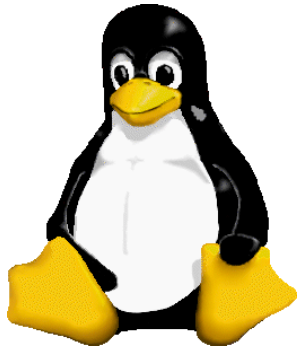
Système

Cours n°2

Isis TRUCK,
Université Paris 8

Remerciements

Ce cours est très fortement inspiré du cours de Marcel Bosc.



Les originaux sont disponibles aux formats OpenOffice et Powerpoint :

http://www-info.iutv.univ-paris13.fr/~bosc

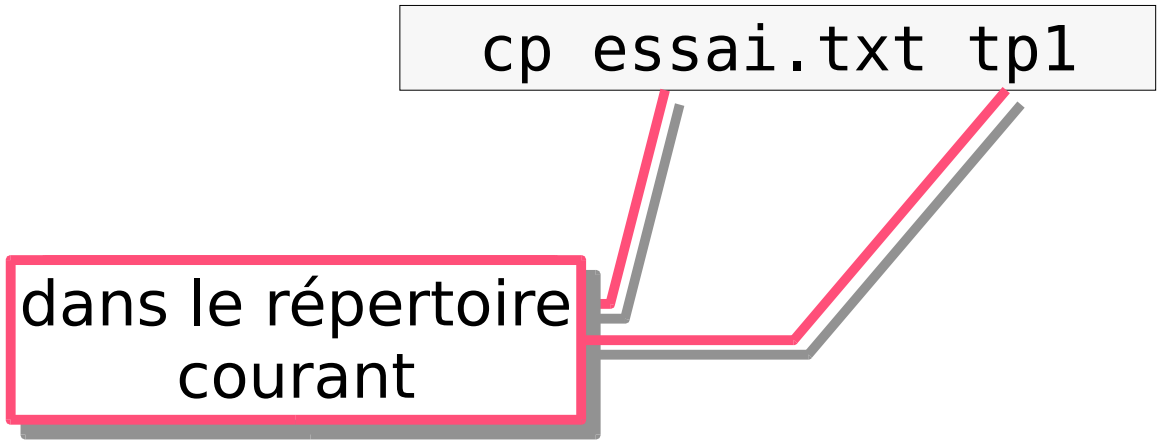
Plan

- chemins
- redirections
- programmation shell

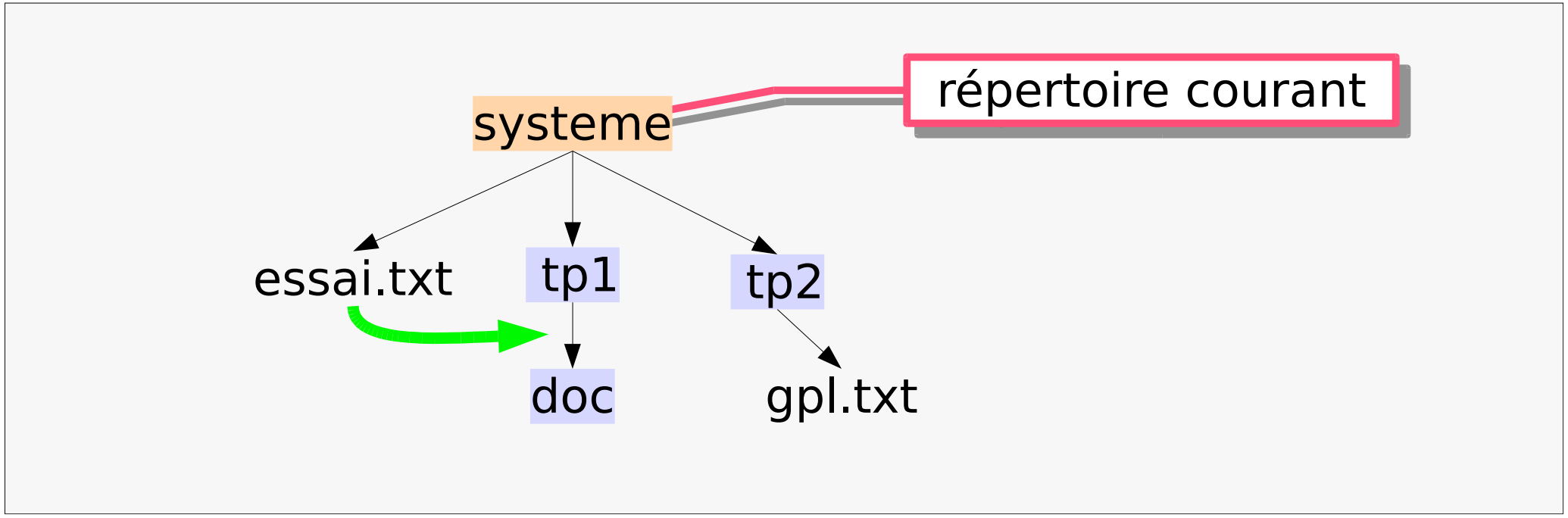
partie pratique

- commandes: find, tar, sed, wc

commandes: chemins simples



chemins relatifs
(très simples)

