Introduction système

Cours-4/5

Marcel.Bosc@iutv.univ-paris13.fr

2005-2006

Département informatique IUT de Villetaneuse Université Paris-13

Plan du cours

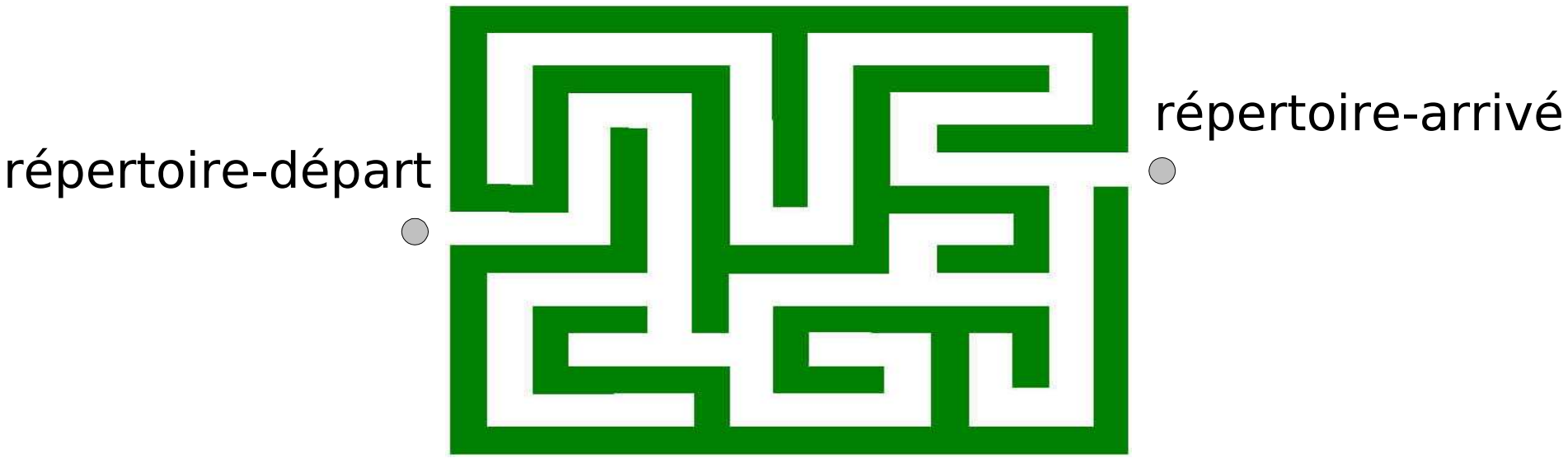
- chemins
- retour sur les tubes
- répertoires et droits d'accès
- montage, disques et partitions
- shell: configuration et scripts

partie pratique

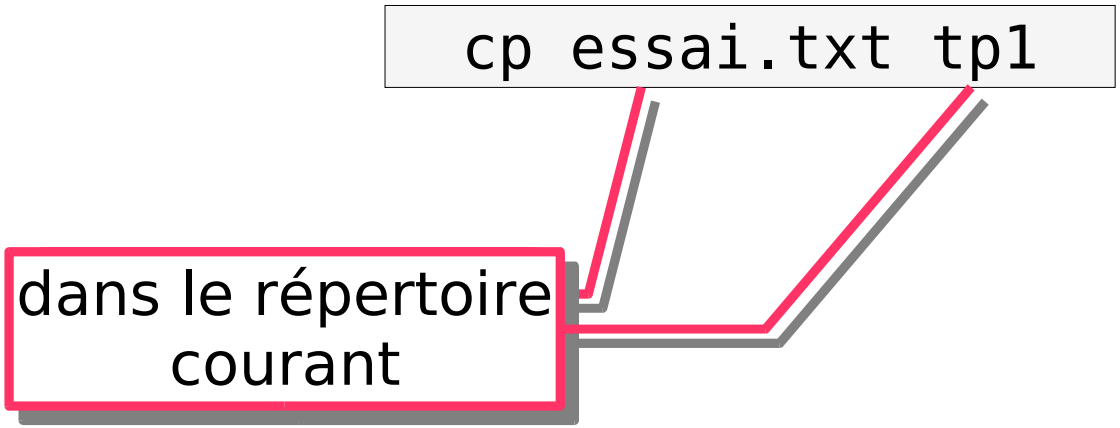
- caractères spéciaux
- quelques commandes

1ère partie

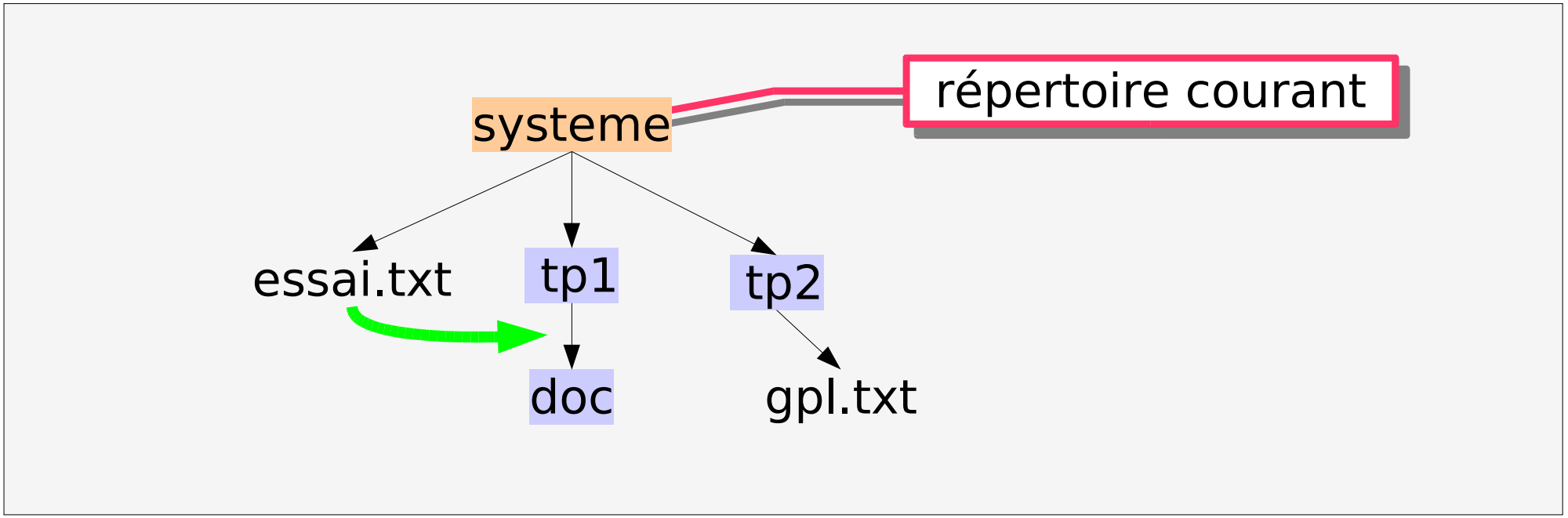
chemins



commandes: chemins simples



chemins relatifs
(très simples)

