, son groupe
- détermine les droits du processus

### Répertoire de travail

- origine de l'interprétation des chemins relatifs

### Hérarchie des processus

- création de processus... par un processus
- chaque processus a donc un processus père
- processus `init` ancêtre de tous les processus
- héritage — répertoire de travail, etc.

## Gérer les processus depuis le shell

### % commande

- création d'un processus qui va exécuter le programme `commande`

### % ps ax

- liste les processus

### % top

- affichage en continu des informations relatives aux processus

— `htop` et `atop` sont des alternatives

% `kill -9 pid`

- « tue » le processus désigné
- envoi d'un signal 9

% `killall nom`

- « tue » les processus désignés par leur nom

## Fichier et système de fichiers

### Données persistantes

#### Processus manipule des données

- conserve en mémoire
- tout au long de son exécution

#### Besoin de conservation des données

- au delà de la fin du processus

#### Besoin de mémoriser de grandes quantités de données

- taille supérieure à la mémoire (virtuelle)

#### Besoin de partage des données

- données accessibles (simultanément) par plusieurs processus

### Fichiers

- mémoriser des données
- sur disques — ou autres « mémoires secondaires »
- de manière persistante

## Système de fichiers

### Système de fichiers = partie du système d'exploitation

- organisation des fichiers
- structuration, nommage, accès, protection, implantation...

### Fichier = mécanisme d'abstraction

- présentation à l'utilisateur
- opérations permises par le système d'exploitation
  - création, lecture/écriture, déplacement, suppression...
- ... à partir d'un nom, chemin d'accès

### Implantation des systèmes de fichiers

- variées — FAT, ext2, ext4, HFS, AFS, NFS...

## Répertoires

### Répertoire = fichier particulier

- mémorise la structure du système de fichiers
- opérations contrôlées par le système d'exploitation

## Fichier ordinaire

- contient les données — suite d'octets (/bits, /blocs)
- sans organisation particulière

## Hiérarchie

### Système de fichiers présente une hiérarchie

- répertoire « *contient* » des fichiers
- racine du système de fichiers
- position courante dans la hiérarchie

### Système de fichiers n'est pas une hiérarchie

- implantation sur la machine est un ensemble de nœuds
- un nœud = un ensemble de blocs de données
- détails d'implémentation cachés

### L'utilisateur doit savoir que le système de fichiers n'est pas une hiérarchie

- répertoire *contient une liste de noms* d'entrées
- manipulation des liens symboliques
- manipulation des liens physiques

## Système de fichiers arborescent

### Le système de fichier est un arbre

- vue simplificatrice (... sur laquelle on reviendra)
- arbre = racine + nœuds à un parent unique + arcs

### Racine

- notée /
- est son propre parent

### Arcs ou entrées

- nommés, tous caractères sauf \0 et /
- éviter les espaces, les non imprimables, et non ASCII

