Introduction aux Systèmes d'Information

TP1: Introduction au système Linux

Sommaire

Sommaire

C’est l’interface entre l’utilisateur et le matériel

Ses fonctions principales sont :

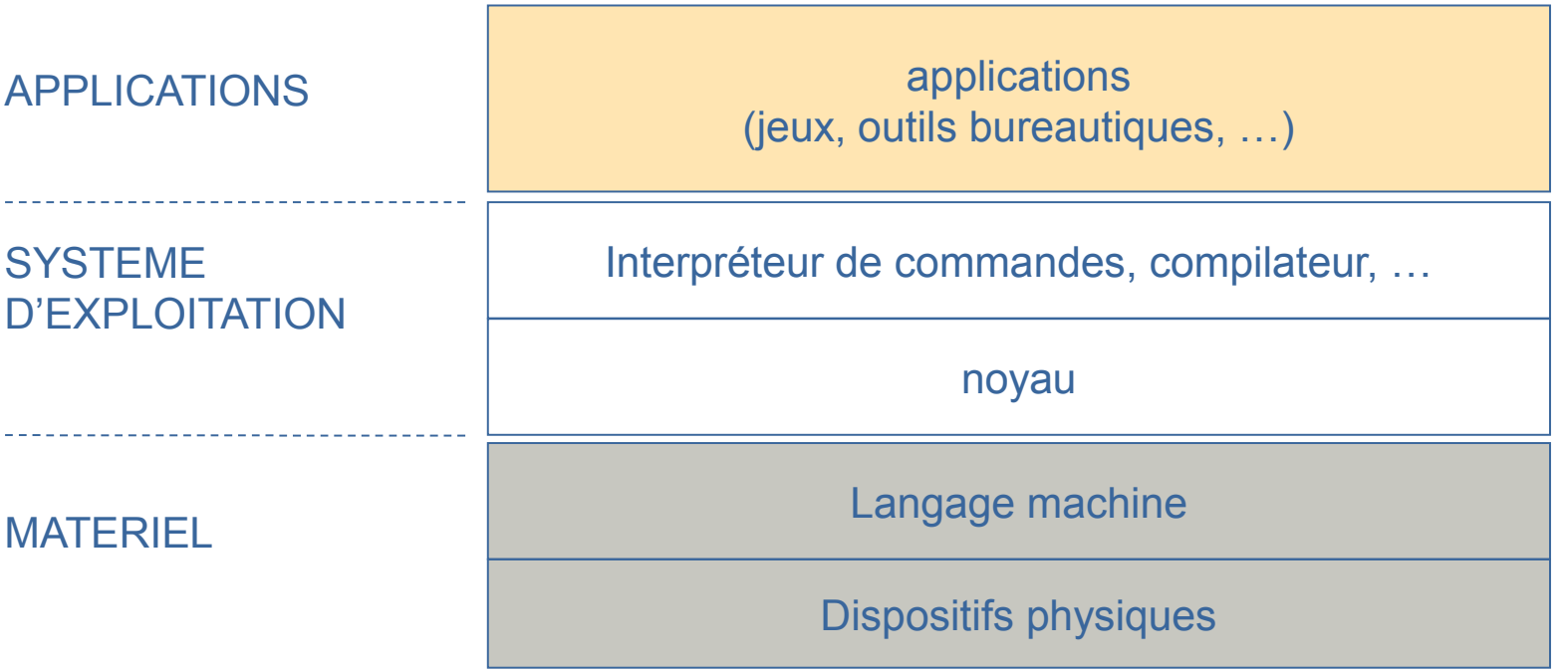
- contrôle des ressources (allocation et gestion du CPU et de la mémoire) ;
- contrôle des processus ;
- contrôle des périphériques ;
- …

Il contient des outils de gestion utilisables par les applications, tels que la manipulation de fichiers, gestion d’impressions, date, …

Exemples:

Unix, DOS, Windows, Mac OS, Linux, OS/2, BSD, ...

•Architecture-type:



Sommaire

Les avantages des systèmes Linux sont :

- le multi-tâches ;
- le multi-utilisateurs ;
- le multi-postes ;
- la liberté ;
- la gratuité !!

CentOS (**Community ENTerprise Operating System**) est une distribution GNU/Linux principalement destinée aux serveurs et dont tous les paquets sont compilés à partir des sources de RHEL (**Red Hat Enterprise Linux**).