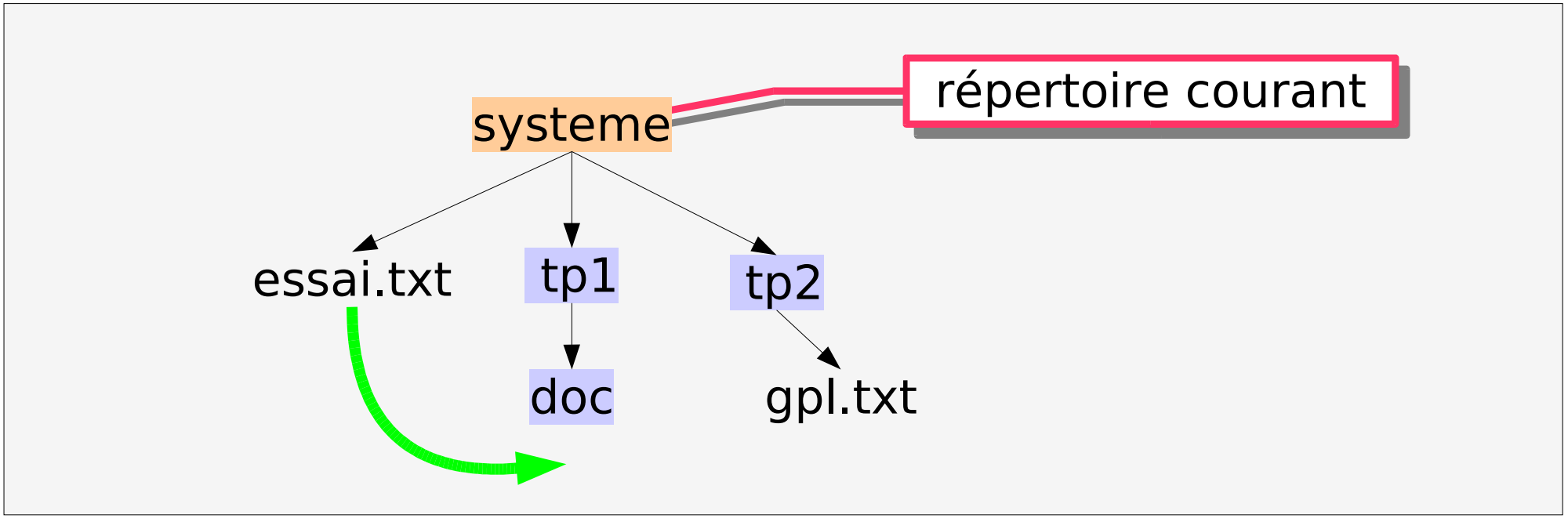


commandes: chemins relatifs

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

dans le répertoire courant

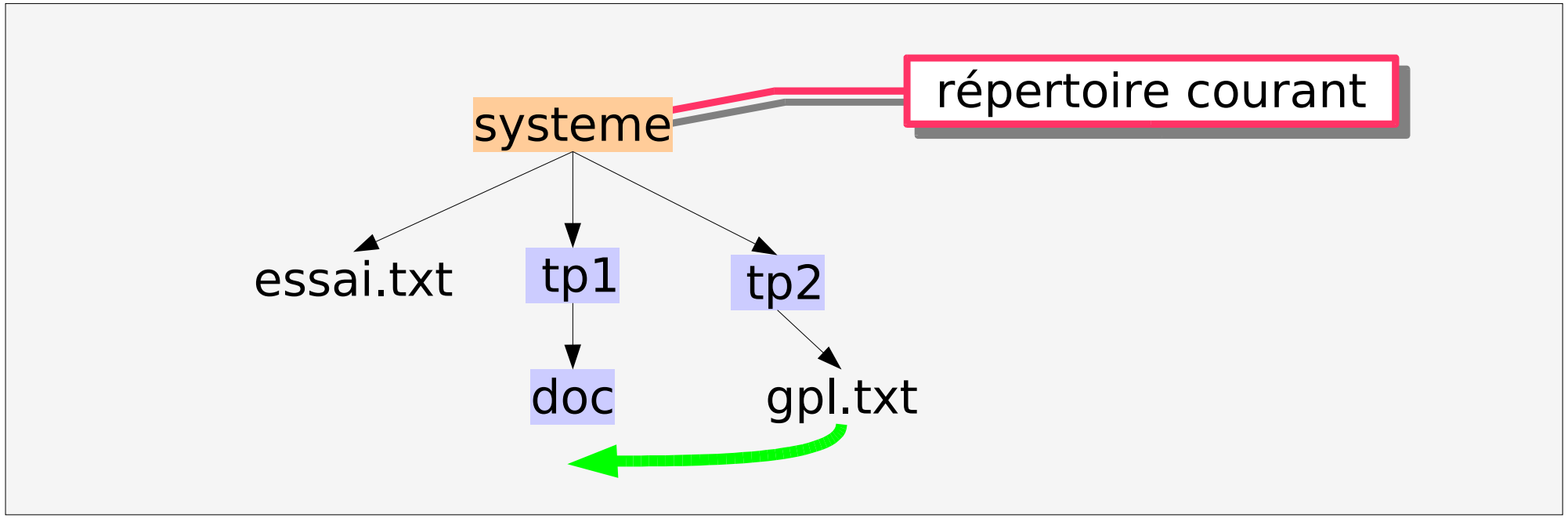
chemin relatif



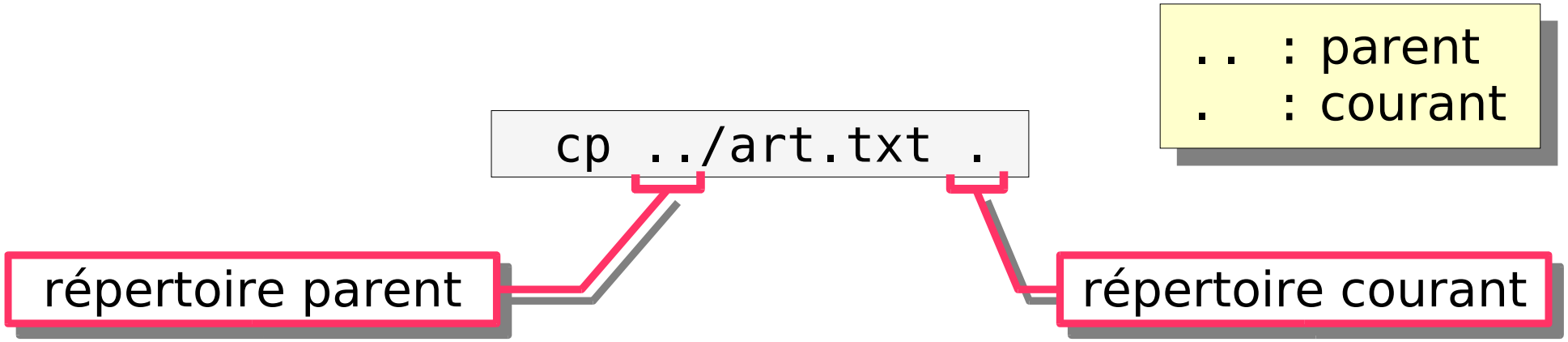
commandes: chemins relatifs