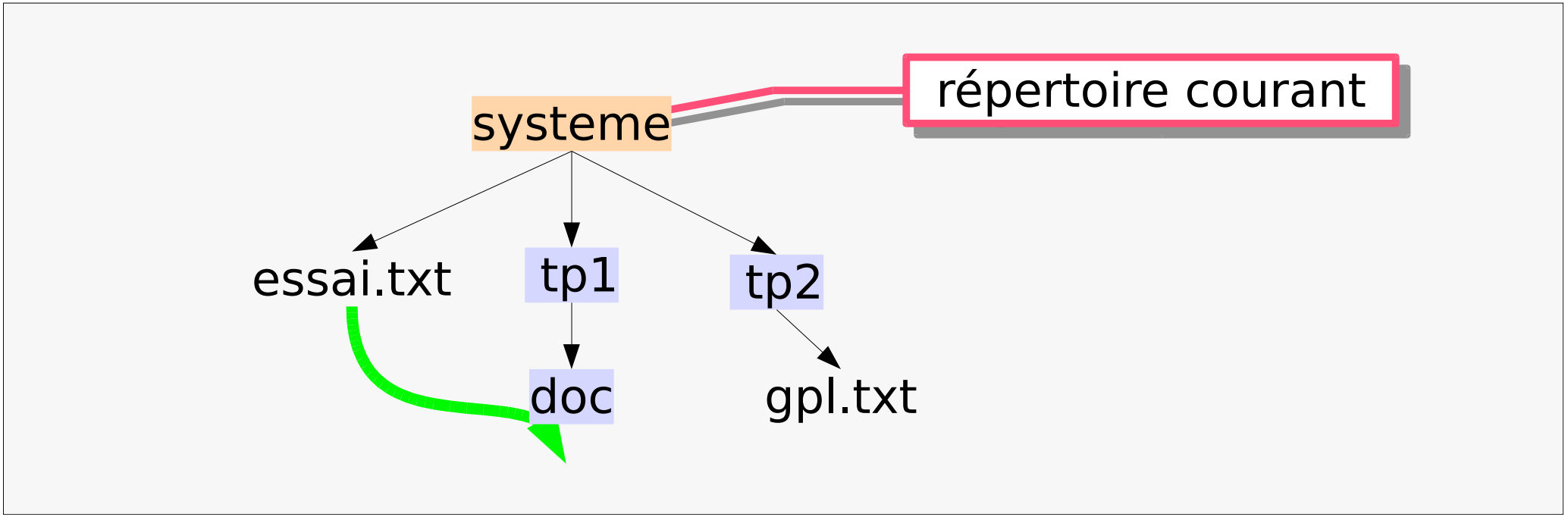


commandes: chemins relatifs

```
cp essai.txt tp1/doc
```

dans le répertoire courant

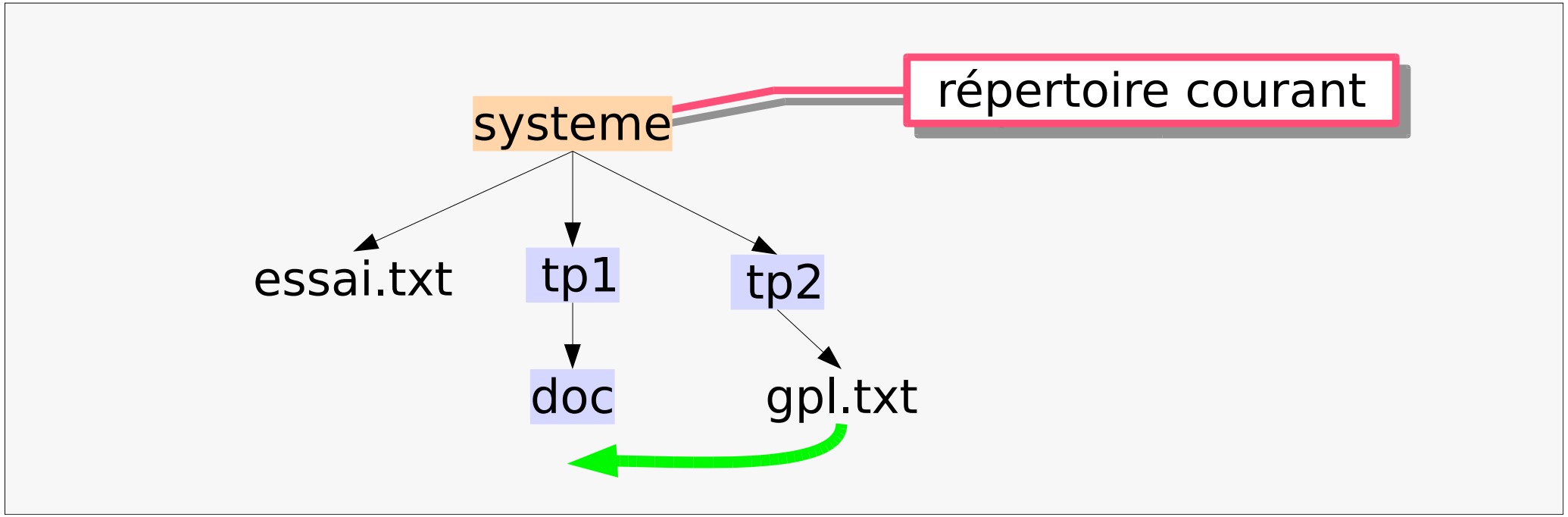
chemin relatif



commandes: chemins relatifs

```
cp tp2/gpl.txt tp1/doc
```

fichier source
pas forcement
dans répertoire courant



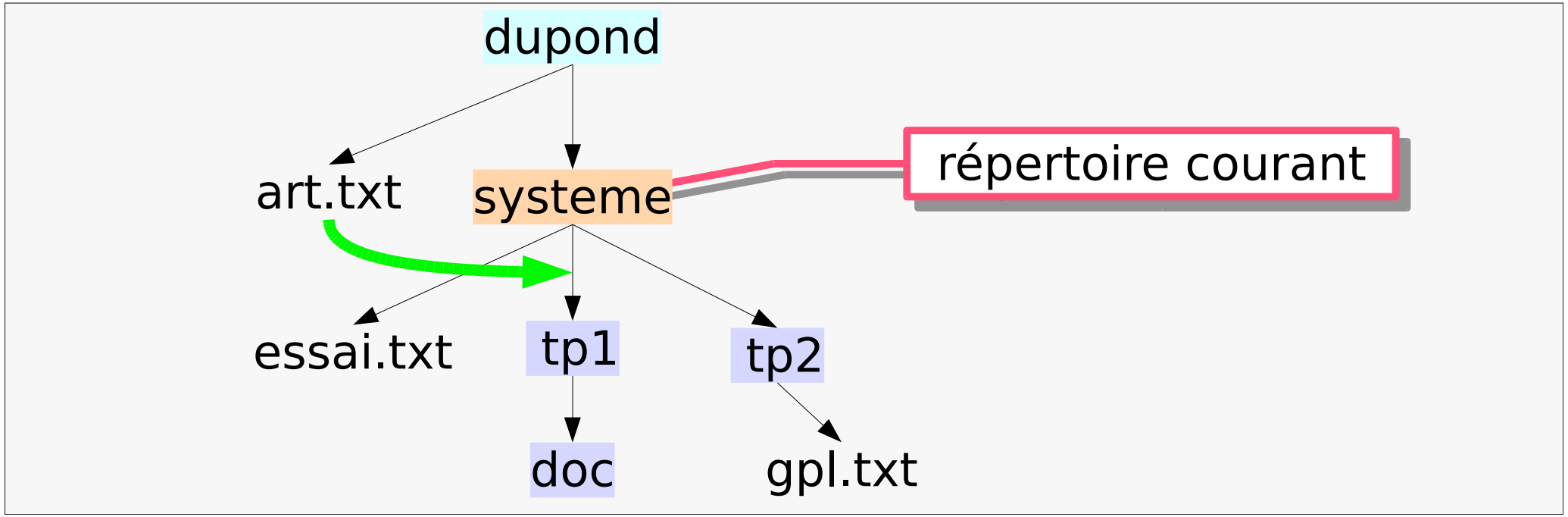
répertoires parent et courant

.. : parent
. : courant

```
cp ../art.txt .
```

répertoire parent

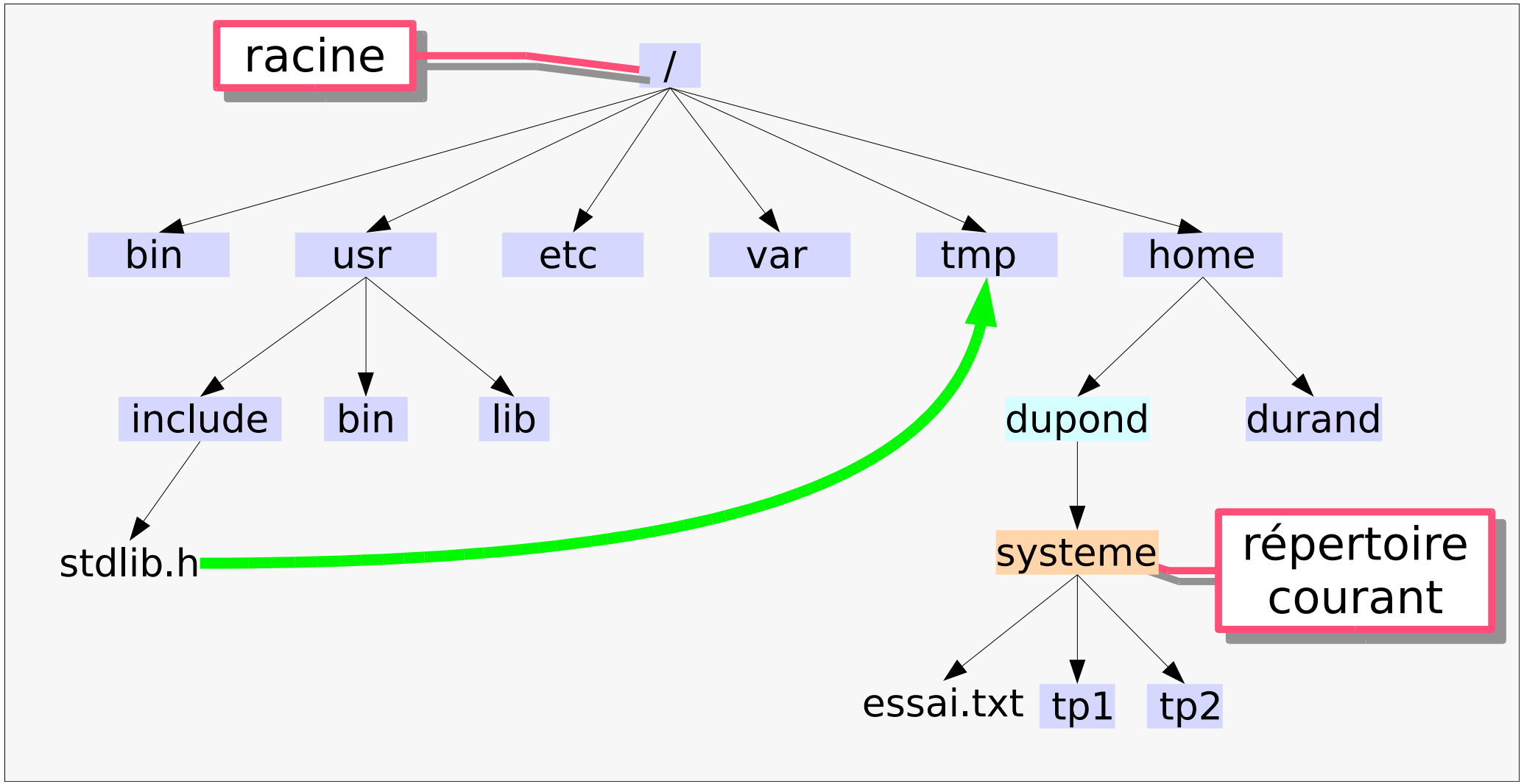
répertoire courant



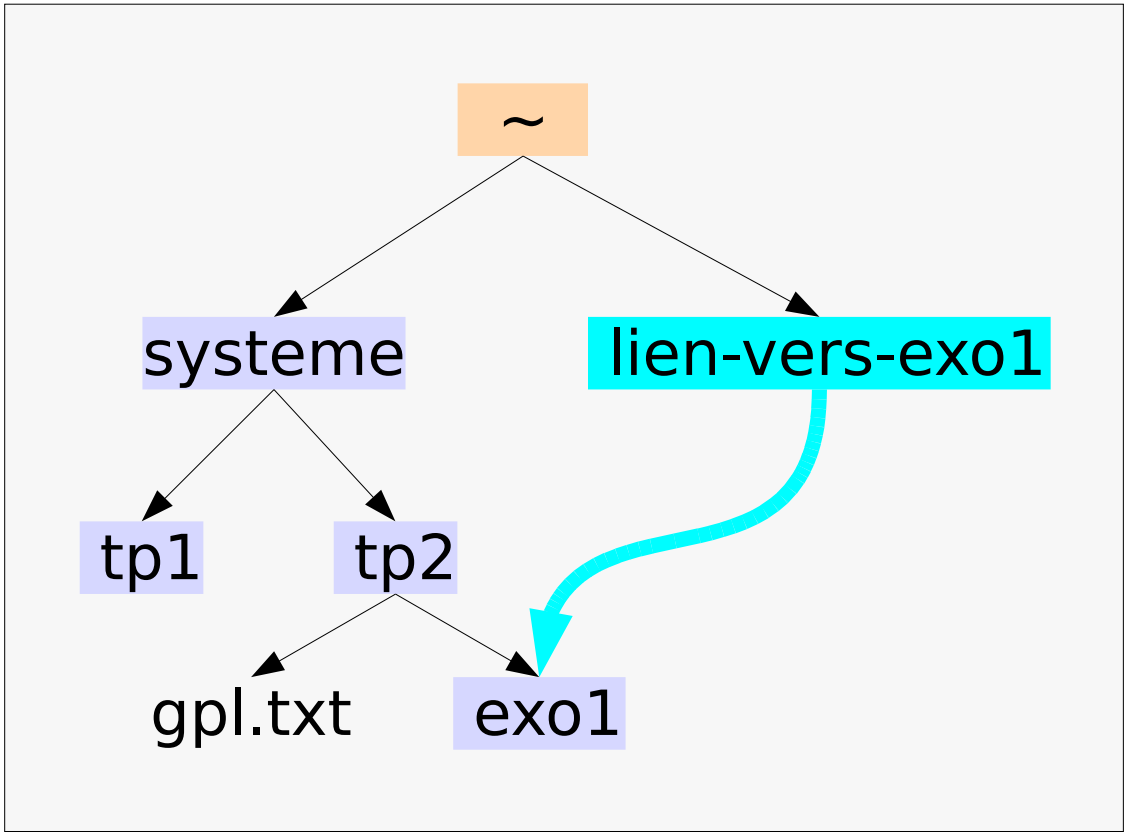
chemins absolus

chemins absolus

```
cp /usr/include/stdlib.h /tmp
```



liens symboliques



lien symbolique:
raccourci pour accéder à un fichier ou répertoire

NB : exo1 est un répertoire

```
ln -s systeme/tp2/exo1 lien-vers-exo1
```

Microsoft Windows « raccourcis »
« junctions »

exemple: on peut écrire
cd ~/lien-vers-exo1

redirections

- redirection de la sortie et de l'entrée
- les tubes
- exemples

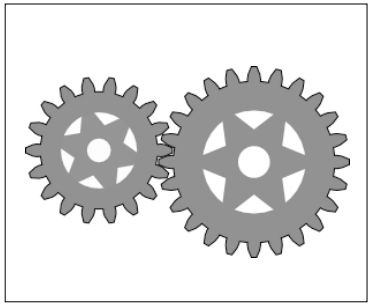
la sortie standard

```
[dupond@r10102 ~/public_html]# ls
README.html
HEADER.html
[dupond@r10102 ~/public_html]#
```