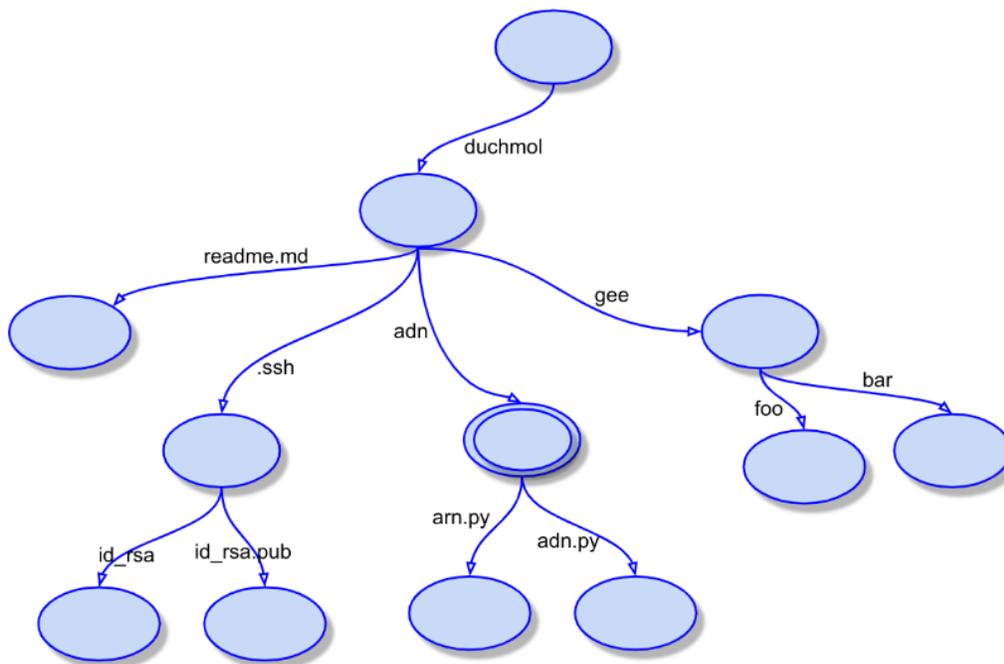
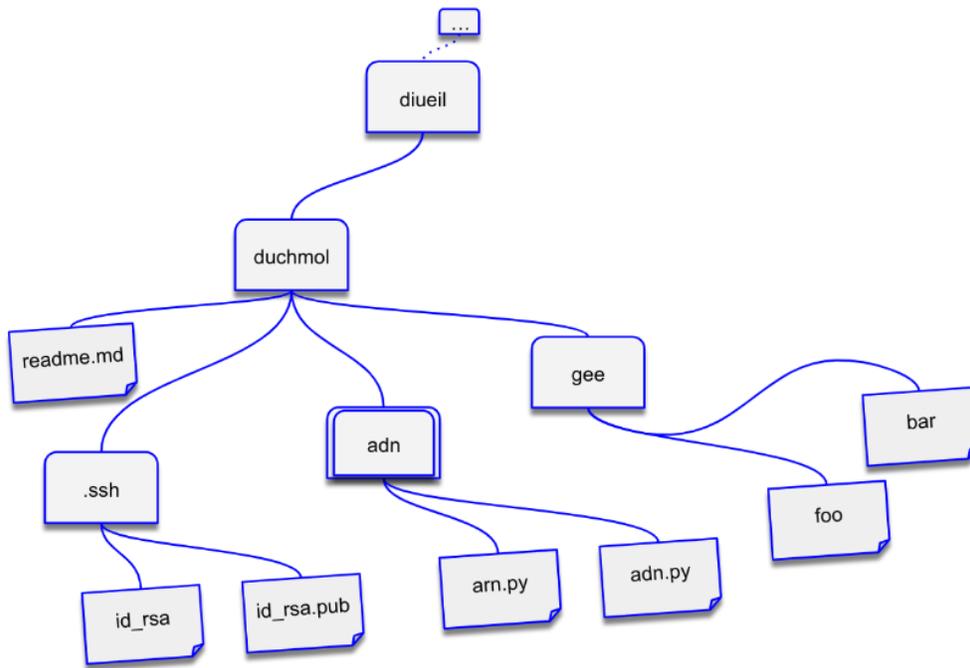
### Nœuds non terminaux

- répertoires
- toujours deux fils : . et ..
- . désigne le nœud lui-même, .. désigne son père