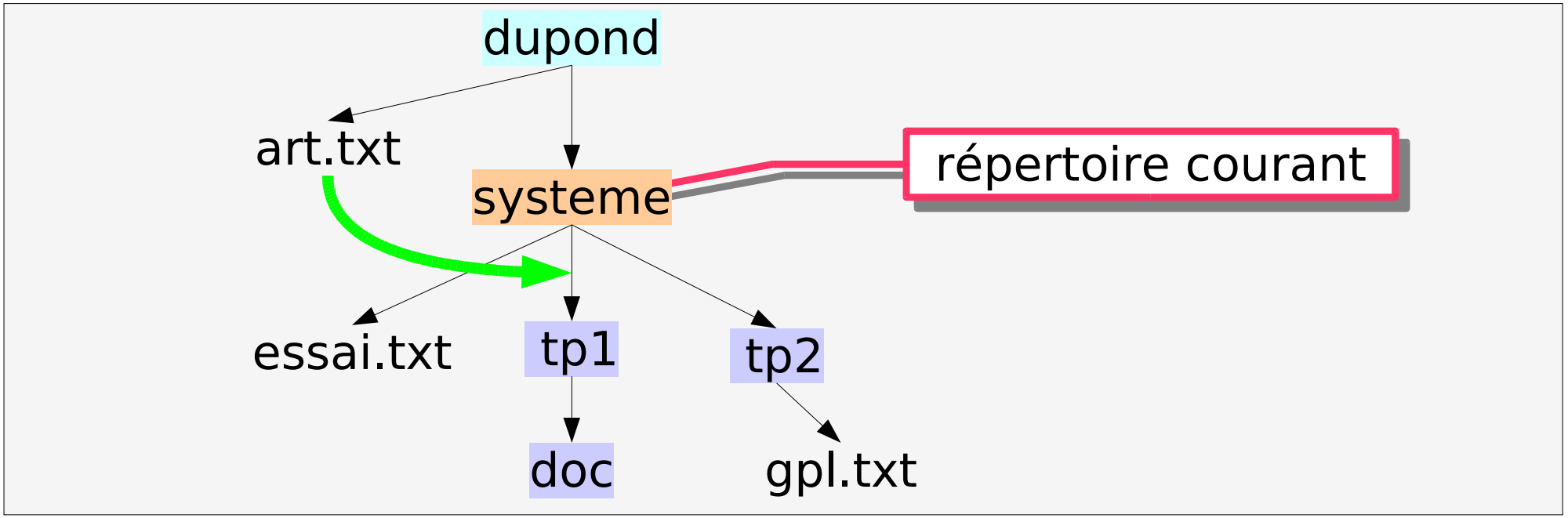
```
cp tp2/gpl.txt tp1/doc
```

fichier source
pas forcement
dans répertoire courant



répertoires parent et courant

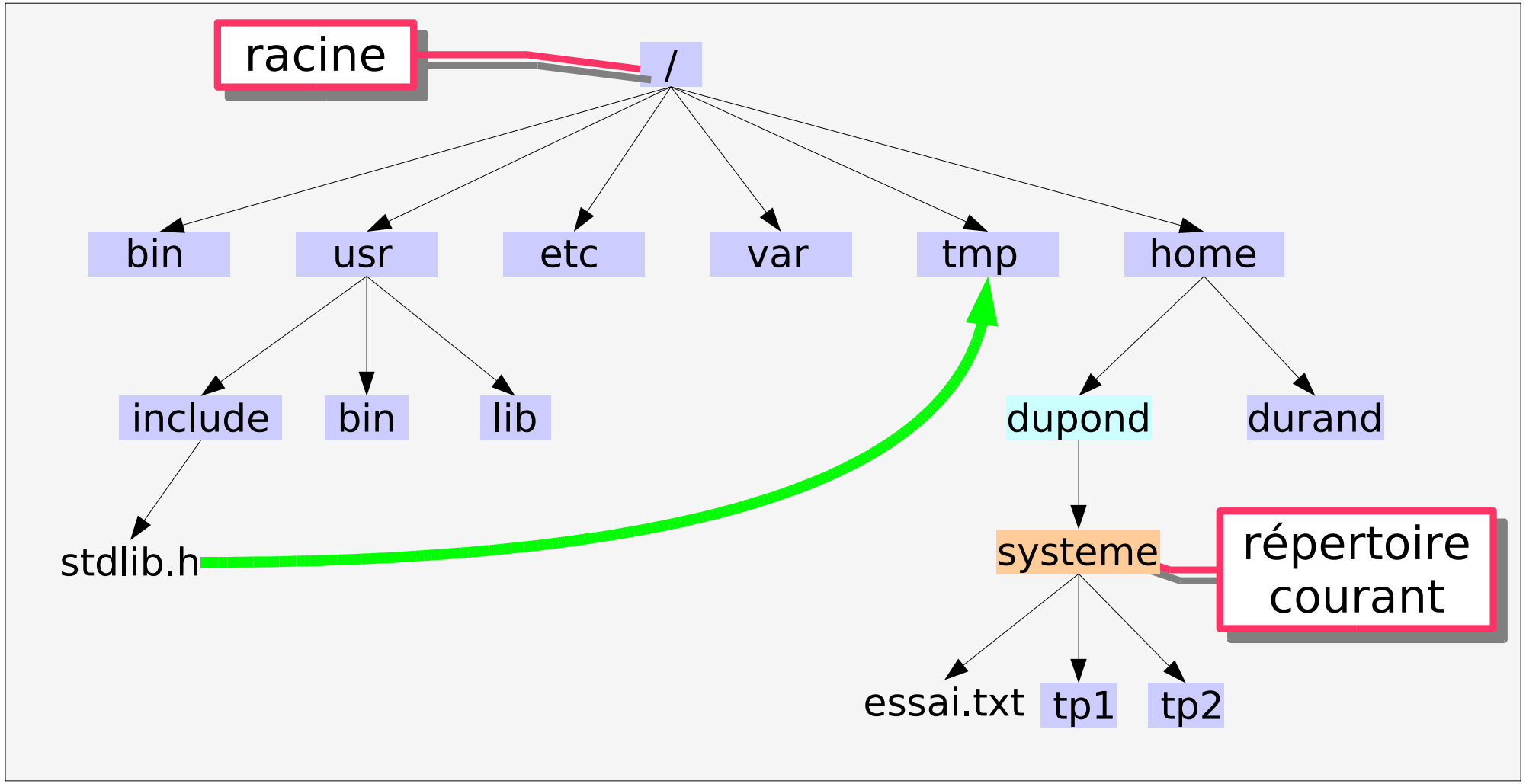




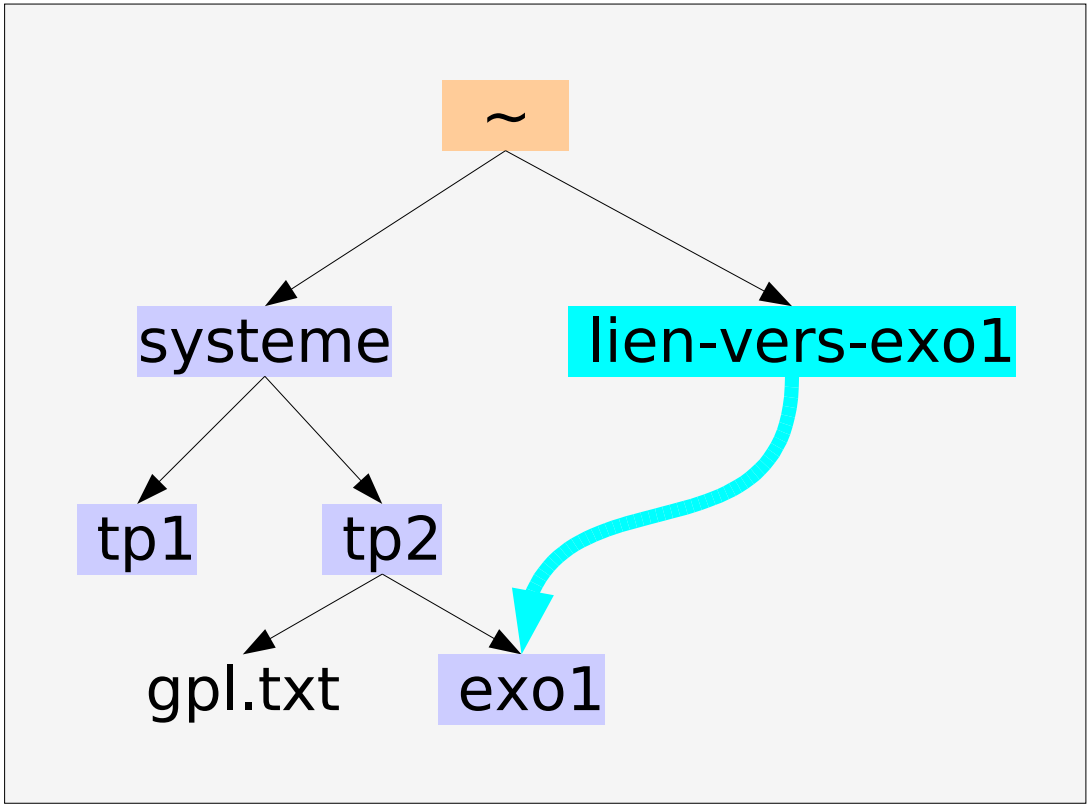
chemins absolus

chemins absolus