sortie standard de la commande ls

sortie standard = ce qui s'affiche normalement lorsqu'on exécute une commande

Commande : ls



sortie standard

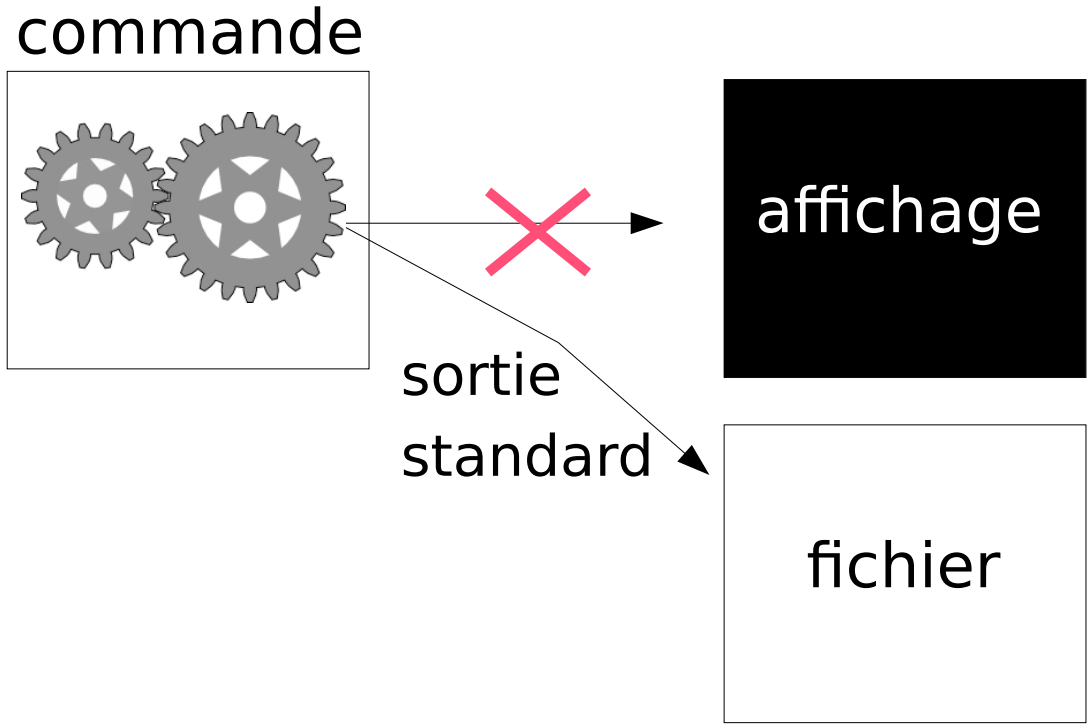
affichage

README.html
HEADER.html

rediriger la sortie standard

l'opérateur >

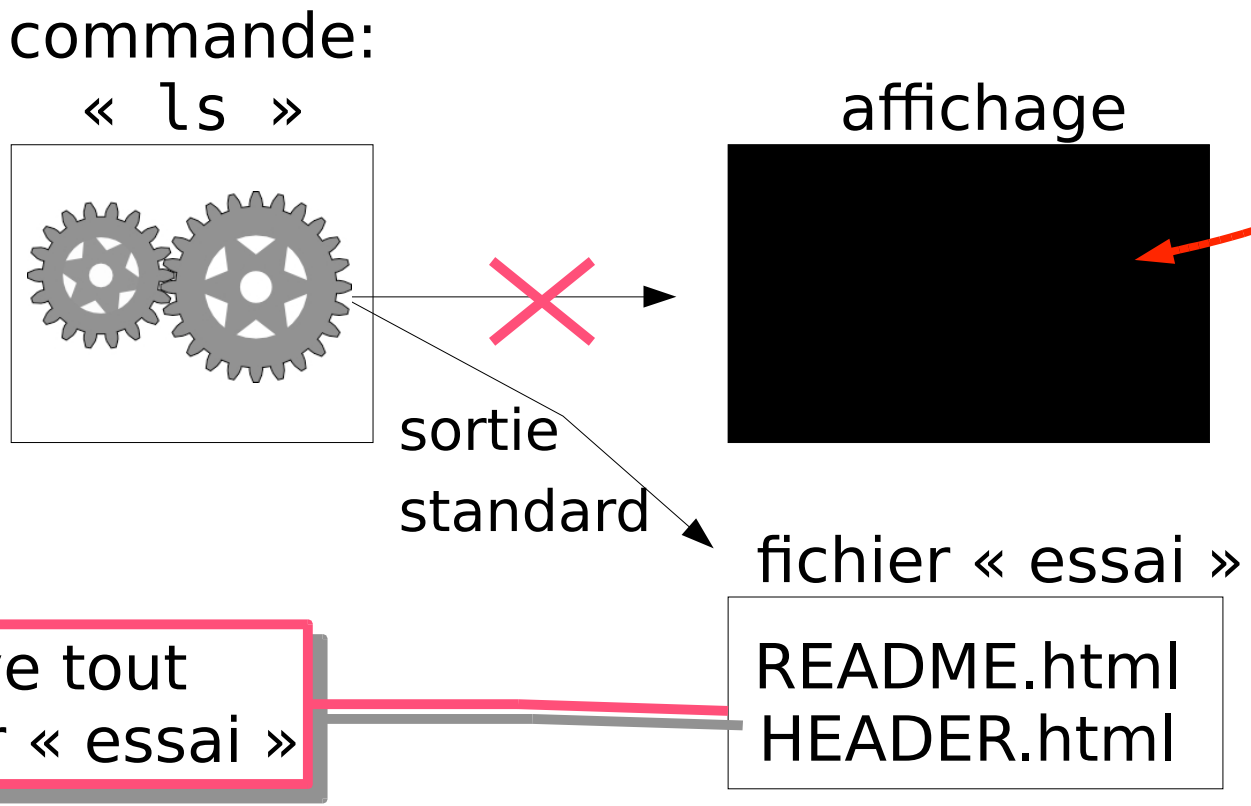
l'opérateur > redirige la sortie standard dans un fichier. Si le fichier existe déjà, il est écrasé



1er exemple de redirection

```
[dupond@r10102 ~/public_html]# ls > essai
[dupond@r10102 ~/public_html]#
```

rien n'est affiché!



on retrouve tout
dans le fichier « essai »

2e exemple de redirection

```
[dupond@r10102 ~/public_html]# pwd
/home/dupond/public_html
[dupond@r10102 ~/public_html]# pwd > essai2
[dupond@r10102 ~/public_html]#
```

affichage normal de 'pwd'

rien n'est affiché!

```
[dupond@r10102 ~/public_html]# ls
essai2
README.html
HEADER.html
[dupond@r10102 ~/public_html]#
```

nouveau fichier

rediriger la sortie standard

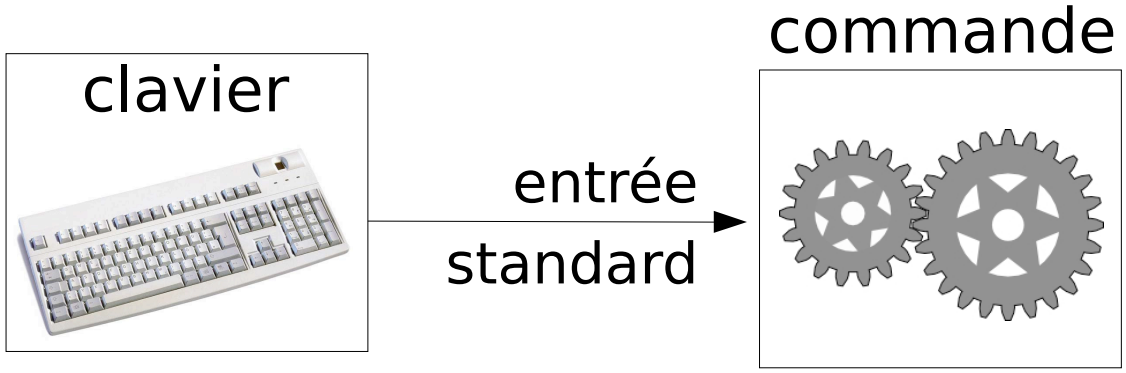
l'opérateur >>

```
[dupond@r10102 ~/public_html]# ls >> essai
```

l'opérateur >> redirige la sortie standard vers *la fin* d'un fichier existant. Le fichier est créé s'il n'existe pas

