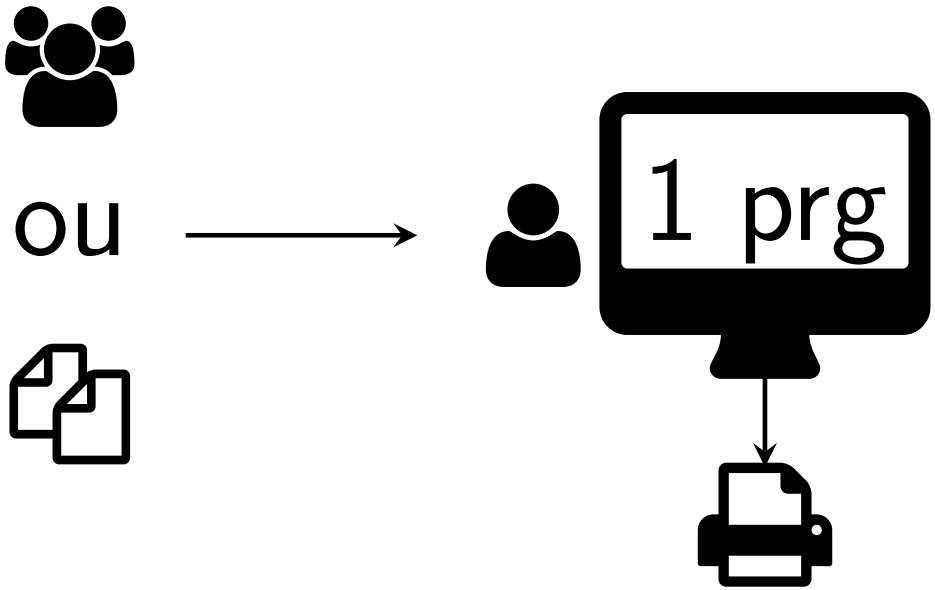
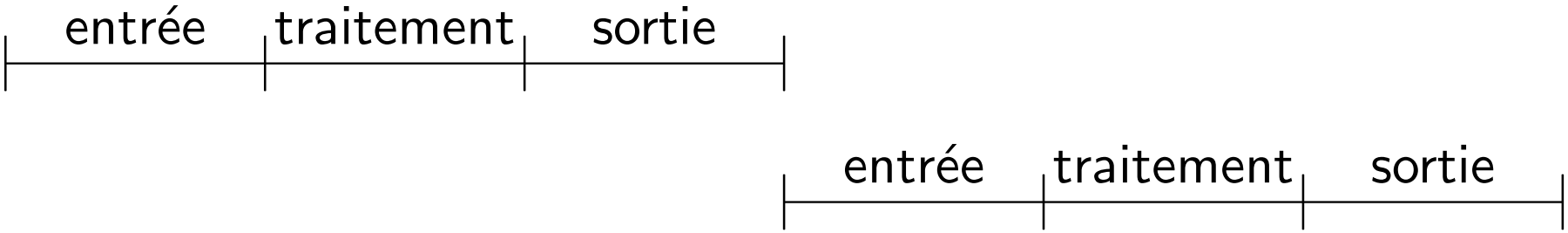
Plan

Évolution des systèmes d'exploitation

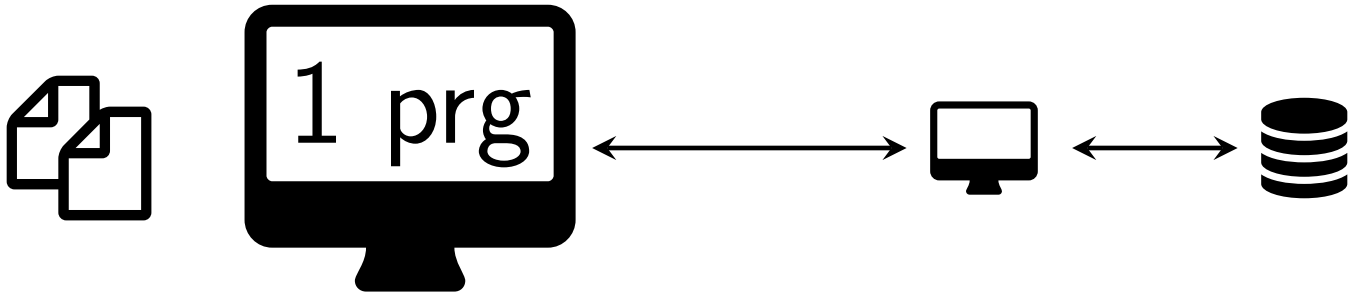
Prémices



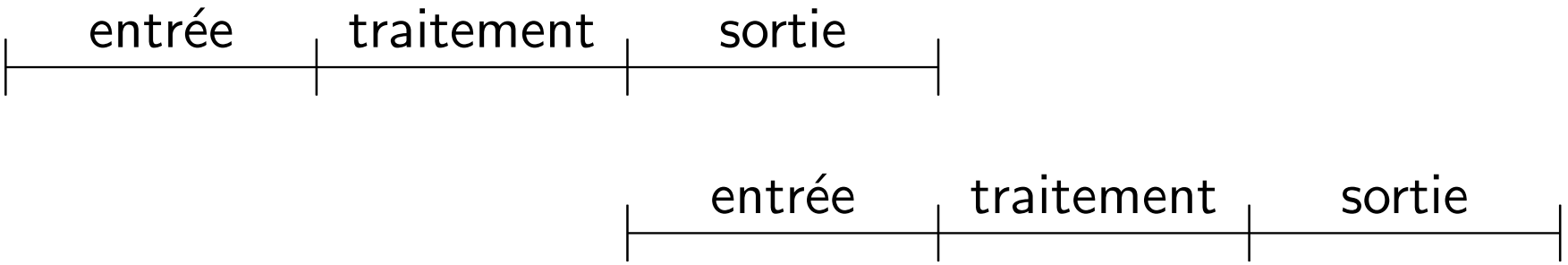


Évolution des systèmes d'exploitation

Traitement par lots (batch)