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



liens symboliques



lien symbolique:
raccourci pour accéder à un fichier ou répertoire

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

Microsoft Windows

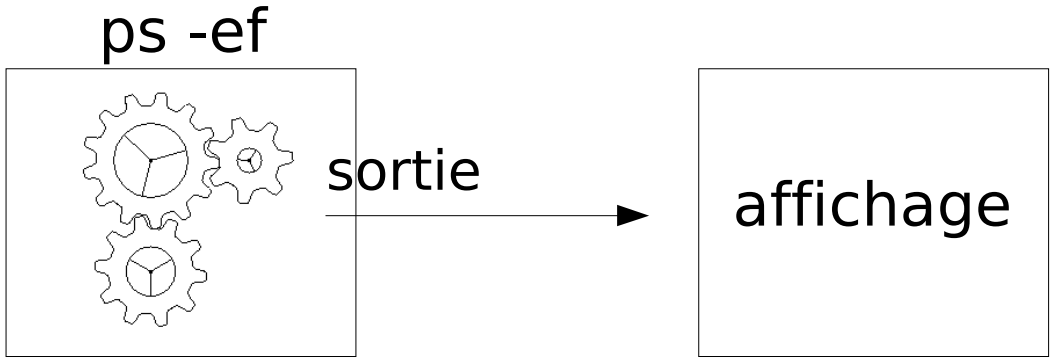
« raccourcis »
« junctions »