l'entrée standard

l'entrée standard = ce qui est saisi au clavier lorsqu'on exécute une commande

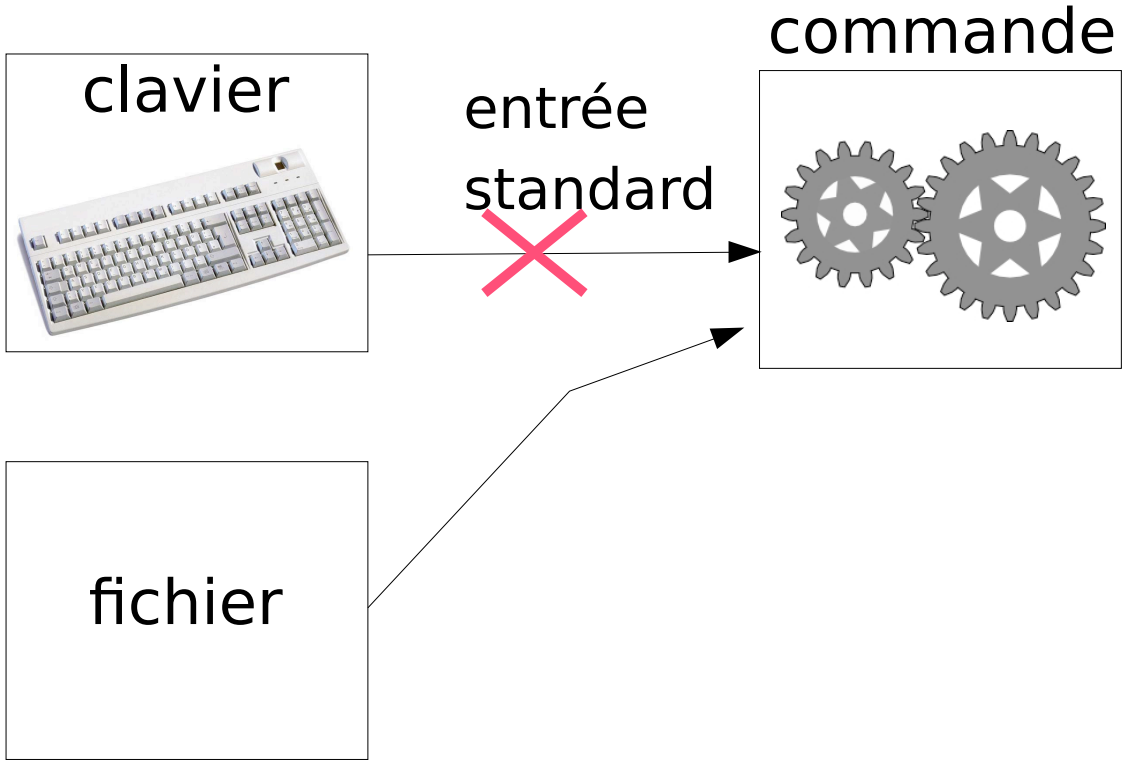


rediriger l'entrée standard

l'opérateur <

peu utilisé, sauf avec cat

```
[dupond@r10102 ~/public_html]# commande < essai
[dupond@r10102 ~/public_html]#
```

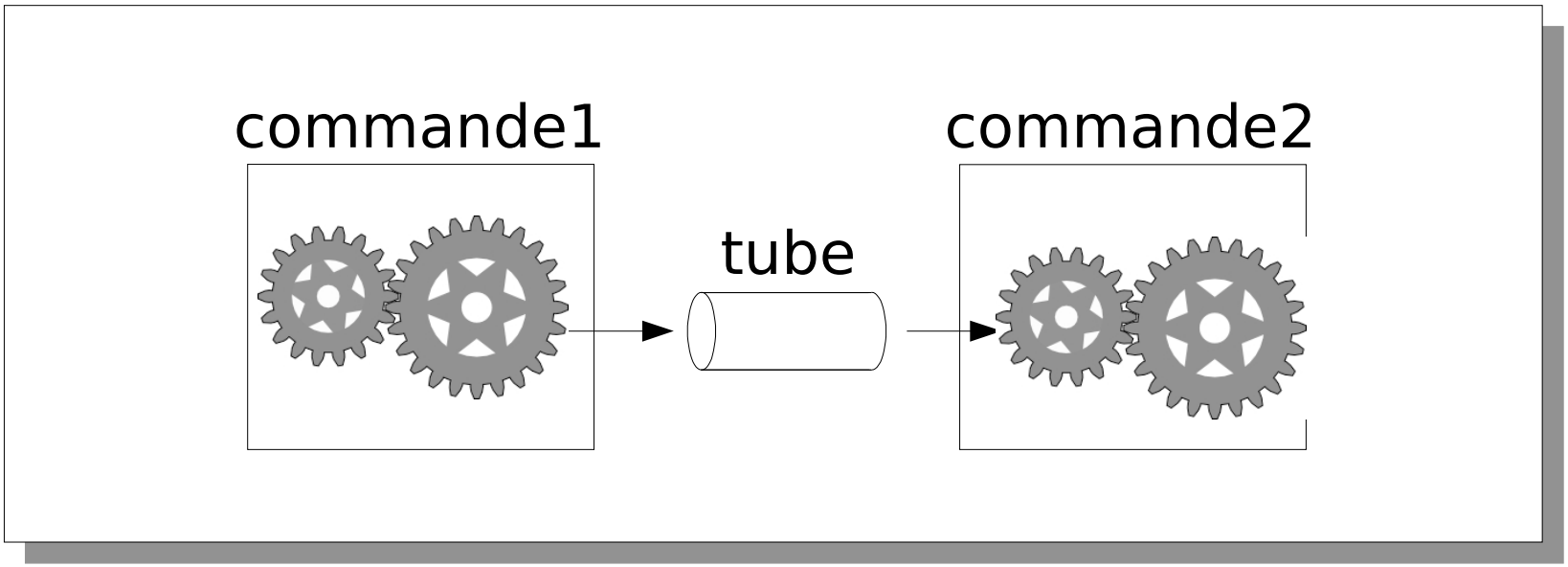


les tubes (*pipes*)

très utilisé!

redirige la sortie standard d'une commande dans l'entrée standard d'une autre commande

commande1 | commande2



la commande: **grep**

rechercher des informations dans un fichier
ligne par ligne

fichier.txt

```
[dupond@r10102 ~]# grep on fichier.txt
bonjour.
tonton.
[dupond@r10102 ~]#
```

pas de fichier en argument : entrée au clavier

commande très utile!

les tubes *(pipe)*

l'opérateur |

```
[dupond@r10102 ~/exemple]# ls
image.png
index.html
estimation.txt
toto.sxc
[dupond@r10102 ~/exemple]# ls | grep ima
image.png
estimation.txt
```

la sortie de **ls** redirigée dans l'entrée de **grep**

exemples

```
[truck@r10102 ~]# ps -ef
...
truck     21744     1  0 Aug16 ?        00:00:38 gnome-terminal
truck     21745 21744  0 Aug16 ?        00:00:00 gnome-pty-helper
root      31434  3096  0 Aug16 ?        00:00:03 spamd child
root      31793  3096  0 Aug16 ?        00:00:13 spamd child
truck      9272 21744  0 Aug16 pts/1    00:00:00 bash
root      17675     1  0 06:26 ?        00:00:17 /usr/sbin/cupsd -F
truck     23935  9272  0 10:44 pts/1    00:03:49 /usr/lib/openoffice
truck     24684     1  1 11:10 ?        00:07:36 /usr/lib/mozilla-fir
truck     25653     1  0 11:40 ?        00:00:02 emacs
truck     25660 25653  0 11:40 ?        00:00:00 /usr/bin/aspell -a
truck     28465 21744  0 13:28 pts/2    00:00:00 bash
truck     31731 23971  0 15:18 pts/4    00:00:00 emacs memo/plan.txt
truck     31746  9272  0 15:18 pts/1    00:00:01 gpdf introinfo.pdf
...
...
```

trop de choses, impossible à lire!

exemples

première idée: tout dans un fichier

```
[truck@r10102 ~]# ps -ef > essai.txt
```