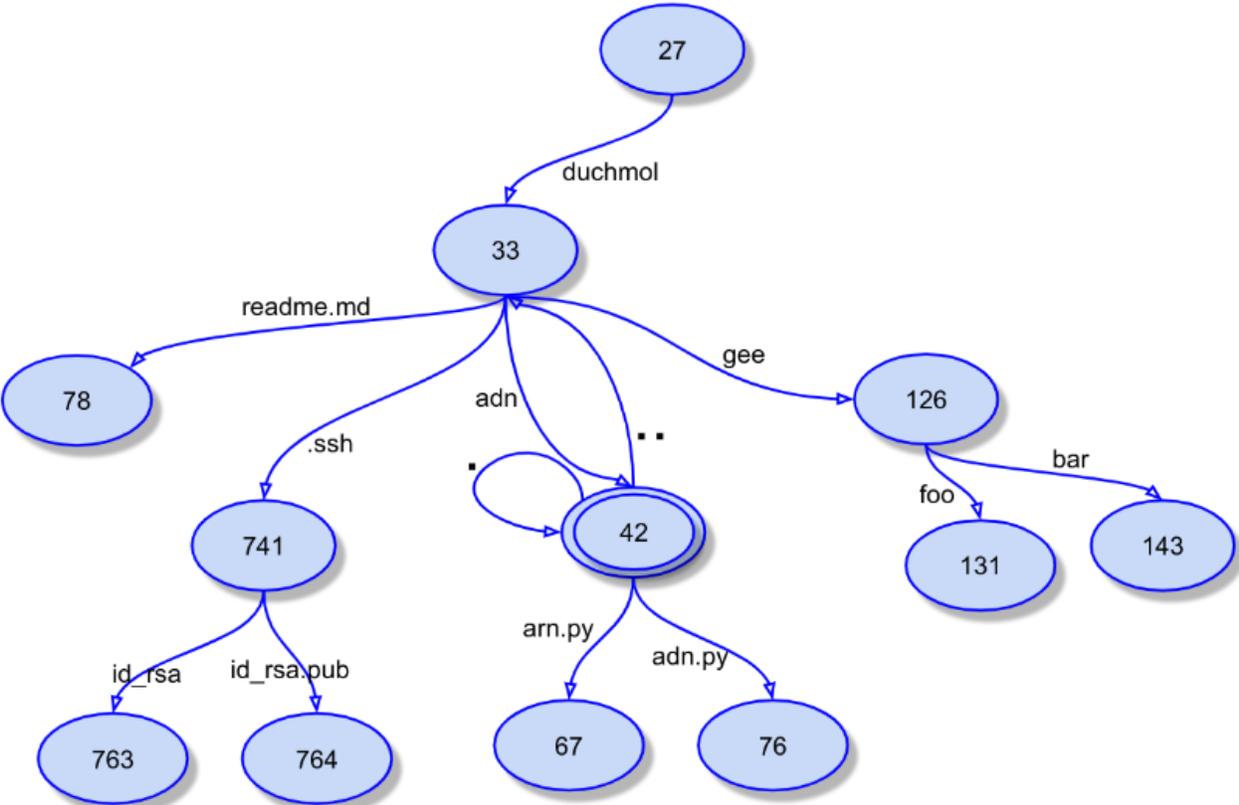
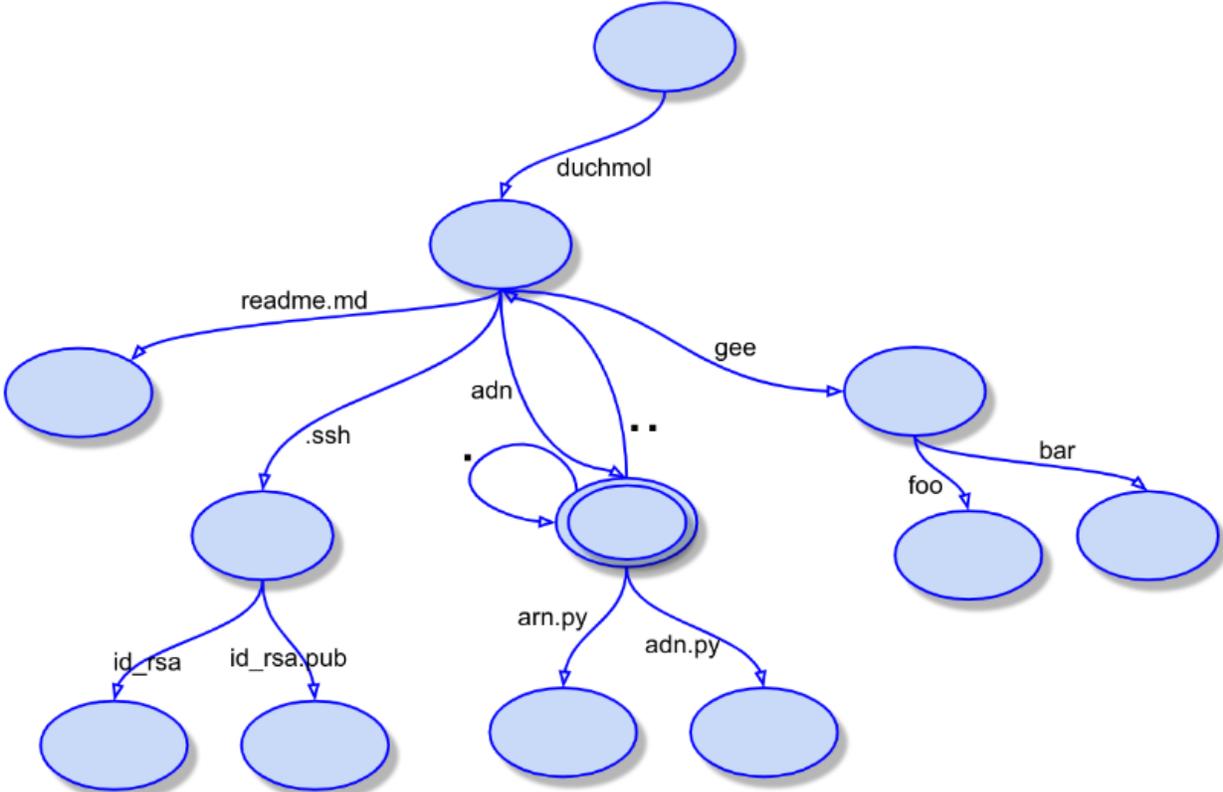
### Nœuds terminaux