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

2ème partie

retour sur les tubes

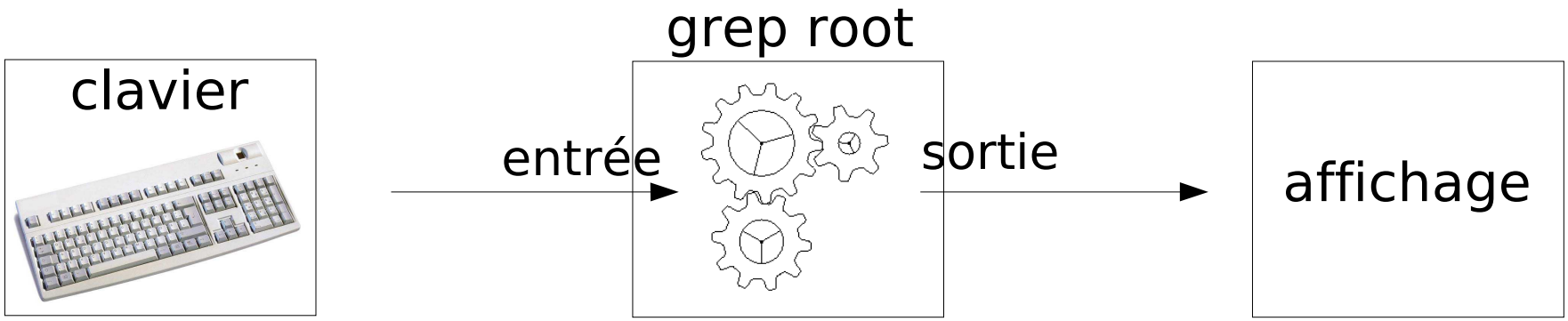
sortie de « ps -ef »

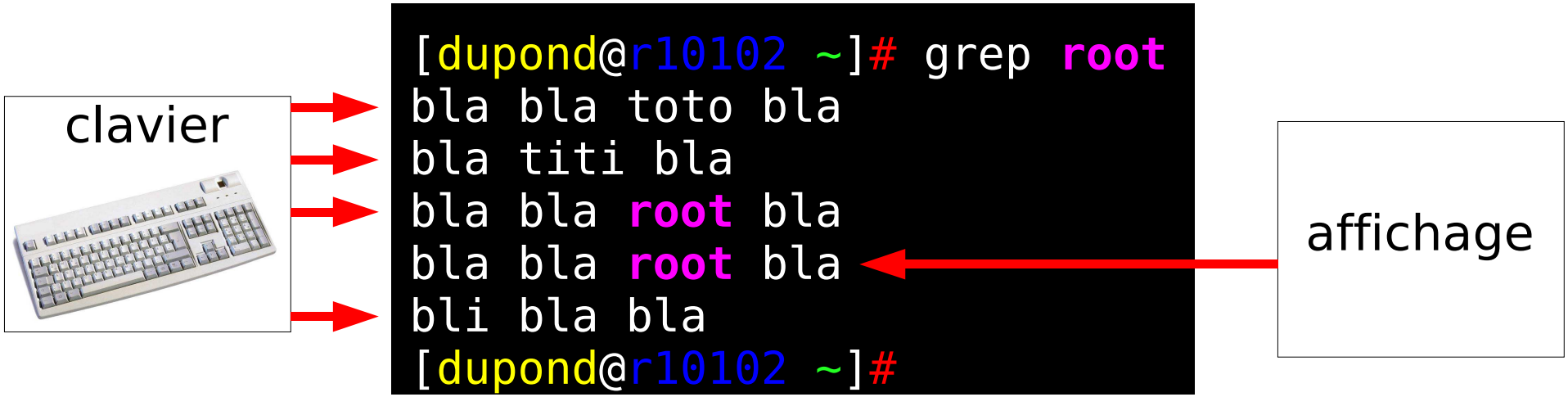


affichage

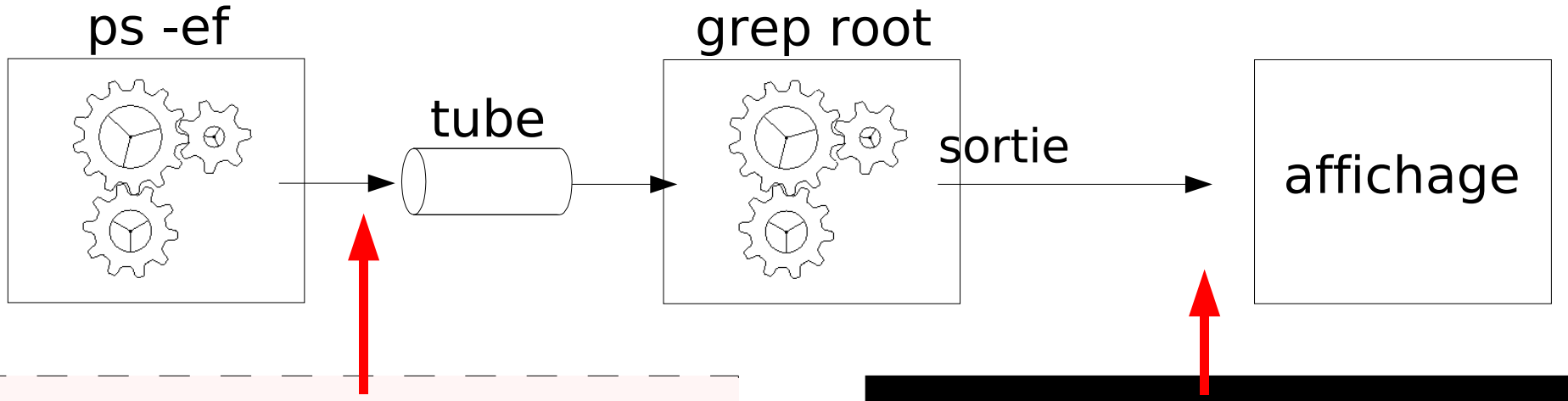
```
[dupond@r10102 ~]# ps -ef
bosc 21745  gnome-terminal
root 31793  spamd child
bosc  9272  bash
root 17675  cupsd -F
...
[dupond@r10102 ~]#
```

entrée et sortie de « grep root »





tube « ps -ef | grep root »




```
bosc 21745  gnome-terminal
root 31793  spamd child
bosc  9272  bash
root 17675  cupsd -F
```

sortie de ps -ef
(pas affiché!)