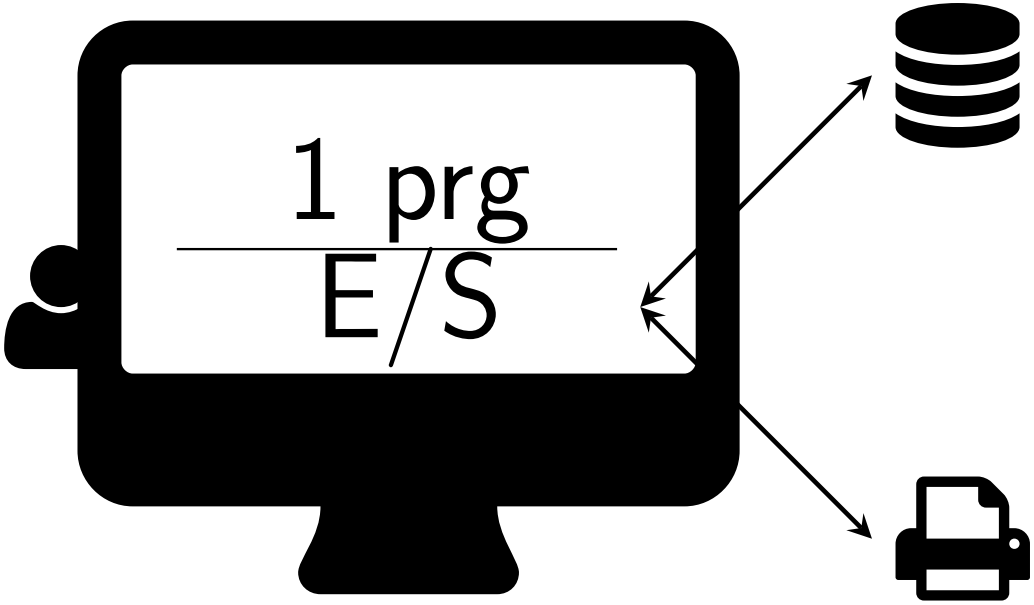


Système par lots

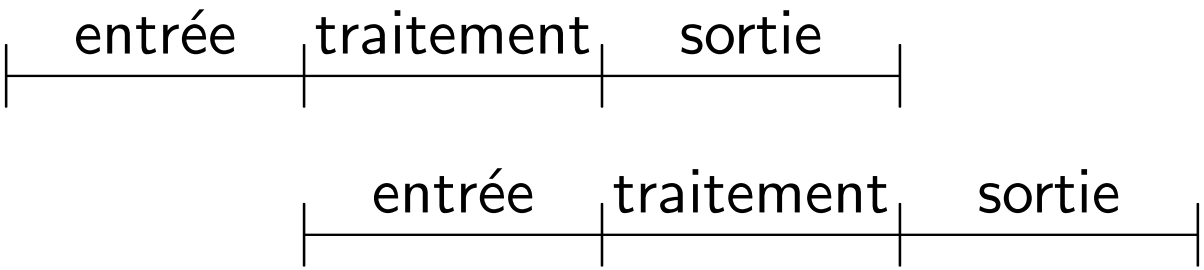


Évolution des systèmes d'exploitation

Indépendance des E/S (1960)



Système avec spooling



Évolution des systèmes d'exploitation

Multiprogrammation (1965-80)