- fichiers standard
  - contiennent des données
-

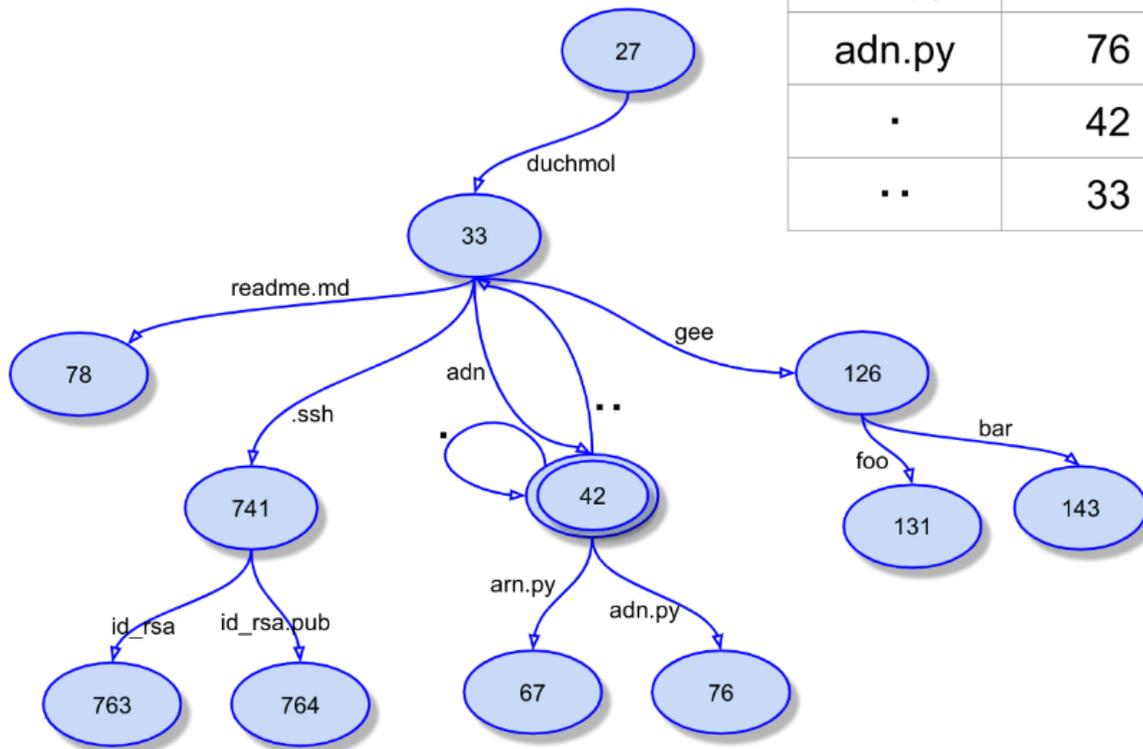
# Système de fichiers, un arbre



Système de fichiers, un arbre, ou presque



<i>nom</i>	<i>numéro</i>
arn.py	67
adn.py	76
.	42
..	33



## Inombre - numérotation des nœuds

### Inœud

- structure de données sur le disque
- informations relatives à un fichier : taille, dates, droits...
- et moyen d'accès aux données