... puis rechercher les informations dedans

```
[truck@r10102 ~]# less essai.txt
[truck@r10102 ~]# gedit essai.txt &
[2] 6586
[truck@r10102 ~]# grep bash essai.txt
```

exemples

meilleure solution : un tube

tous les processus contenant « bash »

```
[truck@r10102 ~]# ps -ef | grep bash
truck     9272 21744  0 Aug16 pts/1    00:00:00 bash
truck    28465 21744  0 13:28 pts/2    00:00:00 bash
```

tous les processus contenant « root »

```
[truck@r10102 ~]# ps -ef | grep root
root     31434  3096  0 Aug16 ?        00:00:03 spamd child
root     31793  3096  0 Aug16 ?        00:00:13 spamd child
root     17675     1  0 06:26 ?        00:00:17 /usr/sbin/cupsd -F
```

pas besoin d'un fichier intermédiaire

programmation shell: introduction

- variables simples
- variables d'environnement
- boucles "for"

exécuter plusieurs commandes

syntaxe:

```
commande1 ; commande2
```

exemples:

```
[dupond@r10102 ~/public_html]# echo bonjour ; ls
bonjour
HEADER.html README.html
[dupond@r10102 ~/public_html]#
```

résultat "echo"

résultat "ls"

```
[dupond@r10102 ~]# cd public_html ; ls ; echo bonjour
HEADER.html README.html
bonjour
[dupond@r10102 ~/public_html]#
```

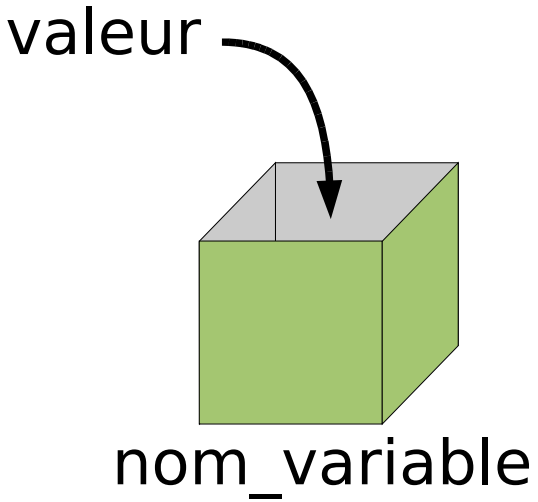
variables shell

affectation:

nom_variable=valeur

référence:

$nom_variable



exemples:

```
[dupond@r10102 ~]# repertoire=/etc
[dupond@r10102 ~]# cd $repertoire
[dupond@r10102 /etc]#
```

```
[dupond@r10102 ~]# prenom=Sabine
[dupond@r10102 ~]# echo bonjour $prenom
bonjour Sabine
[dupond@r10102 ~]#
```

script shell

script shell:
programme constitué
de commandes shell
dans un fichier texte

fichier: *essai.sh*

```
echo bonjour
pwd
cp fichier1 copie1
echo au revoir
```

```
[dupond@r10102 ~]# ./essai.sh
bonjour
/home/dupond
au revoir
[dupond@r10102 ~]#
```

programme simple!

script shell : exécution

on doit préciser le chemin du script, même dans le répertoire courant

```
[dupond@r10102 ~/bin]# ./essai.sh
```

./ exécuter le fichier essai.sh qui se trouve dans le **répertoire courant**

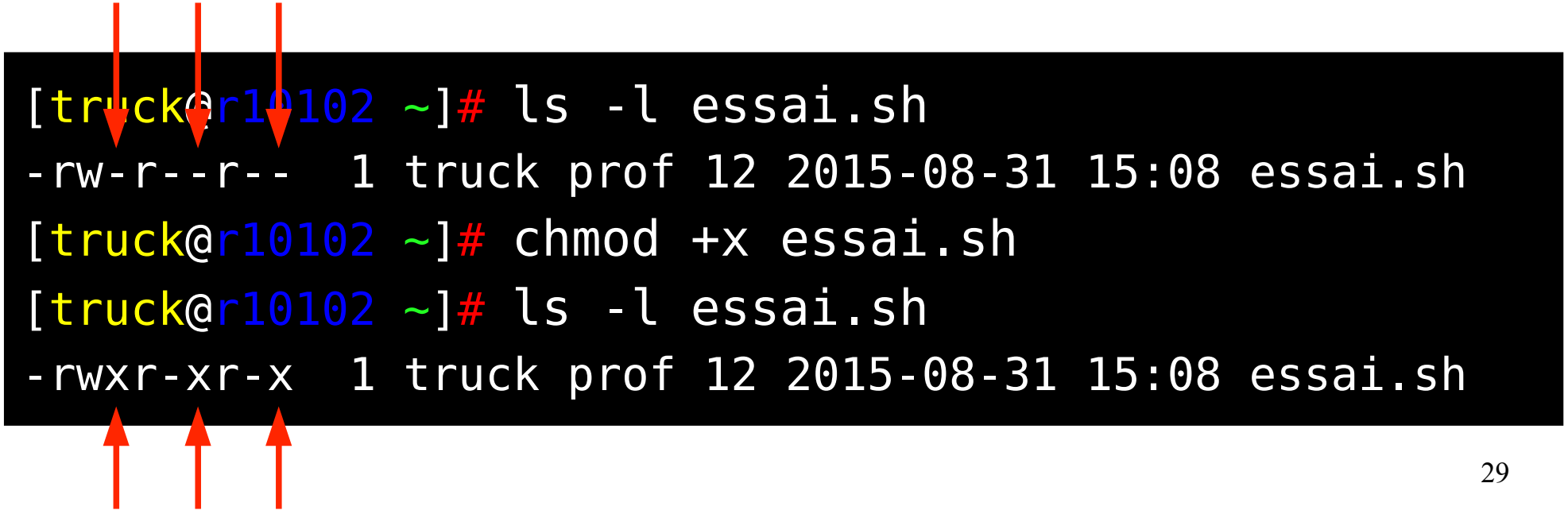
```
[dupond@r10102 ~]# ~/bin/essai.sh
```

~/bin/ **chemin complet**

script shell : exécution

droits d'exécution
sur un fichier

```
chmod +x nom-fichier.sh
```



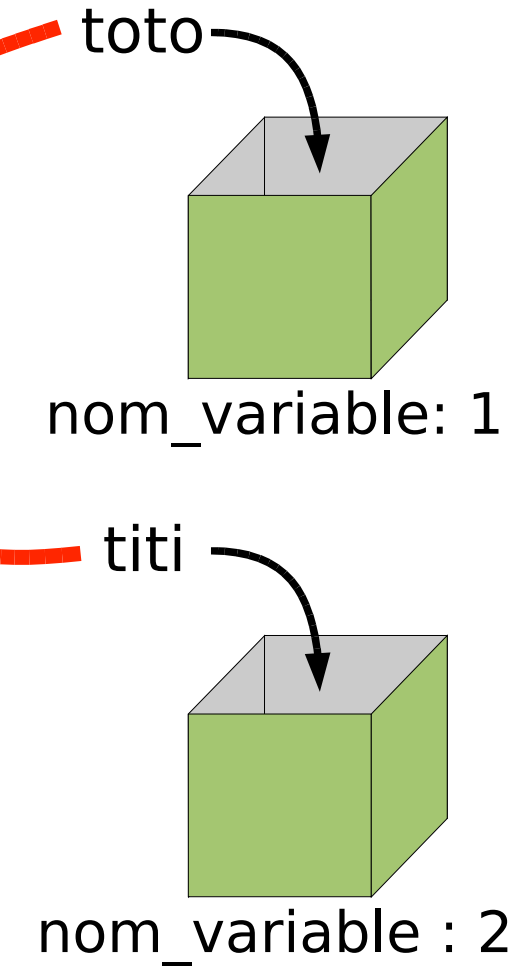
```
[truck@r10102 ~]# ls -l essai.sh
-rw-r--r--  1 truck prof 12 2015-08-31 15:08 essai.sh
[truck@r10102 ~]# chmod +x essai.sh
[truck@r10102 ~]# ls -l essai.sh
-rwxr-xr-x  1 truck prof 12 2015-08-31 15:08 essai.sh
```

arguments d'un script shell

fichier: *essai.sh*

```
echo bonjour
echo premier  argument: $1
echo deuxieme argument: $2
```

```
[dupond@r10102 ~]# ./essai.sh toto titi
bonjour
premier  argument: toto
deuxieme argument: titi
[dupond@r10102 ~]# ./essai.sh aaa bbb
bonjour
premier  argument: aaa
deuxieme argument: bbb
[dupond@r10102 ~]#
```



boucles "for"

boucle: exécuter plusieurs fois des commandes

boucles "for"