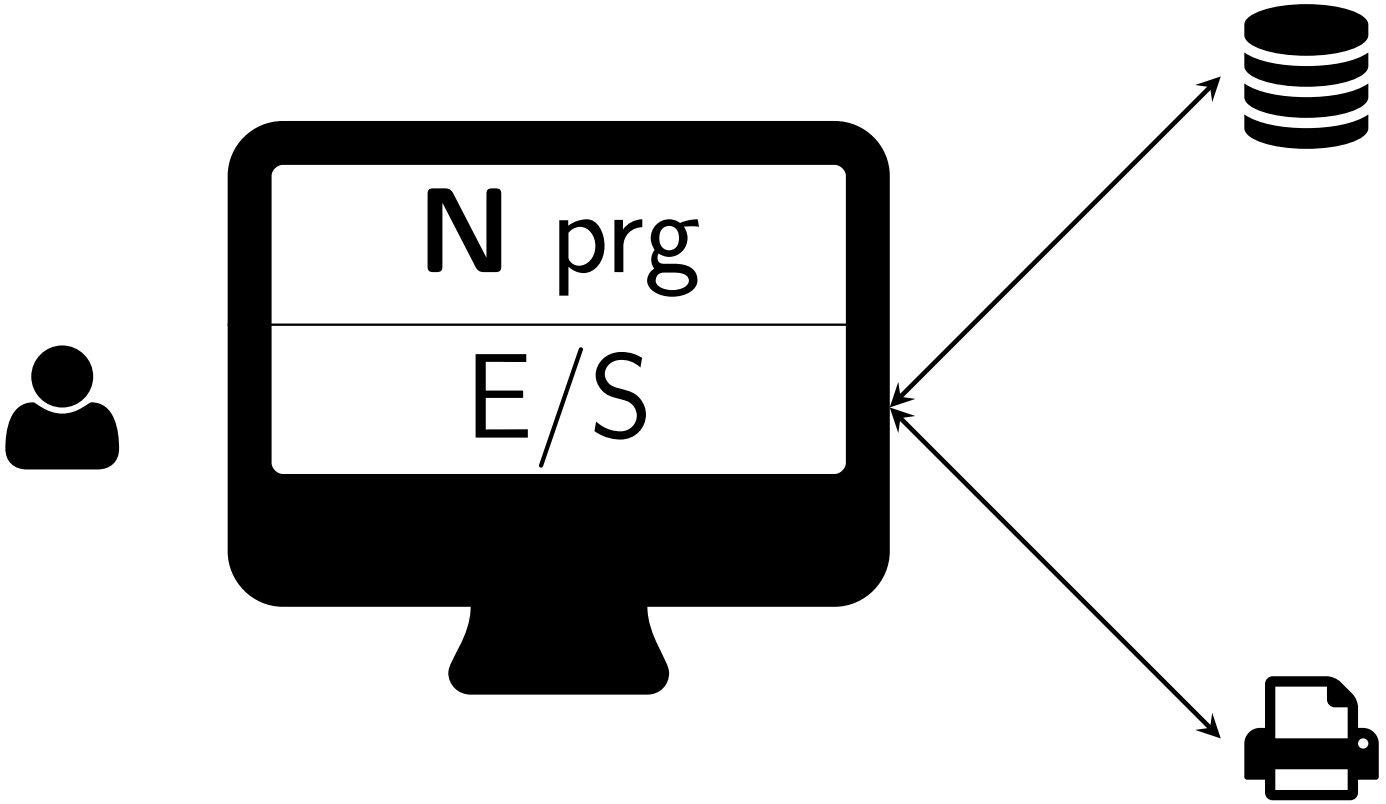


Figure – Système multiprogrammé

Évolution des systèmes d'exploitation

Système à temps-partagé (*time sharing*)