### Désignation d'un fichier sur le support matériel

- numéro d'inœud (*inode*)
- (numéro de périphérique)

### Association d'une numérotation à un nœud

- lien entre le nommage et le contenu

### Nommages multiples d'un nœud

- de part les arcs `.` et `..`
- de part les *liens physiques* (à suivre...)
- accès au même inœud - au même contenu
- partage des modifications du contenu

## Liens multiples I

### Entrées multiples pour un nœud

- plusieurs entrées (arcs)
- d'un même répertoire ou de répertoires différents
- désignent le même nœud

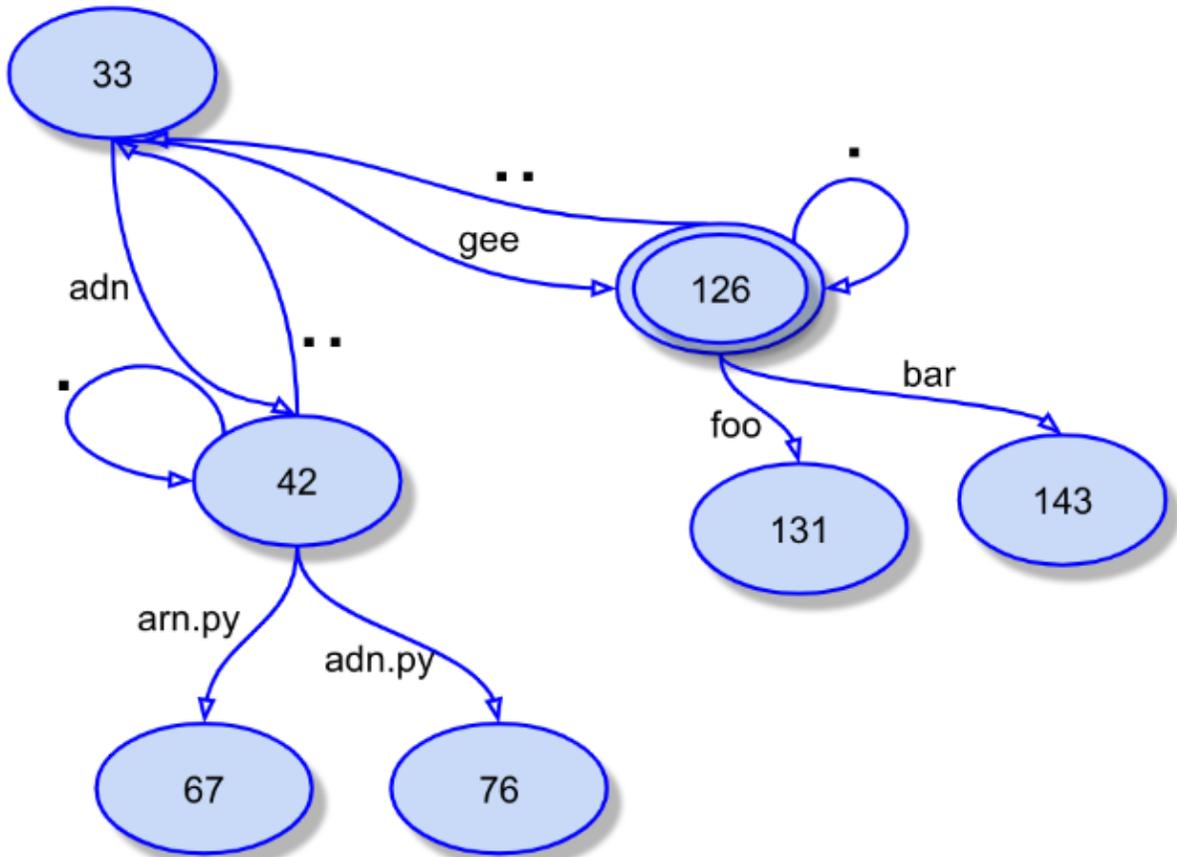
### Lien physique

- ensemble des liens désignant un même nœud
- ensemble des chemins désignant un même nœud