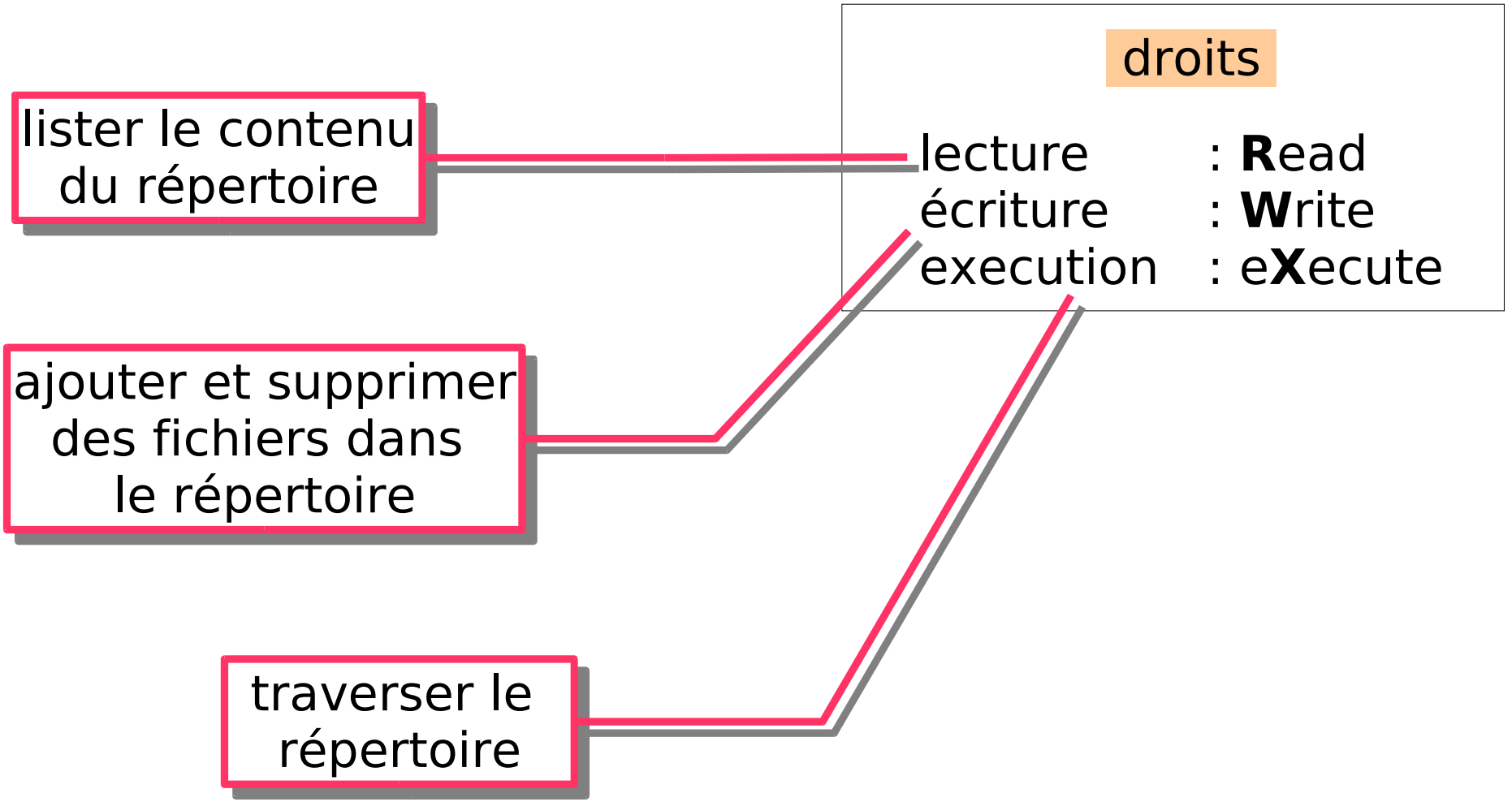
```
root 31793  spamd child

root 17675  cupsd -F
```