Depuis janvier 2012, c'est la seconde distribution la plus utilisée (27,5 %) sur les serveurs web, derrière Debian (32,6 %) et devant Ubuntu (21,9 %).

CentOS

Sommaire

Ouverture/Fermeture d'une session :

Travailler sous Linux implique une connexion au système que l'on appelle **Login**.

Pour cela vous avez besoin :

- Identification de l'utilisateur: *login* + *mot-de-passe ;*
- droits accordés par le *super-utilisateur* (**root**).

Une fois votre travail terminé, le logout permet de se déconnecter de la machine.

Il ne faut **PAS ETEINDRE** une machine "sauvagement".
Utilisez les commandes :

- « ***exit*** » ou ***ctrl + D*** en ligne de commande (CLI) ;
- « **logout** » dans l'interface graphique.

Le Shell :

Une fois connecté, le système ouvre une session à notre nom et attend nos instructions via un interpréteur de commande appelé « ***Shell*** ».

C'est une interface utilisateur “de base” qui interprète ligne à ligne les commandes

Il en existe plusieurs : sh, csh, tcsh, bash, ksh, zsh, …

Il existe des fichiers commençant par '.' et qui nous permettent de le configurer (fichiers d’environnement) :

- “**`.login`**” ;
- “**`.logout`**” ;
- “**`.bashrc`**”.

Format des commandes :

commande [-option(s)] [argument(s)]

Respectez la casse et les espaces!!

Format des commandes :

Exemples:

- **`date`**
- **`whoami`** **:** affiche le nom de l'utilisateur connecté
- **`echo`** : affiche un message (`echo "bonjour !"`)
- **`ls`** **:** liste le contenu d'un répertoire
- **`man <cde>`** **:** manuel en ligne

Format des commandes :

```
[root@localhost ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog
[root@localhost ~]#
[root@localhost ~]# whoami
root
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# date
mar. nov. 12 21:35:38 CET 2013
[root@localhost ~]#
[root@localhost ~]# _
```

Caractères spéciaux :

! ^ * ? [] ' \ ; & < > | >>

- L'astérisque ou étoile : *

interprété comme toute suite de caractères alphanumériques utilisés avec précaution (commande rm par ex...)

- Le point d'interrogation : ?

remplace 1 seul caractère alphanumérique

Caractères spéciaux :

• Le point-virgule: ;

séparateur de commandes

• Les crochets: []

remplacent un caractère choisi parmi ceux énumérés entre les crochets

• L'anti-slash: \

inhibe la signification du méta-caractère qui suit

Caractères spéciaux :

- Interprétation des chaînes de caractères

Texte entre 'mon_texte' (simples quotes): le texte n'est pas interprété mais considéré comme un mot

Texte entre "mon_texte" (doubles quotes): seuls sont interprétés les métacaractères $, \ et `

Texte entre `ma_commande` (anti quotes): considéré comme une commande à interpréter, et c'est le résultat qui sera utilisé.

Caractères spéciaux :

- Exemples:
  - ls *

Tous les fichiers sauf ceux dont le nom commence par un point

  - ls *c

Tous les fichiers dont le nom se termine par un ‘c’

  - ls .*

Tous les fichiers dont le nom commence par un point

  - ls [0-9]*

Tous les fichiers dont le nom commence par un chiffre

Caractères spéciaux :

```
[root@localhost test]# ls [0-9]*
1file  9file
[root@localhost test]# ls c*
cfile
[root@localhost test]# ls .*
.file

.:
1file  9file  cfile

..:
anaconda-ks.cfg  install.log  install.log.syslog  test
[root@localhost test]# _
```

Sommaire

Stocke les données:

- de façon hiérarchique ;
- structure arborescente ;
- **TOUT** est fichier ;

3 types de fichiers:

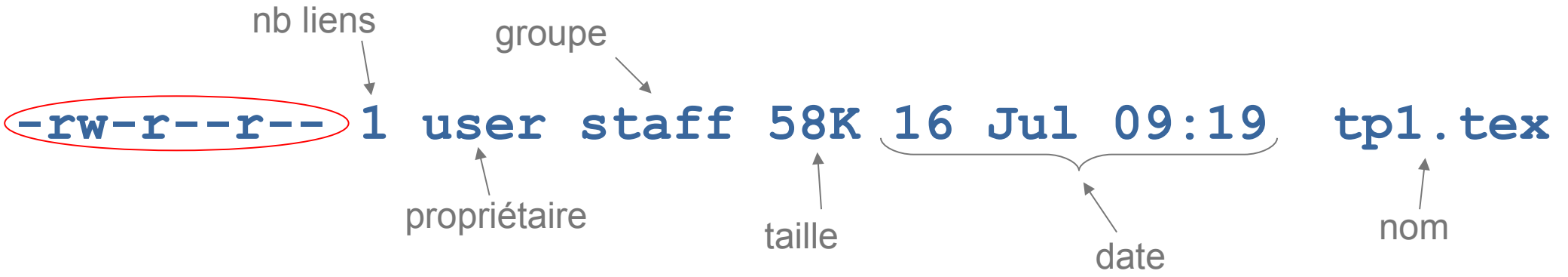
- fichiers ordinaires ;
- répertoires ;
- fichiers spéciaux (périph., …) ;

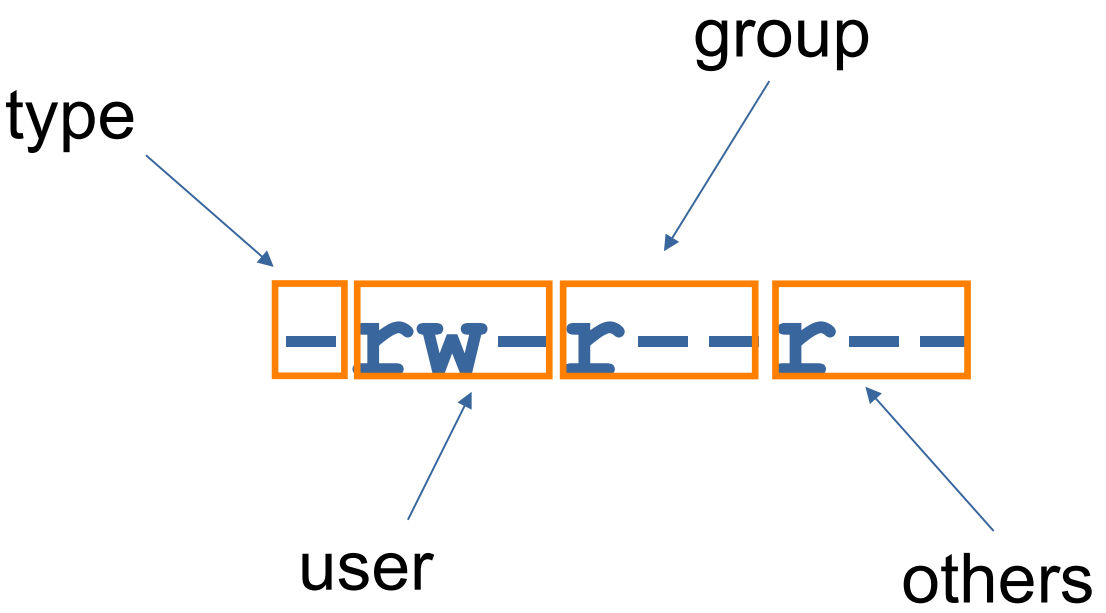
Accès aux fichiers réglementé (sauf: pour root)

- 3 types de contrôle d'accès :
  - propriétaire (user) ;
  - personnes du même groupe (group) ;
  - les autres (others)
- 3 types de permissions

lecture (**r**)	afficher le contenu	afficher le contenu
écriture (**w**)	modifier	créer / supprimer des fichiers
éxécution (**x**)	exécuter	traverser
	fichier	répertoire

Affichage des caractéristiques: **ls -l**





Changer les permissions: **chmod**

```
chmod <classe op perm, ...>|nnn <fichier>
```

Classe	**Opération**	**Permission**
User → u	affectation **=**	lecture (**r**)
Group → g	suppression **-**	écriture (**w**)
Other → o	ajout **+**	exécution (**x**)
All → a		

Chaque permission équivaut à une valeur :

On définit les permissions par addition pour chaque classe.

Permission	Valeur
lecture (**r**)	4
écriture (**w**)	2
exécution (**x**)	1

Pour attribuer des droits sur un fichier, on se pose deux questions :

- Quel niveau de permission ?
- Pour qui ?