### Non autorisé pour les répertoires

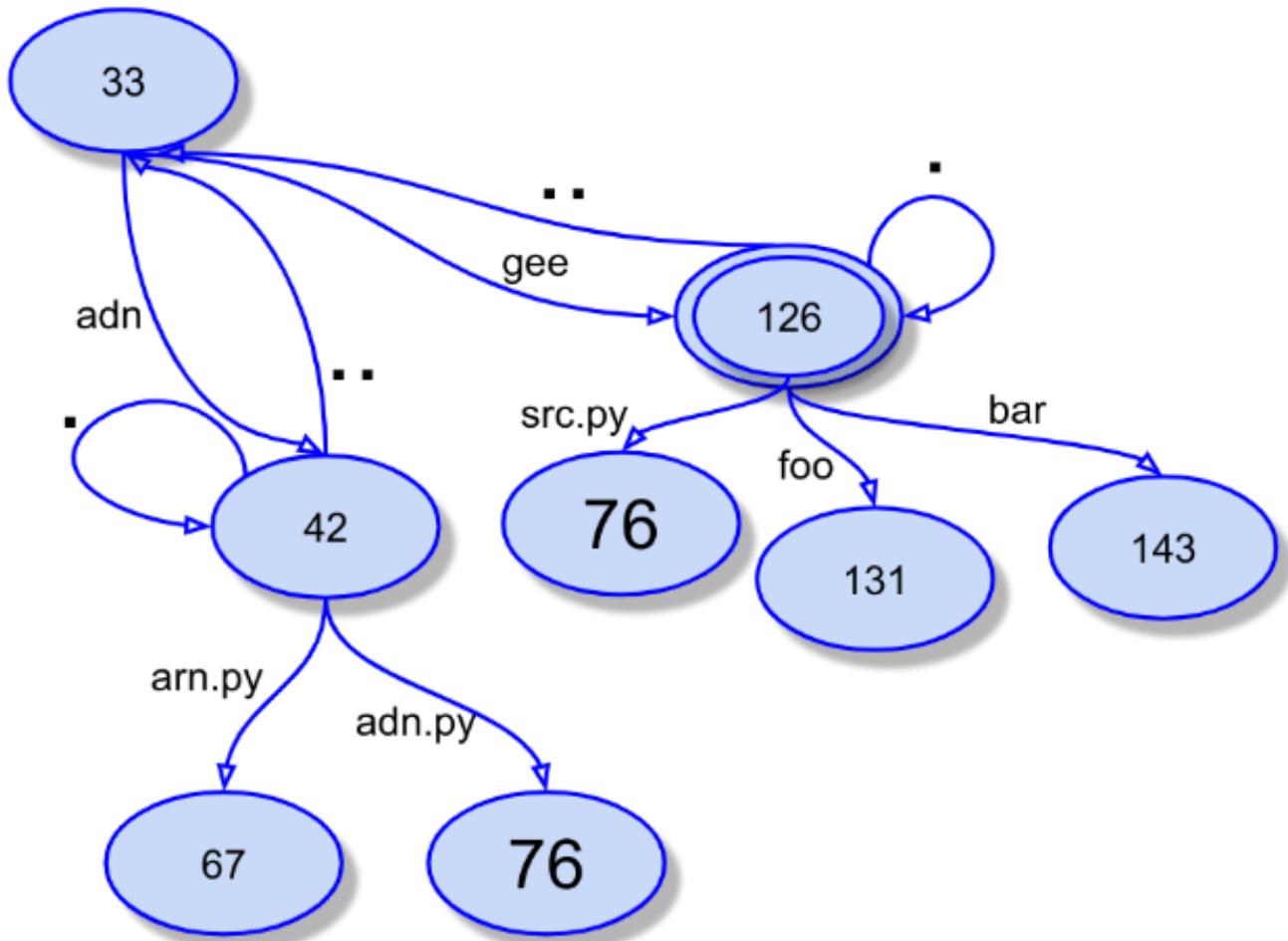
- assurer la cohérence de la hiérarchie

### Liens multiples II



### Liens multiples III

```
ln ../adn/adn.py src.py
```



### Liens multiples IV

```
% pwd
/home/diueil/duchmol/gee
% ls -a ..
.      .ssh      gee
..     adn      readme.md

% ls -li ../adn/
total 16
14255183 -rw-r--r--  1 phm  staff  1858 28 mai 12:21 adn.py
14255577 -rw-r--r--  1 phm  staff   973 28 mai 12:21 arn.py

% ln ../adn/adn.py src.py

% ls -li
total 24
14255207 -rw-r--r--  1 phm  staff    41 28 mai 12:22 bar
14255198 -rw-r--r--  1 phm  staff    31 28 mai 12:22 foo
14255183 -rw-r--r--  2 phm  staff  1858 28 mai 12:21 src.py
```