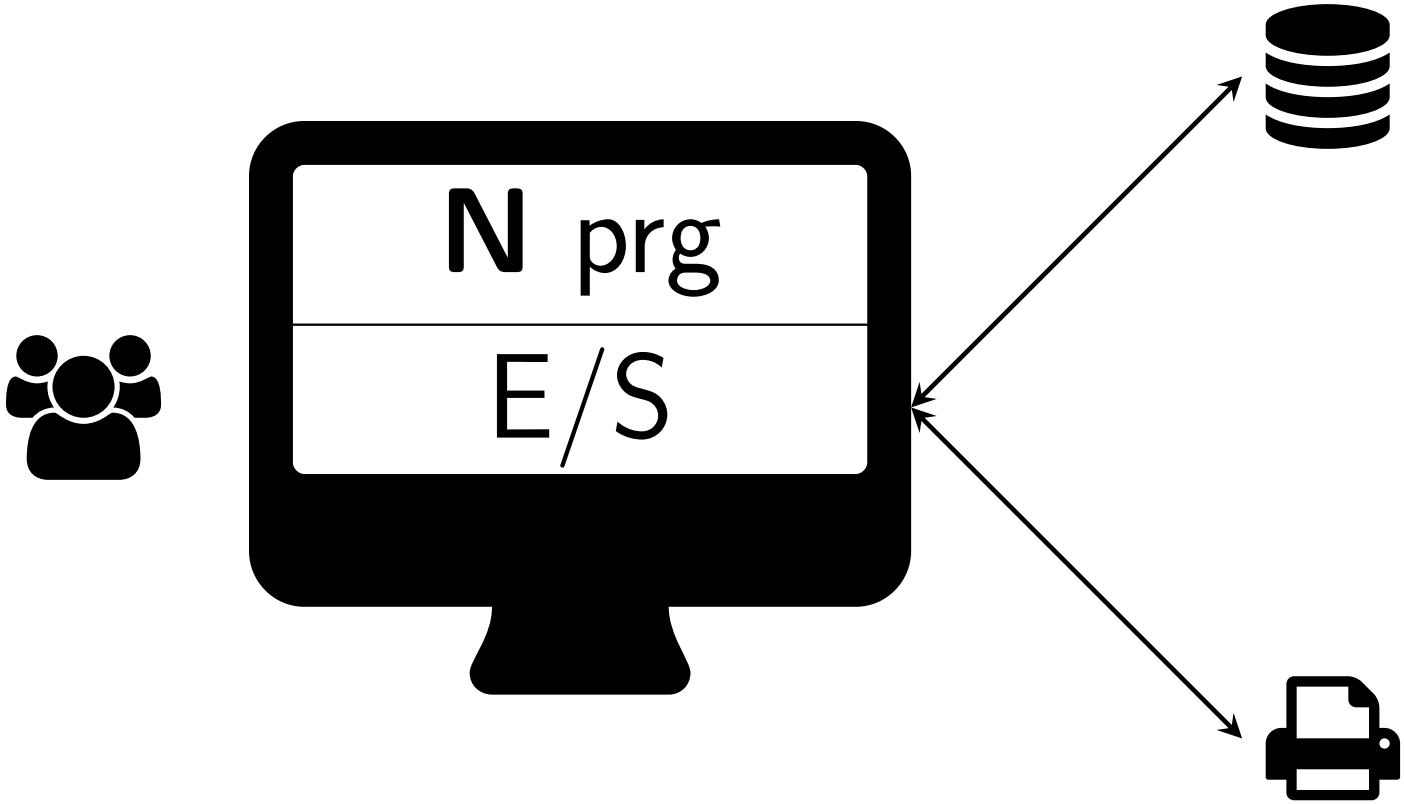


Figure – Système multi-utilisateurs ou temps partagé

Structure d'un système d'exploitation

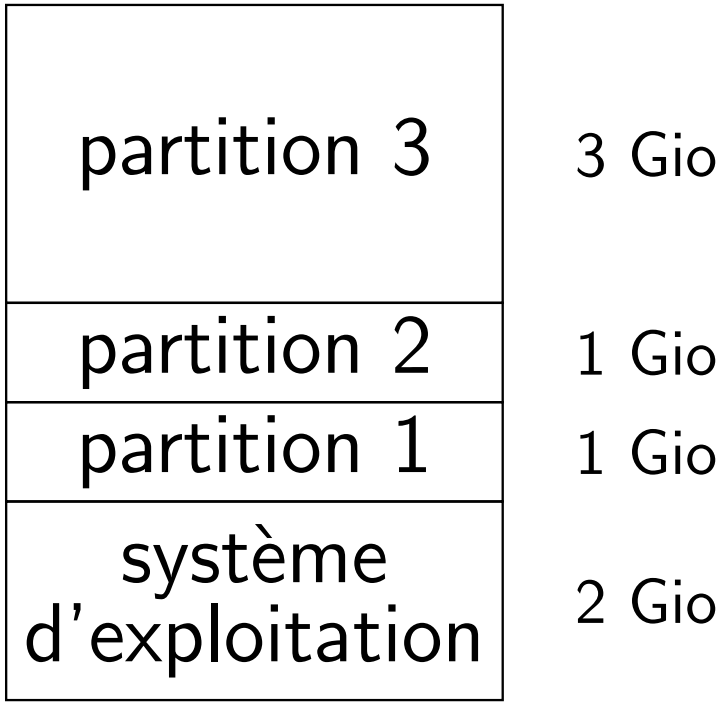
interface utilisateur	mode utilisateur
interprète de commandes	
bibliothèques standards	
OS (gestion des processus, de la mémoire, des entrées/sorties, des fichiers)	mode superviseur