Exemple :

Pour attribuer les droits en lecture / écriture sur le fichier toto.txt au propriétaire et au groupe :

- un premier 6 pour le propriétaire → r+w <=> 4+2 = 6
- un deuxième 6 pour le groupe → r+w <=> 4+2 = 6
- un troisième 0 pour les autres → rien <=> 0

Ce qui donne :
chmod **660** toto.txt

On peut également écrire (moins pratique) :
chmod **u=rw,g=rw** toto.txt

Manipulation des fichiers :

- copier

 # cp fichier_source fichier_destination

- déplacer / renommer

 # mv fichier_source fichier_destination

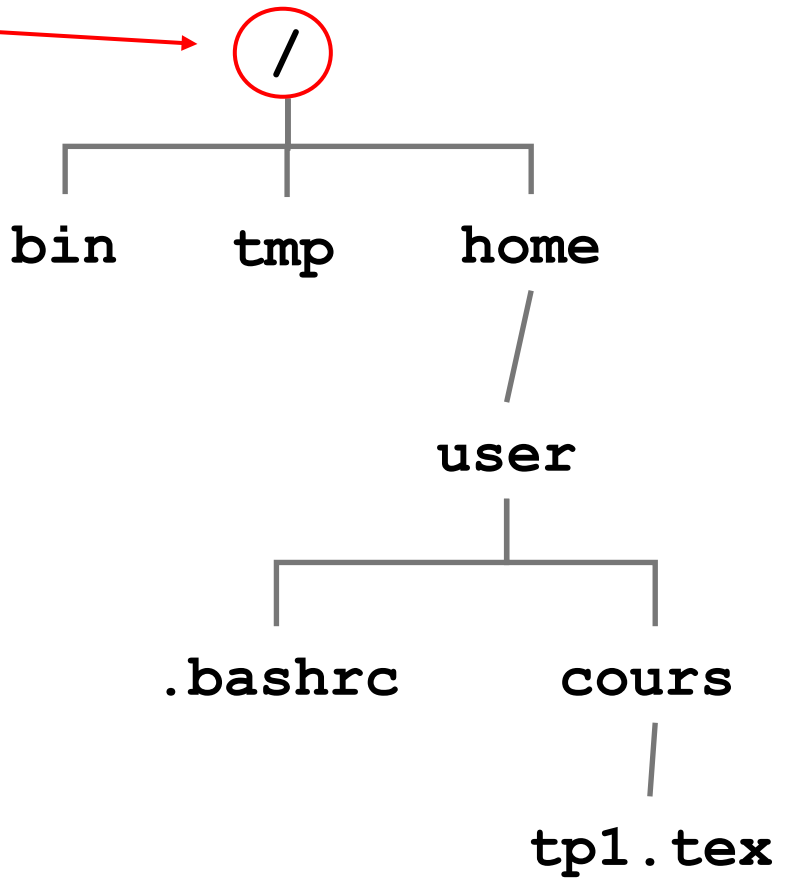
- effacer

 # rm fichier

- afficher le contenu

 # cat fichier

répertoire racine



- le répertoire de login: **~**
- le répertoire courant: **.**
- le répertoire supérieur: **..**
- connaître le rép. courant: **pwd**
- lister le contenu: **ls**
  (voir "**man ls**")

Chemin d'accès au fichier **tp1.tex**:

- **/home/user/cours/tp1.tex**
- **~/cours/tp1.ex**

pwd retourne: **`/home/user/cours`**

Se déplacer grâce à « ***cd*** »:

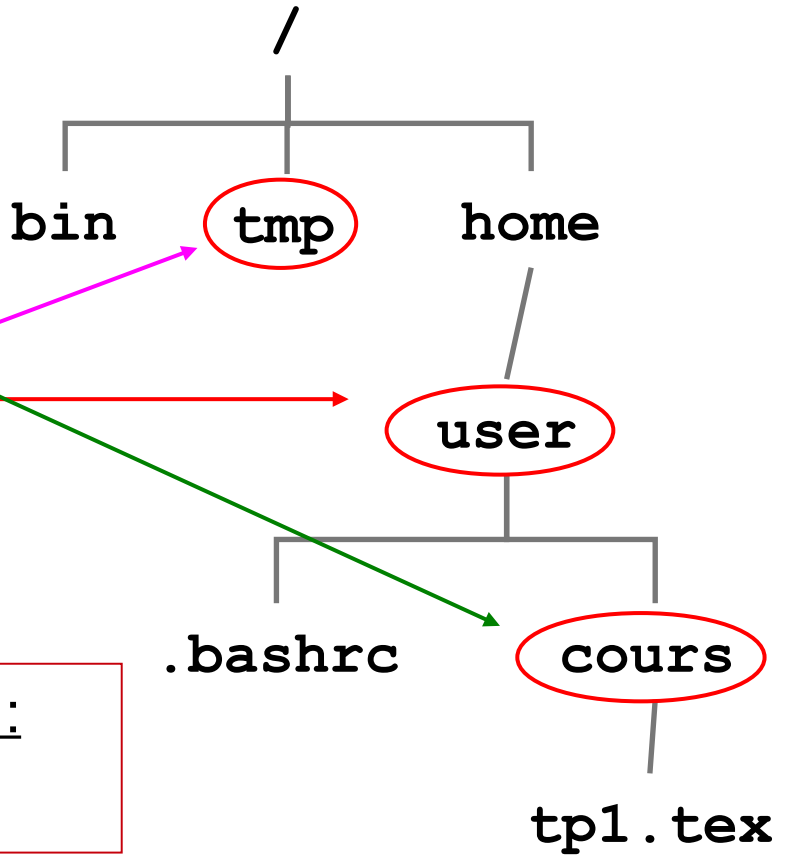
[/home/user/cours]#

[/home/user/cours]# cd ..

[/home/user]#

[/home/user]# cd /tmp

[/tmp]#



Créer un répertoire grâce à « ***mkdir*** » :

[/tmp]# mkdir buzz

Supprimer un répertoire grâce à « ***rmdir*** » :

[/tmp]# rmdir buzz

Se déplacer grâce à « ***cd*** »:

[/home/user/cours]#

[/home/user/cours]# cd ..

[/home/user]#

[/home/user]# cd /tmp

[/tmp]#

répertoire courant

chemin relatif

chemin absolu

Les liens :

- **Liens physiques**

  `ln <nom_fic> <nouveau_nom_fic>`

  - permet de donner plusieurs noms à un fichier
  - pas pour les répertoires
  - ne traverse pas les partitions
  - un fic est détruit quand TOUS ses liens physiques sont supprimés (≠ raccourcis)

- **Liens symboliques**

  `ln -s <nom_fic> <nouveau_nom_fic>`

  - crée un **raccourci**
  - traverse les partitions
  - fonctionne aussi pour les répertoires

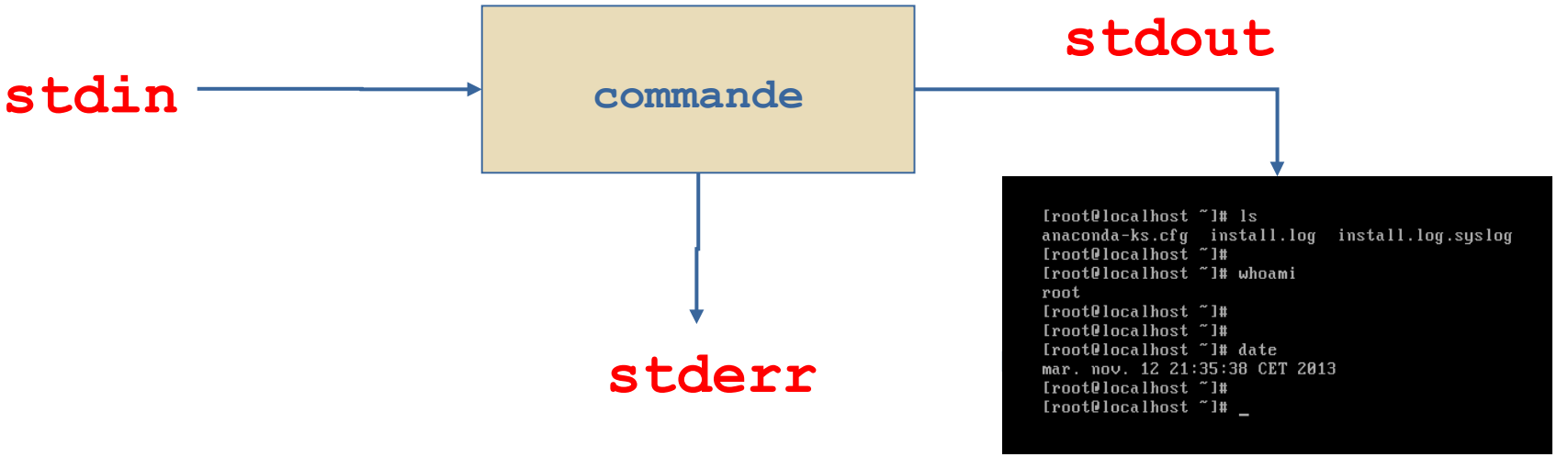
- **Lister les liens d'un fichier:**

  `ls -l <nom_fic>`

Sommaire

Les redirections :