### Différents types de fichiers

```
% ls -l
```

```
total 8
drwxr-xr-x  4 phm  staff  128 28 mai 12:21 adn
drwxr-xr-x  5 phm  staff  160 28 mai 12:25 gee
-rw-r--r--  1 phm  staff   32 28 mai 12:15 readme.md
```

## Fichiers ordinaires

## Répertoires

## Liens symboliques

## Liens symboliques I

Contient des données = chemin qui désigne un autre nœud

- chemin absolu, ou
- chemin relatif

## Chemin désigné = chemin

- chemin d'un répertoire, ou
- chemin d'un fichier ordinaire

## Interprétation du nom

- le lien symbolique lui-même, ou
- le fichier qu'il désigne
- peut dépendre du contexte d'utilisation
  - % rm symlink
  - % cat symlink

## Liens symboliques II

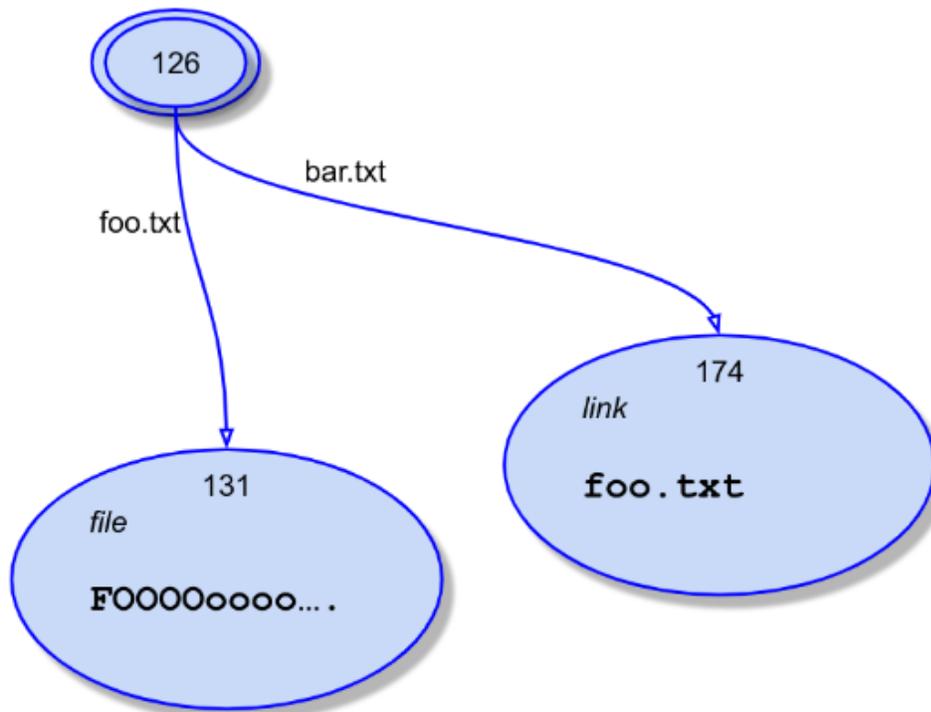
### Création par ln -s

```
% ls
foo.txt
% cat foo.txt
f000000oooo.....
% ln -s foo.txt bar.txt
% ls
bar.txt  foo.txt
% ls -l
total 16
lrwxr-xr-x  1 phm  phm    7 28 mai 23:33 bar.txt -> foo.txt
-rw-r--r--  1 phm  phm   17 28 mai 23:33 foo.txt
% cat bar.txt
f000000oooo.....
```

### Lien symbolique pas toujours valide

```
% rm foo.txt
% cat bar.txt
cat: bar.txt: No such file or directory
```

## Liens symboliques III



## Opérations sur les fichiers

### Informations

- numéro inœud
- type du fichier, taille. . .
- dates. . .
- propriétaire et groupe propriétaire
- droits