sortie de grep = affichage

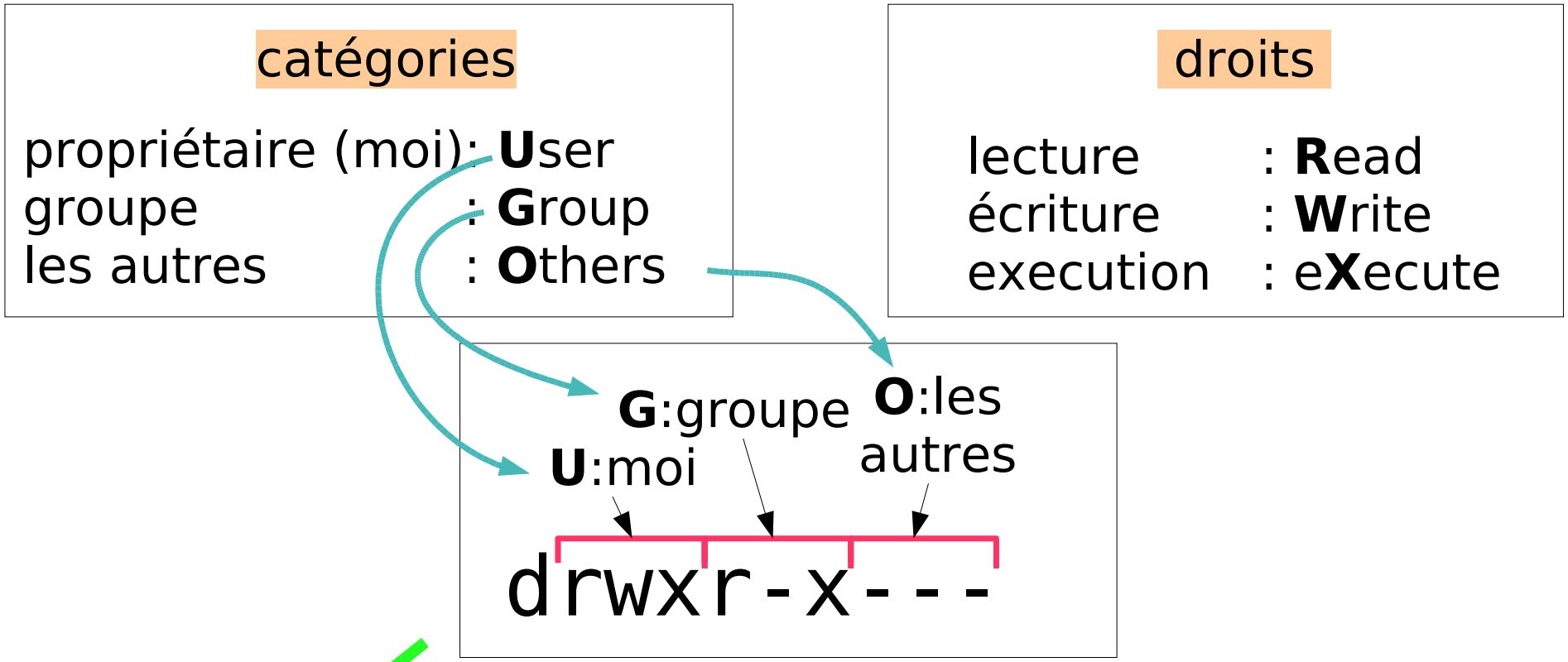
3ème partie

répertoires et droits d'accès

droits d'accès aux répertoires

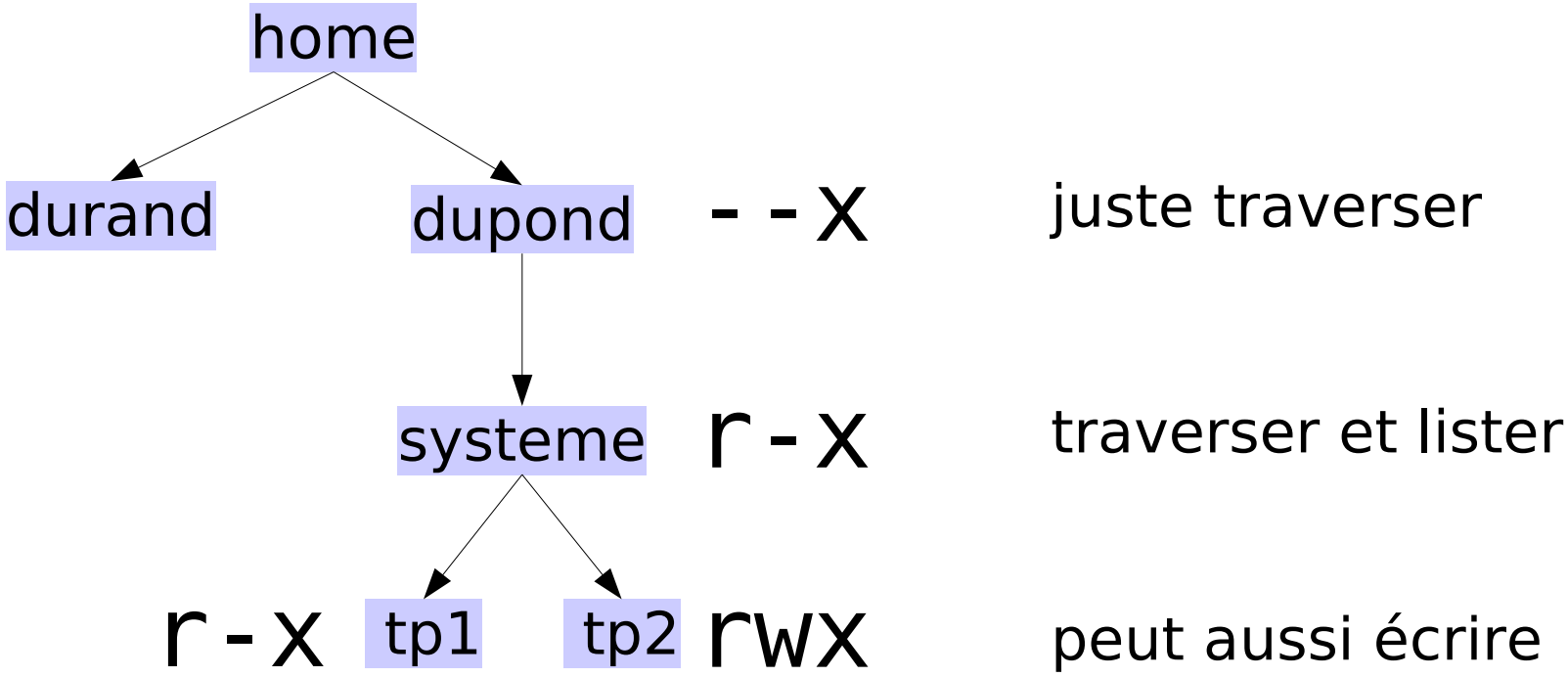


droits d'accès aux répertoires



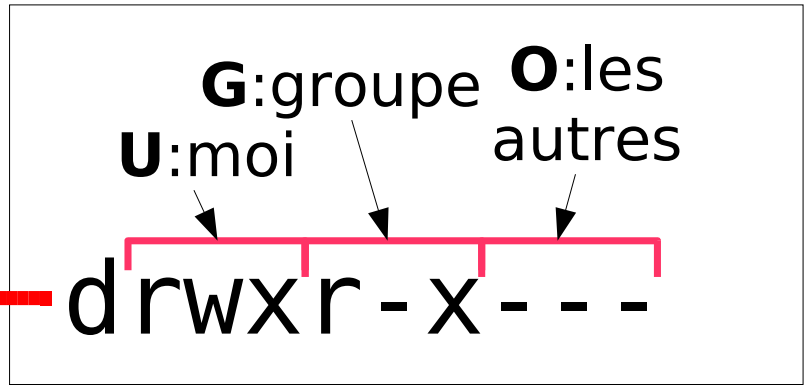

```
[dupond@r10102 ~/exemple]# ls -l
-rw-r-----  1 bosc prof 17:44 fichier.txt
drwxr-x---  1 bosc prof 15:31 repertoire
```

droits d'accès aux répertoires



fichier, répertoire ou lien?

- : fichier normal
d : répertoire
l : lien symbolique




```
[dupond@r10102 ~/exemple]# ls -l
-rw-r-----  1 bosc prof 17:44 fichier.txt
drwxr-x---  1 bosc prof 15:31 repertoire
lrwxrwxrwx  1 bosc prof 15:31 lienverstoto -> tp1/toto
```