Noyau (*kernel*), mémoire

Les fonctions principales du **noyau** sont :

- l'allocation du CPU,
- la gestion des processus,
- la gestion des interruptions.

Structure de la **mémoire** : partitions de taille fixe



Mémoire

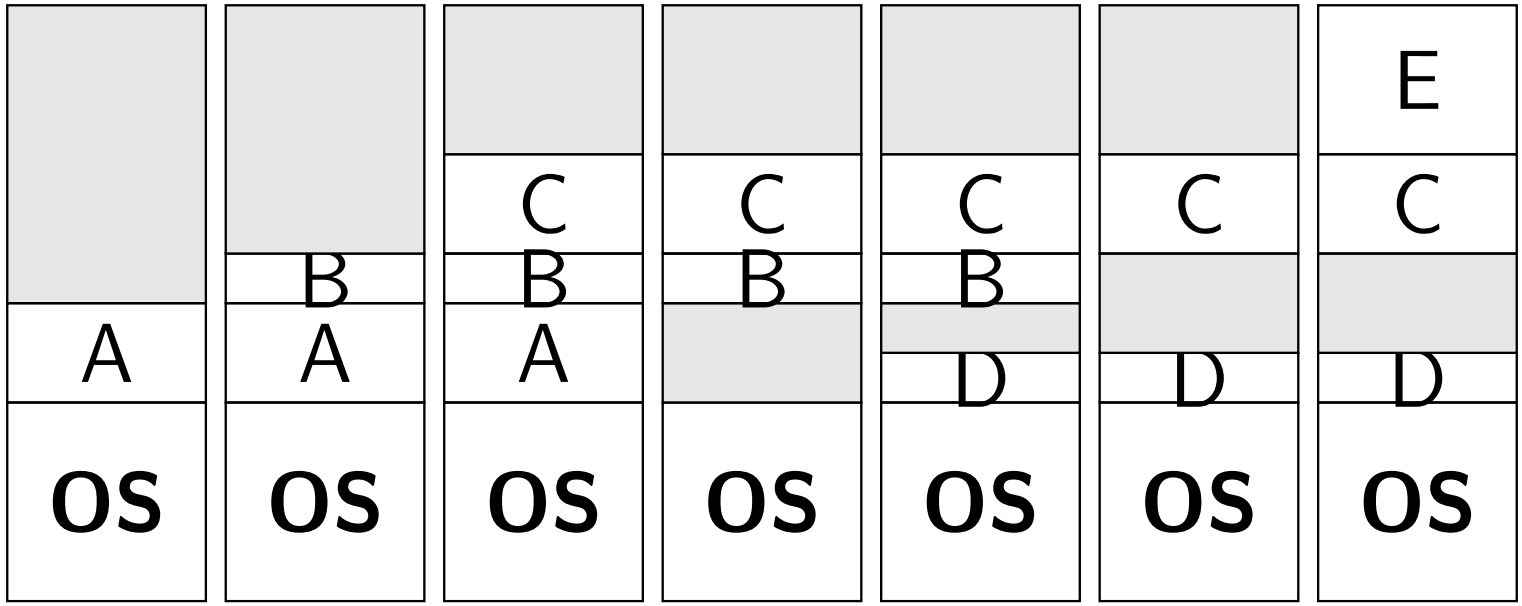
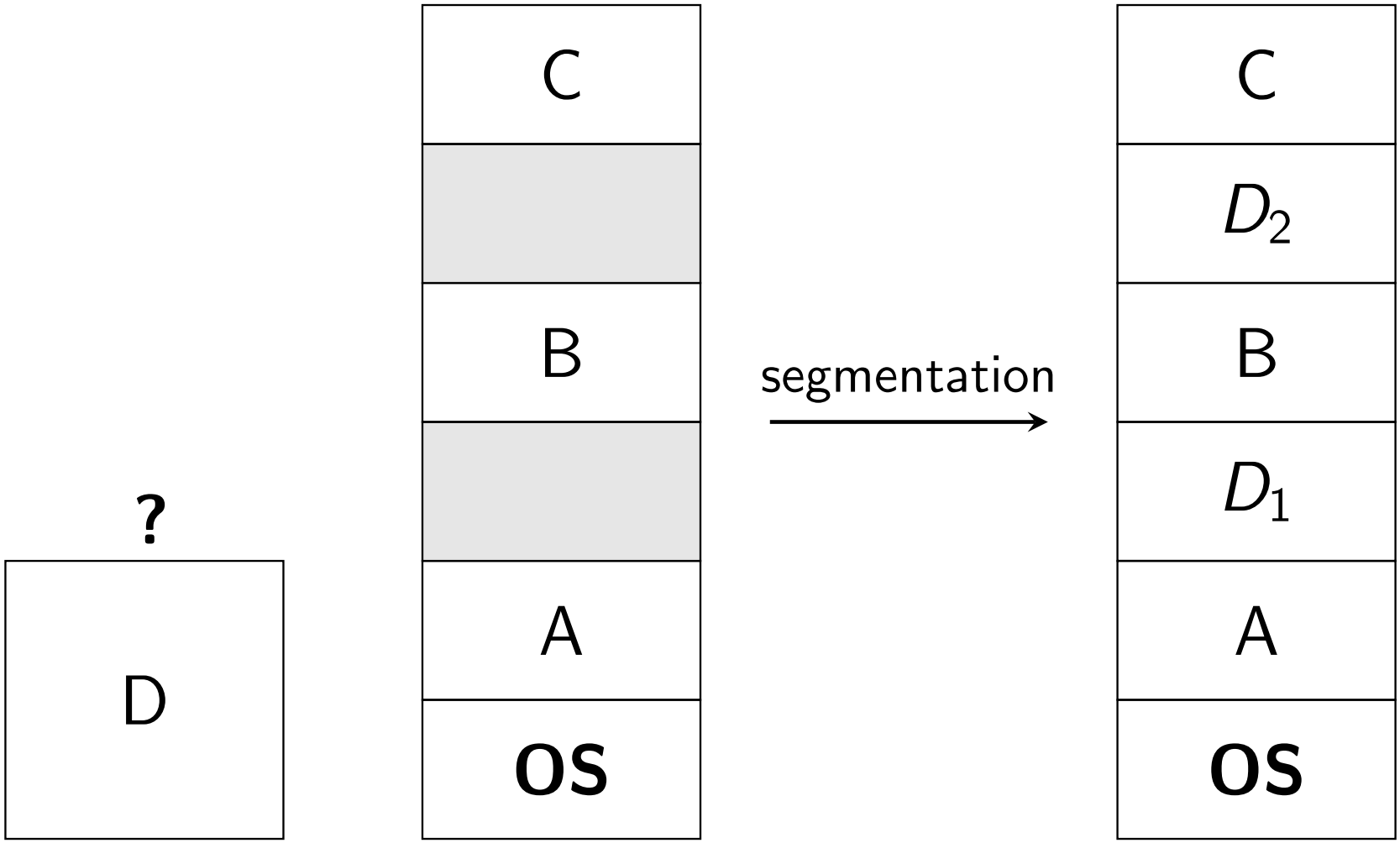