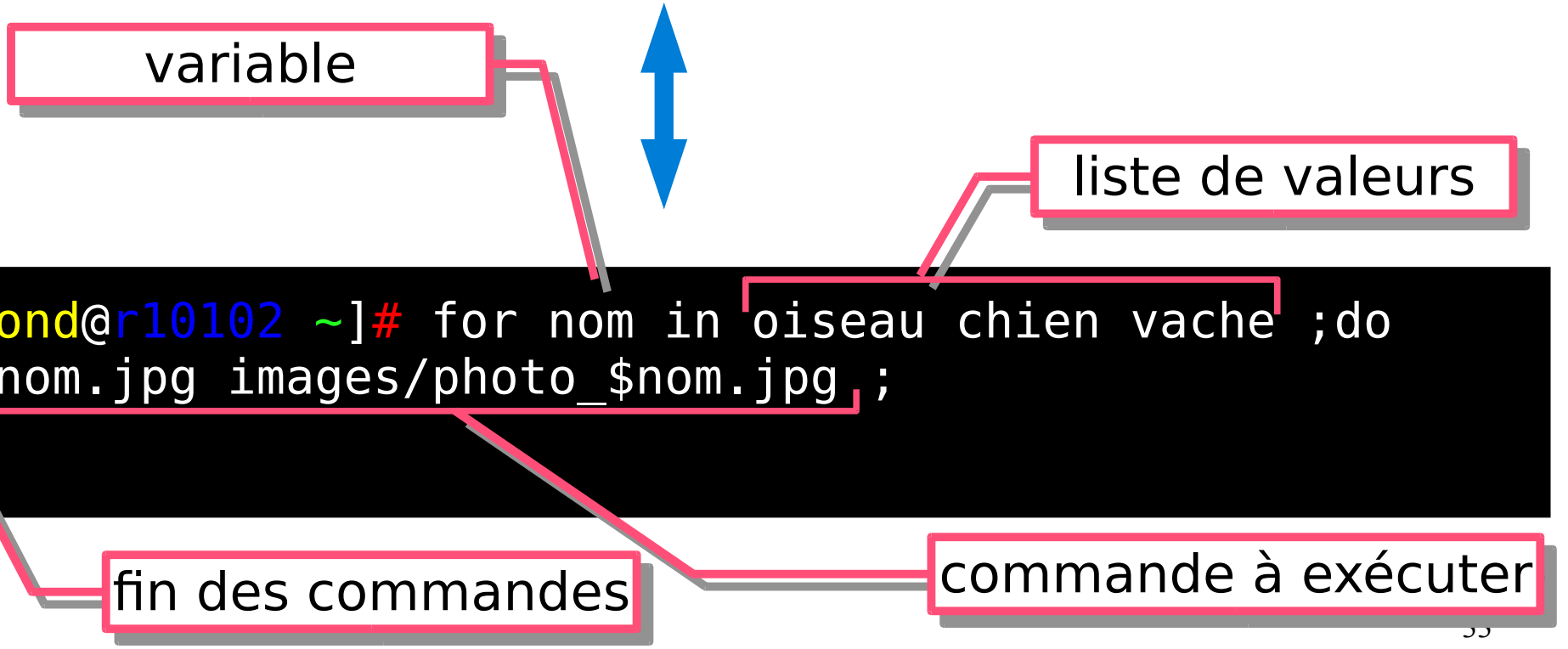
fichier: essai-boucle.sh

```
for prenom in Wassila Jean Alain ; do
     echo bonjour $prenom
done
```

```
[dupond@r10102 ~]# chmod +x essai-boucle.sh
[dupond@r10102 ~]# ./essai-boucle.sh
bonjour Wassila
bonjour Jean
bonjour Alain
[dupond@r10102 ~]#
```

boucles "for"

```
[dupond@r10102 ~]# cp oiseau.jpg images/photo_oiseau.jpg
[dupond@r10102 ~]# cp chien.jpg  images/photo_chien.jpg
[dupond@r10102 ~]# cp vache.jpg  images/photo_vache.jpg
...
```