### Parcours de la hiérarchie

- listage
- déplacement dans la hiérarchie

### Modification de la hiérarchie

- création, destruction de nœuds
- liens physiques et symboliques

### Écriture et lecture des données des fichiers ordinaires

## Shell

---

*Très et trop brève introduction à UNIX, à l'interpréteur de commandes, sur le portail [gitlab-fil.univ-lille.fr/diu-eil-lil/bloc3/](http://gitlab-fil.univ-lille.fr/diu-eil-lil/bloc3/)*

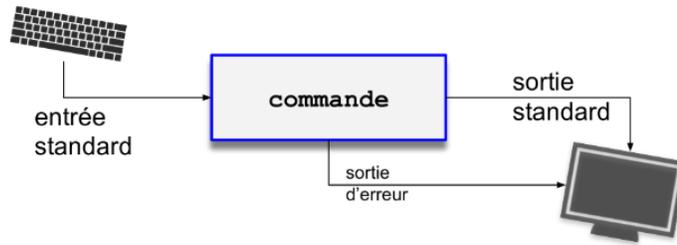
### Précisions

- entrées-sorties, redirection

- substitutions

## Entrées-sorties

% commande [options]... [arguments]...



- création d'un processus qui va exécuter le programme `commande`
  - entrée standard
  - sortie standard
  - associées au terminal
- une telle commande est appelée *filtre*

## Redirection des entrées-sorties

> fichier

- redirige la sortie standard sur le fichier

< fichier

- redirige l'entrée standard depuis le fichier

>> fichier

- concatène la sortie standard au fichier

...

```
% pwd
/home/diueil/duchmol/gee
% pwd > pwd.txt
% ls
pwd.txt
% cat pwd.txt
/home/diueil/duchmol/gee
% ls >> pwd.txt
% cat pwd.txt
/home/diueil/duchmol/gee
pwd.txt
```

...

- possible redirection de la sortie d'erreur avec `2>`, `2>>` et `2>&1`

## cat un éditeur !

```
% cat > foo
Ici je tape le contenu de foo
Je finis par un Control-D en debut de ligne
^D
% cat foo
Ici je tape le contenu de foo
Je finis par un Control-D en debut de ligne
% cat >> foo
```

```
Voici la suite de foo
^D
% cat < foo
Ici je tape le contenu de foo
Je finis par un Control-D en debut de ligne
Voici la suite de foo
%
```

### cat concatène !

```
% cat bar
Bar bar
% cat foo bar >> gee
% cat gee
Ici je tape le contenu de foo
Je finis par un Control-D en debut de ligne
Voici la suite de foo
Bar bar
%
```

### Digression who, wc

Commande who affiche la liste des utilisateurs connectés

```
% who
marquet          tty1      Sep 24  6:39
duchmol          tty3      Sep 29 15:16
marquet          tty2      Sep 24  7:14
```

Commande wc compte les caractères, wc -l les lignes

```
% wc gee
4      23     104 gee
% wc -l < gee
4
```

### Connecter les commandes via les pipes

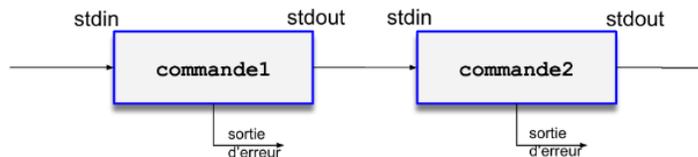
— combien d'utilisateurs connectés ?

```
% who > temp ; wc -l < temp ; rm temp
22
```

### Connecter deux commandes par un « pipe »

```
% who | wc -l
22
```

...



— Forme générale :

```
commande1 | commande2 | ... | commanden
```

— autre exemple

```
% cat foo.txt bar.txt | spell -french | sort > err.txt
```

## Substitutions

### Substitutions réalisées par le shell

- expressions régulières pour les fichiers — `*.py`
- substitution de variables — `$HOME`
- substitutions de commandes — `$(commande)`
- protections `\`, `"`, et `'`

### Expressions régulières pour les noms de fichiers

- `?` — un caractère quelconque (y compris le `.`)
- `*` — tout motif (y compris le vide)
  
- `[list]` — un caractère quelconque de la `list`
- `[lower-upper]` — un caractère quelconque entre `lower` et `upper`

### Substitution *par le shell*

```
% ls *.py
```

- la commande `ls` ne connaît rien de `l*`

```
% echo *.py
```