Figure – Partitions de taille variable

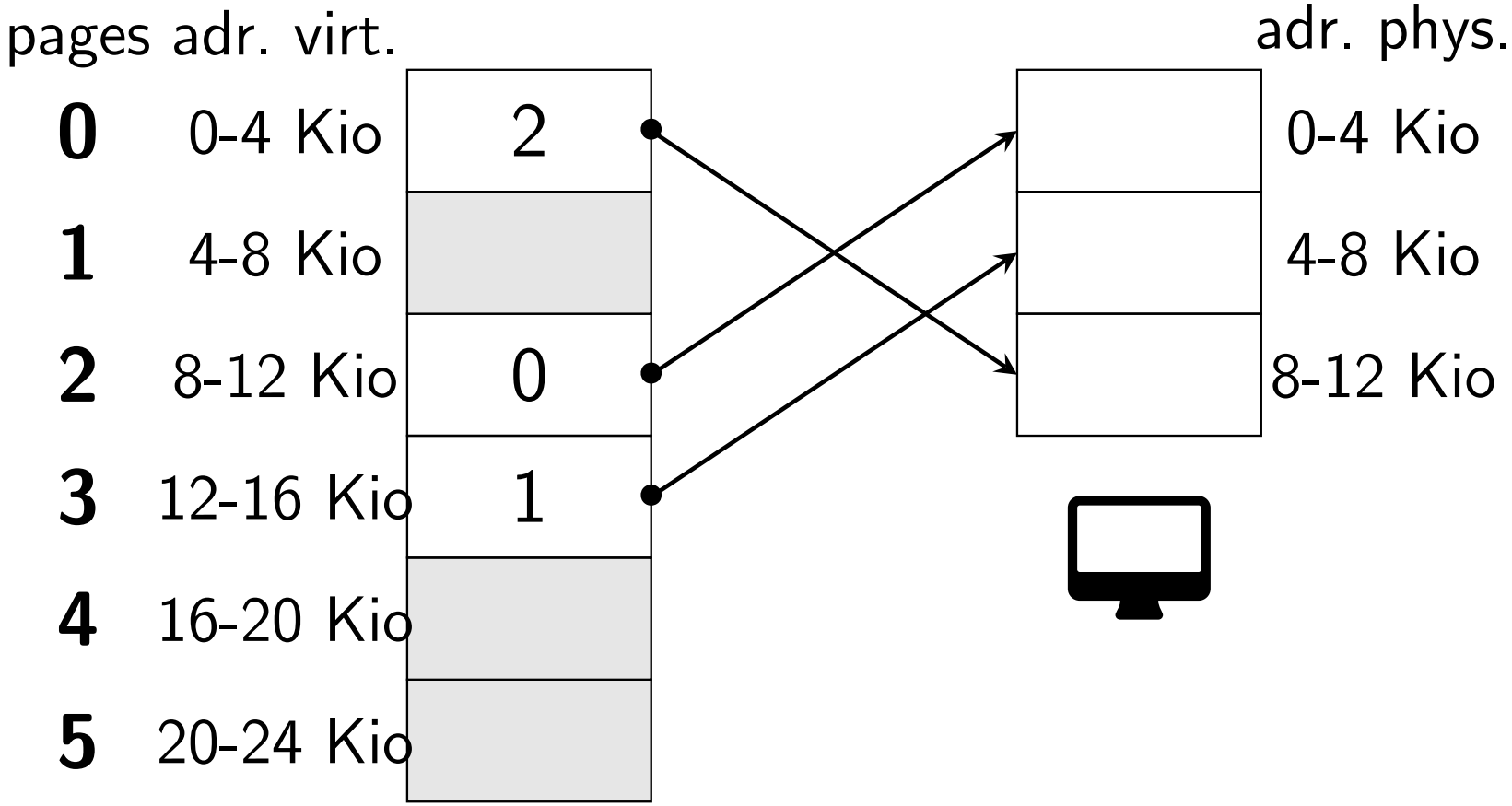
Mémoire

Segmentation

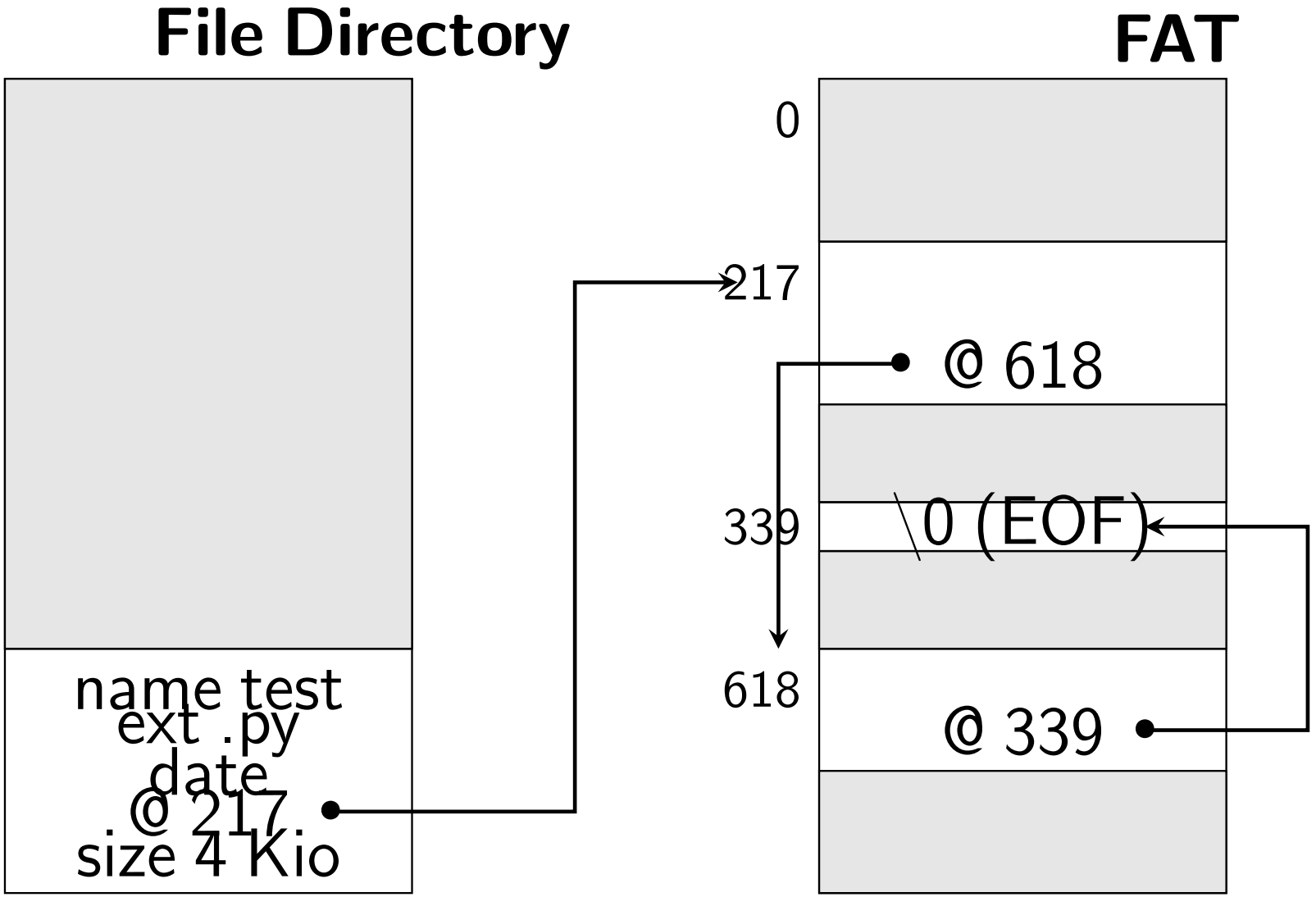


Mémoire

Pagination



Fichiers


File Directory
FAT
name test
ext .py
date
@ 217
size 4 Kio
0
217
@ 618
339
0 (EOF)
618
@ 339

Unix

Unix est un système multi-tâche, multi-utilisateur.

Il possède un langage de commande, le *Shell*.

La documentation est accessible en ligne depuis la commande :

`man <mot-cle>`

Démarrage : le shell est représenté par le caractère $ suivi d'un petit carré clignotant appelé l'invite (*prompt*).