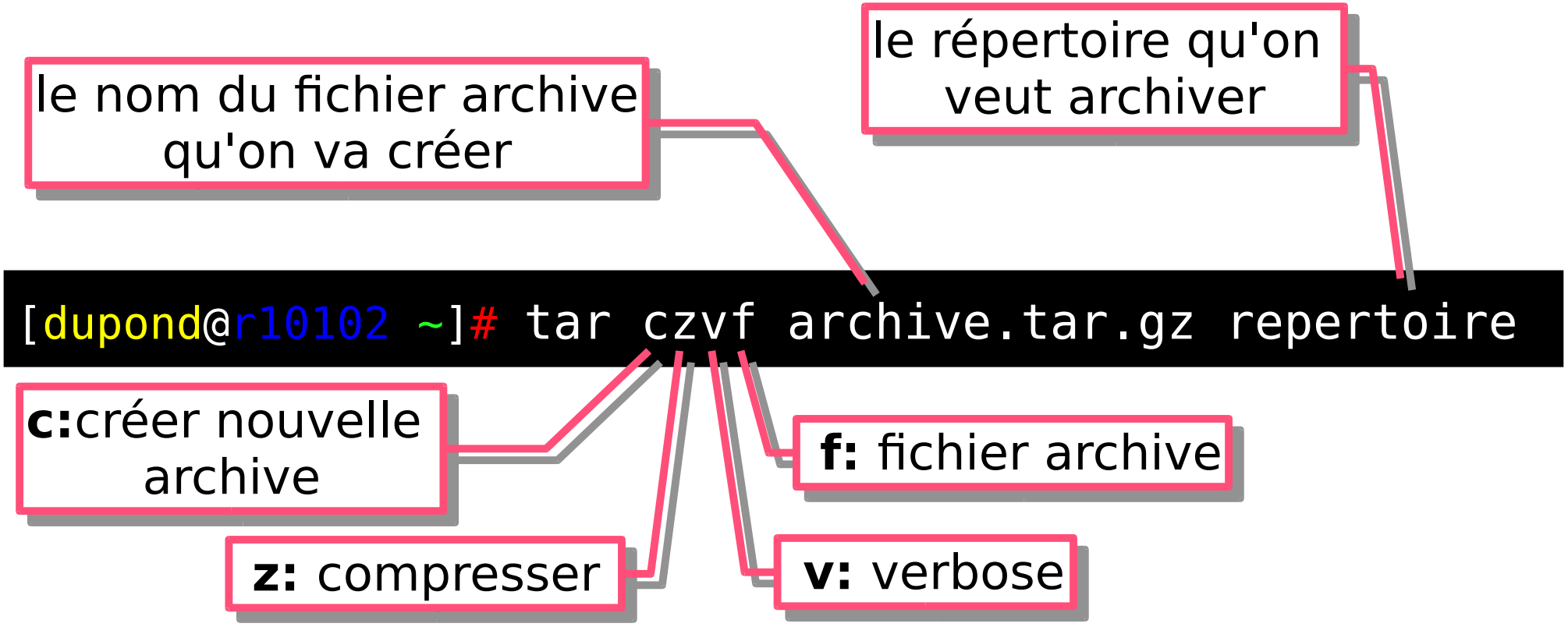


partie pratique

la commande **tar**

manipuler des archives

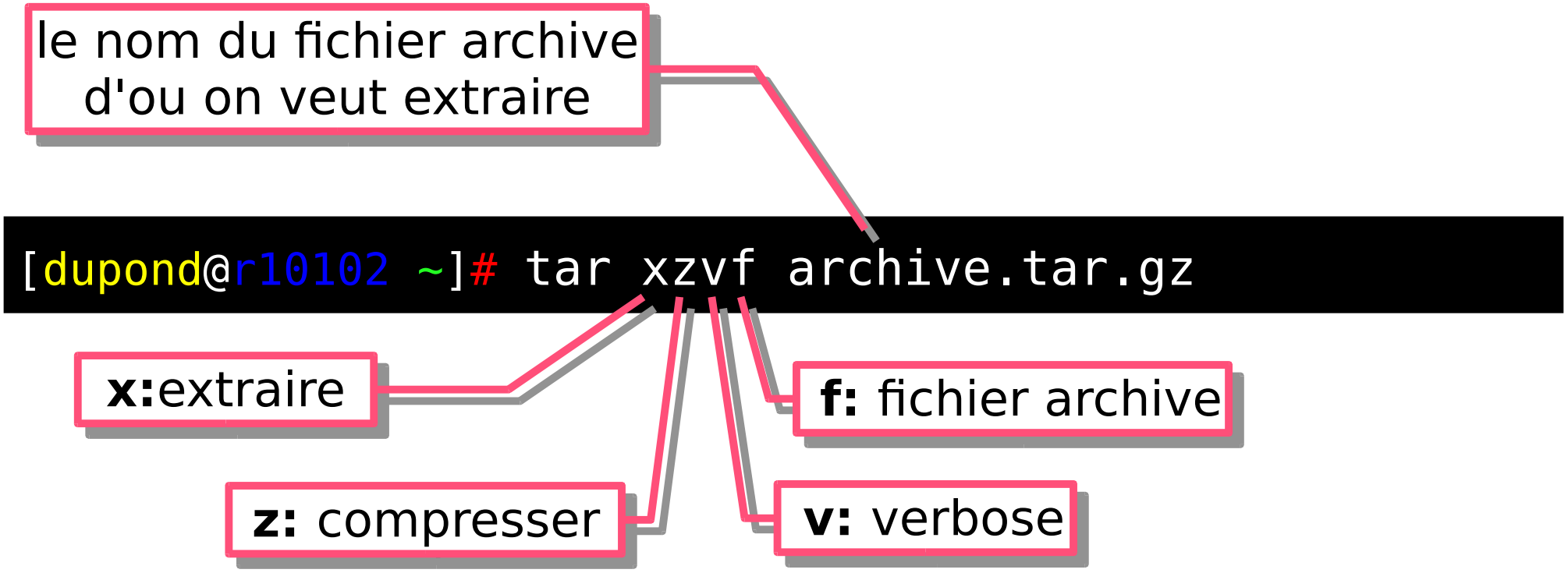
créer une archive



la commande **tar**

manipuler des archives

désarchiver



la commande **sed**

remplacer dans des fichiers

remplacer "on" par "af" dans fichier.txt

```
[dupond@r10102 ~]# sed -i 's/on/af/g' fichier.txt
```

fichier.txt

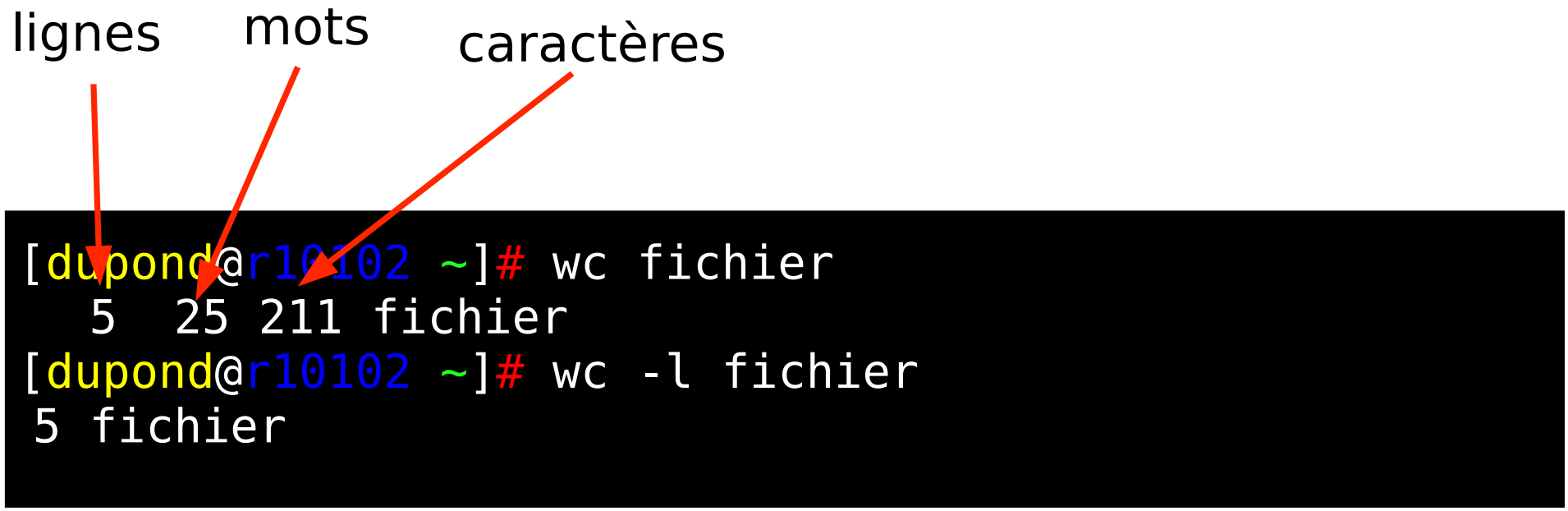
```
bonjour.
aurevoir.
tonton.
grouik.
paf.
```

fichier.txt

```
bafjour.
aurevoir.
taftaf.
grouik.
paf.
```

la commande **wc**

word count: compter le nombre de lignes, mots ou caractères dans un fichier



la commande **find**

rechercher des fichiers dans une arborescence

rechercher les fichiers terminant par .jpg

```
find . -iname "*.jpg"
```

rechercher les fichiers .h contenant "color" dans /usr/include

```
find /usr/include -iname "*color*.h"
```

rechercher les fichiers .jpg et les copier dans un répertoire

```
find . -iname "*.jpg" -exec cp {} ~/images \;
```

commande très utile!

la commande **find**

rechercher des fichiers dans une arborescence

rechercher les fichiers terminant par .jpg

```
find . -iname "*.jpg"
```

rec…lude

rec…ire

…;

**L'option '-iname' permet, au contraire de '-name',
de ne pas distinguer les majuscules des minuscules
(insensibilité à la casse).
Ainsi, on cherche tous les fichiers
finissant par .jpg, .JPG, .jPg, .Jpg, .jPG, etc.**

commande très utile!