•Une commande ouvre 3 descripteurs de fichiers par défaut :



Redirections= remplacer les canaux par défaut, rediriger vers une autre commande ou un fichier

Les redirections :

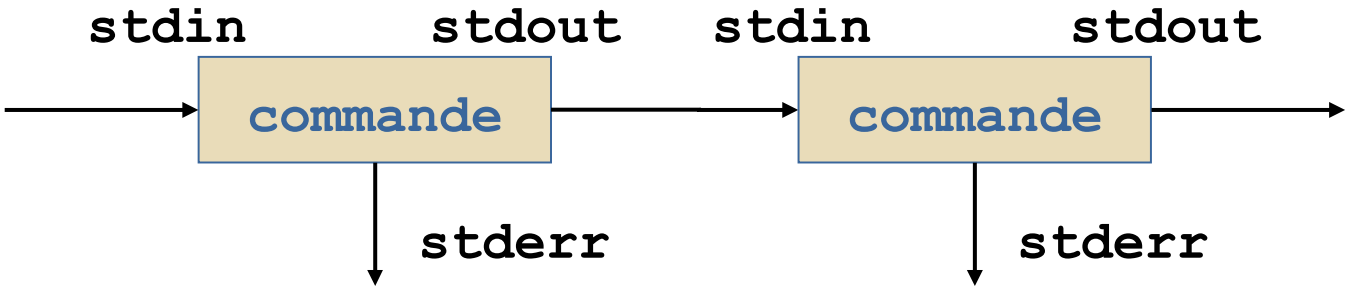
<	redirige l'entrée standard
>	redirige la sortie standard
>>	concatène la sortie standard
2>	redirige la sortie d'erreur
&>	redirige la sortie standard **et** la sortie d'erreur

exemples:

`ls . > liste`	crée/écrase le fichier liste et y dirige la sortie de 'ls'
`date >> liste`	ajoute à la fin du fichier liste la sortie de 'date'
`wc -l < liste`	envoie comme entrée à la commande 'wc' le fichier liste

Les tubes « | » :

Pour “connecter 2 commandes”



ex: combien de fichiers dans le répertoire courant ?

sans pipe:
ls > temp ; wc -l < temp ; rm temp

avec un pipe:
ls | wc -l

Sommaire

Processus = objet dynamique qui représente un programme en cours d'exécution et son contexte

Caractéristiques:

- identification (pid) ;
- identification du processus parent (ppid) ;
- propriétaire ;
- priorité ;
- …

Pour voir les processus en cours: ***ps***

Infos retournées par « ***ps*** » :

Numéro du processus

temps CPU utilisé

```
[centos:~] ps
  PID  TT  STAT      TIME COMMAND
 3899  p1  S      0:00.08 -zsh
 4743  p1  S+     0:00.14 emacs
 4180 std  S      0:00.04 -zsh
```

terminal associé

état du processus

R	actif
T	bloqué
P	en attente de page
D	en attente de disque
S	endormi
IW	swappé
Z	tué

commande exécutée

Options de « ***ps*** » :

-a	liste tous les processus actifs
-u	format d'affichage long
-x	inclut les processus sans terminal

Tuer un processus :

```
# kill -9 <PID>
```

Processus en arrière-plan « ***&*** » :
(le terminal n'est pas bloqué)

```
# gedit monfichier.c &
```

Reprendre le comtrôle avec « ***fg*** »

Sommaire

Filtre simples :

cat	– affiche le contenu des fichiers passés en paramètres (par défaut, **stdin**) – options **-b**, **-n**, **-v**
more	– affiche page par page les fichiers passés en paramètres (par défaut, **stdin**) **h** pour avoir le détail des commandes
tee	– recopie l'entrée standard sur la sortie standard **et** dans le fichier passé en paramètre – option **-a**

exemples :

```
cat fic1 fic2                ls | tee liste.fic
more enormous_file           cat -n toto | more
```

Filtre avancé « ***grep*** » :

recherche, dans le fichier passé en paramètre, les lignes vérifiant une expression régulière donnée

grep *expression_régulière* [fichier]

Exemples :

- **grep 'toto' essai**
  cherche dans essai toutes les lignes qui contiennent le mot toto

- **grep '^[A-Z]' essai**
  cherche dans essai toutes les lignes qui commencent par une majuscule