Le système de fichier

Trois **types** :

- les répertoires ou catalogues (sous le nom de *directory*),
- les fichiers spéciaux (*special file*),
- les fichiers ordinaires (*regular file*) tout le reste.

Noms des fichiers : jusqu'à 255 caractères, différence entre majuscules et minuscules, lettres, chiffres, certains caractères spéciaux.

Protections : catégories d'utilisateurs potentiels (propriétaire, membres d'un groupe et les autres utilisateurs), à chaque type d'utilisateur est associé droits de lecture, écriture et exécution.

Le système de fichier

Le propriétaire d'un fichier peut modifier ses droits d'accès par la commande `chmod <qui><permission><operation> <fichier>` :

- `<qui>` : u (user), g (group), o (other), a (all).
- `<permission>` : + (pour autoriser), - (pour interdire).
- `<operation>` : r (read), w (write), x (execute).

Exemple : `chmod og-rw toto.txt`

Abréviations (caractères spéciaux ou jokers) : permettent de désigner des ensembles de fichiers.

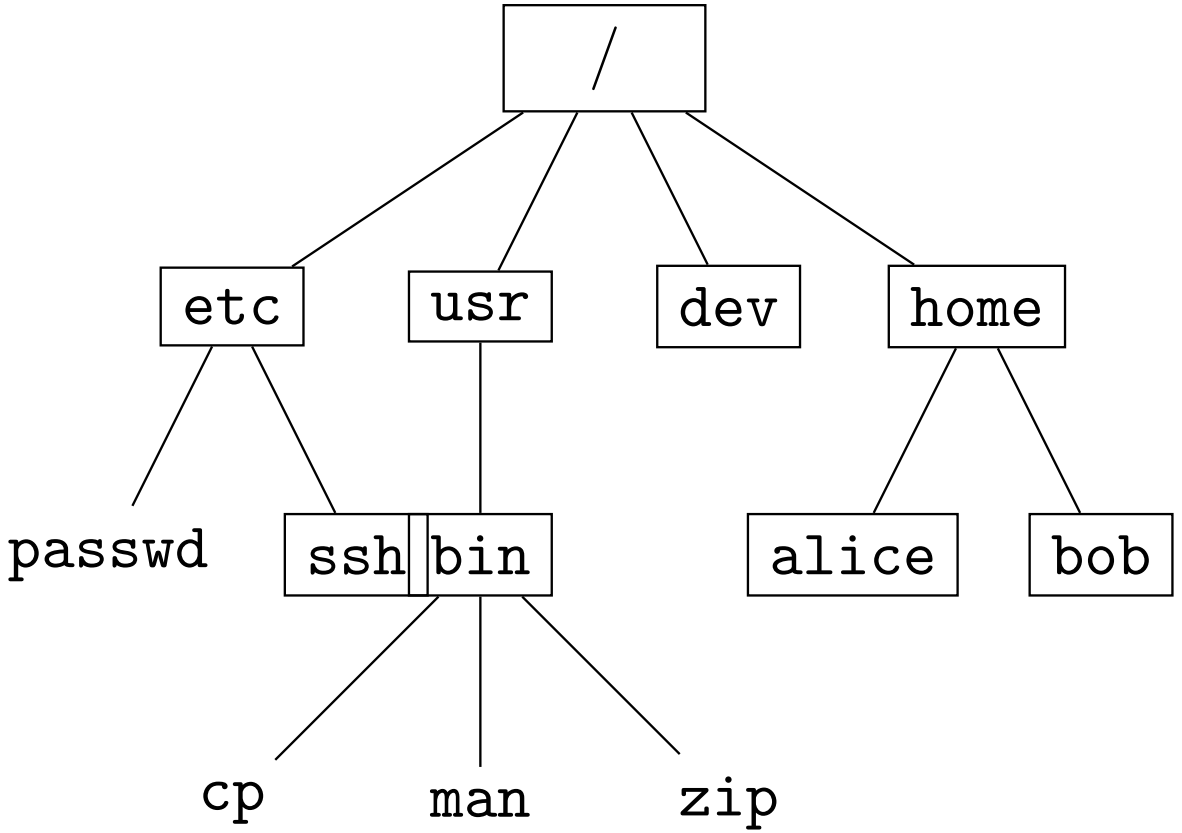
- * : n'importe quelle chaîne de caractères y compris rien.
- ? : n'importe quel caractère.
- [−] : un ensemble de caractère précisé entre crochets.

Commandes de manipulation de fichiers

- **cat**
- **cp**
- **mv**
- **ln**
- **rm**

Les répertoires

Structure arborescente



Un chemin peut être représenté de deux façons différentes :

- absolu : description faite à partir de la racine (/),
- relatif : description faite à partir du répertoire de travail (•).

Commandes associées aux répertoires

- **cd**
- **pwd**
- **ls**
- **mkdir**
- **rmdir**

Les processus

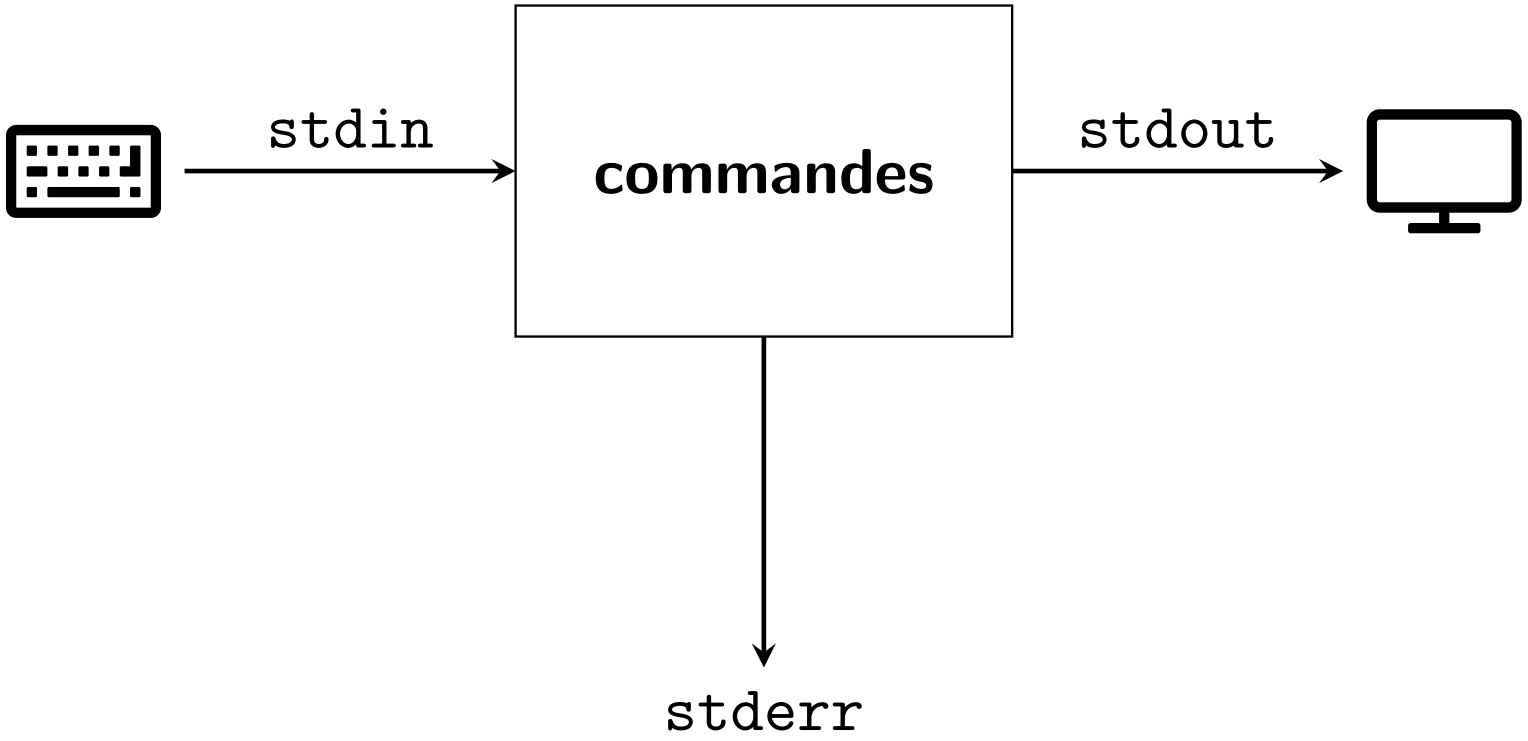
Un processus possède un certain nombre d'attributs.

```
$ ps l
```

FLAGS	UID	PID	PPID	PRI	VSZ	STAT	TTY	TIME	COMMAN
0	500	366	360	0	2744	S	p2	0 :00	bash
0	500	438	366	4	14168	S	p2	0 :43	emacs
0	500	624	366	13	1764	R	p2	0 :00	ps l

- FLAGS : (0 = en mémoire centrale)
- UID : numéro du propriétaire du processus
- PID : numéro par lequel le système connaît le processus
- PPID : processus père
- PRI : priority (grand = faible)
- VSZ : taille du processus
- STAT : state, (s)leep (en attente d'un évènement), (r)un (actif, utilise le CPU) et s(t)opped (processus suspendu)
- TTY : numéro de la console
- TIME : temps d'exécution cumulé du processus
- COMMAND : programme correspondant au processus

Les fichiers standards et leur redirection



Les entrées / sorties standards

Redirection de `stdin` : commande > fichier (stderr : 2>).

Double redirection de `stdin` : commande >> fichier.

L'enchaînement de processus

On peut enchaîner les processus de façon totalement indépendante et en séquence.

Syntaxe : commande1 ; commande2 ; commande3

On peut passer en paramètre d'une commande le résultat d'une autre commande par le caractère spécial ‘.

Exemple : `ls ‘pwd‘`

Il est aussi possible de rediriger la sortie d'un processus dans l'entrée d'un autre processus par le mécanisme du « pipeline ».

Syntaxe : commande1 | commande2 | commande3

L'environnement bash

Les variables d'environnement :

- PATH : contient les noms de répertoires à analyser pour trouver les commmandes à exécuter.
- TERM : contient l'identification du terminal.
- HOME : répertoire initial lorsqu'on vient juste de se connecter.
- PS1 : "l'invite" sur l'écran.
- PS2 : symbole qui visualise une ligne suite (>).
- IFS : les blancs UNIX (blanc, TAB, <CR>).

Initialisation d'une variable par la syntaxe : var = chaîne.
Exemple : `x=coucou` (attention : pas d'espace !).

Les variables s'utilisent sous la forme : $x. Exemple : `echo $x`, affiche sur `stdin` le contenu de la variable x soit `coucou`.

Mécanisme d'inhibition

Permet d'éviter l'interprétation des caractères spéciaux.

« Dépersonnaliser » un seul caractère à l'aide du caractère anti-slash \. Exemple : `echo \?`.

« Dépersonnaliser » un ensemble de caractères à l'aide des apostrophes ' '. Exemple : `echo '***'`.

Quelques commandes

- **tr [option] <chaîne1> <chaîne2>**
- **sort [option] <fichiers>**
- **tail ± <nombre> <fichier>**
- **head [-n] <fichier>**
- **wc**
- **grep**