4ème partie

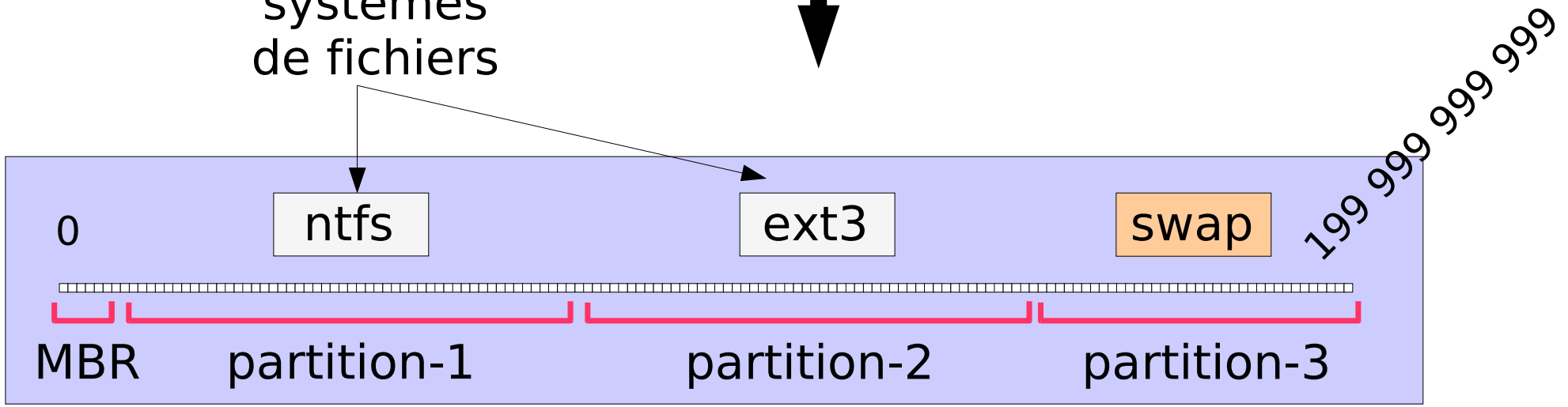
montage, disques et partitions

partitions d'un disque dur

rappel

disque dur
200 Giga Octets

systèmes
de fichiers



noms de périphériques Linux

IDE

4 périphériques

hda
hdb
nappe1

hdc
hdd
nappe2

partitions: 1,2,3 ...
hda1, hda2, hda3 ...
hdb1, hdb2, hdb3 ...

SCSI / SATA / USB

sda
sdb
sdc
sdd
sde
...

partitions: 1,2,3 ...
sda1, sda2, sda3 ...
sdb1, sdb2, sdb3 ...

dans le répertoire : /dev

emplacement physique des fichiers

disque réseau

sda1

hdc1

serveur:/share

partitions 1,2,3:
hda1,hda2,hda3

noms des périphériques

par quel chemin y accéder?

emplacement logique

rtition 1
(hda1)

/

usr

etc

var

tmp

home

include

bin

lib

dupond

durand

partition 2
(hda2)

disque
réseau

montage de disques

rtition 1
(hda1)

/

usr

etc

var

tmp

home

include

bin

lib

partition 2
(hda2)

```
[root@r101 ~]# mount /dev/hda2 /usr
[root@r101 ~]#
```

la commande « df »

```
[dupond@r10102 ~]# df -h
Sys. de fich.  Tail.  Occ.  Disp.  %Occ.  Monté sur
/dev/sda1       14G   6,7G   6,5G   51%   /
/dev/sda3       58G    46G   9,3G   84%   /home
/dev/sdb1      151G    97G    47G   68%   /disk2
[dupond@r10102 ~]#
```

sda

sda1: 14Go➡ /
sda3: 58Go➡ /home

sdb

sdb1:151Go➡/disk2

5ème partie

shell: configuration et scripts

- la variable PATH
- scripts shell
- personnalisation (.bashrc)

variables d'environnement

rappel

configuration de votre environnement

```
export nom_variable=valeur
```

```
printenv
```

exemples importants:

PATH:	où chercher vos programmes
HOME:	votre répertoire personnel (~)
LD_LIBRARY_PATH:	où chercher des librairies dynamiques

```
[dupond@r10102 ~]# echo $HOME
/home/dupond
[dupond@r10102 ~]#
```

la variable PATH

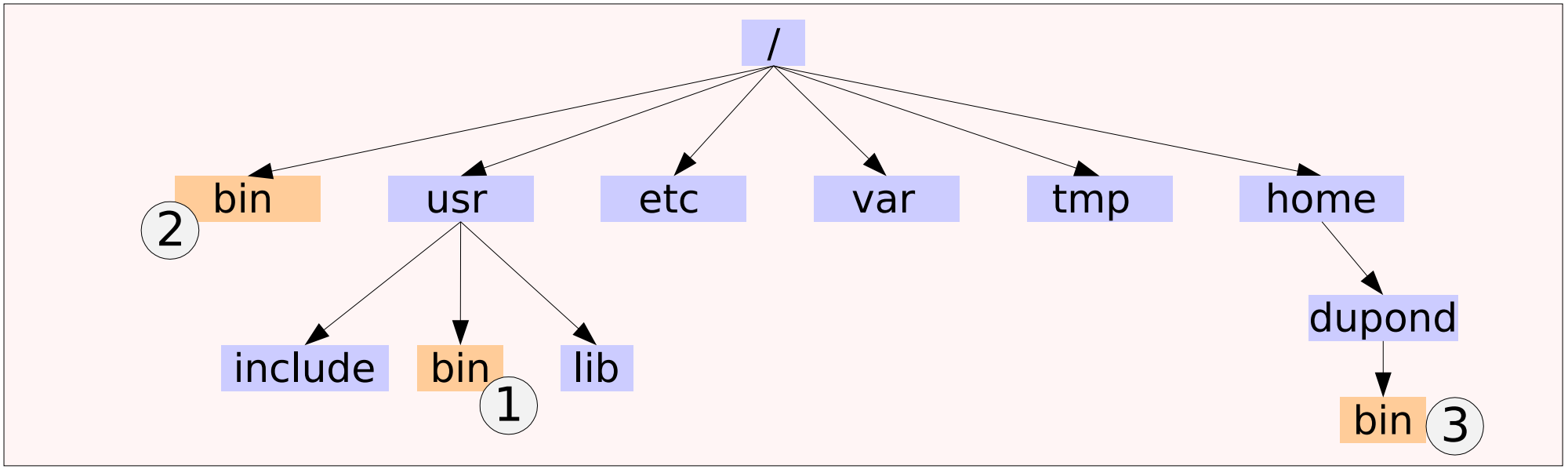
L'ordre est important!

liste de répertoires où le système cherche des programmes

séparateur ":"

```
/usr/bin:/bin:/home/dupond/bin
```

1 2 3



la variable PATH

la commande "which"

```
[dupond@r10102 ~]# which ls
/usr/bin
```

changer la variable PATH

```
[dupond@r10102 ~]# echo $PATH
/usr/bin:/bin
[dupond@r10102 ~]# export PATH=$PATH:/home/dupond/bin
[dupond@r10102 ~]# echo $PATH
/usr/bin:/bin:/home/dupond/bin
```

script shell : exécution

```
[dupond@r10102 ~/test]# ls
essai.sh
[dupond@r10102 ~/test]# essai.sh
bash: essai.sh: command not found
```

solutions:

```
./essai.sh
```

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

$PATH

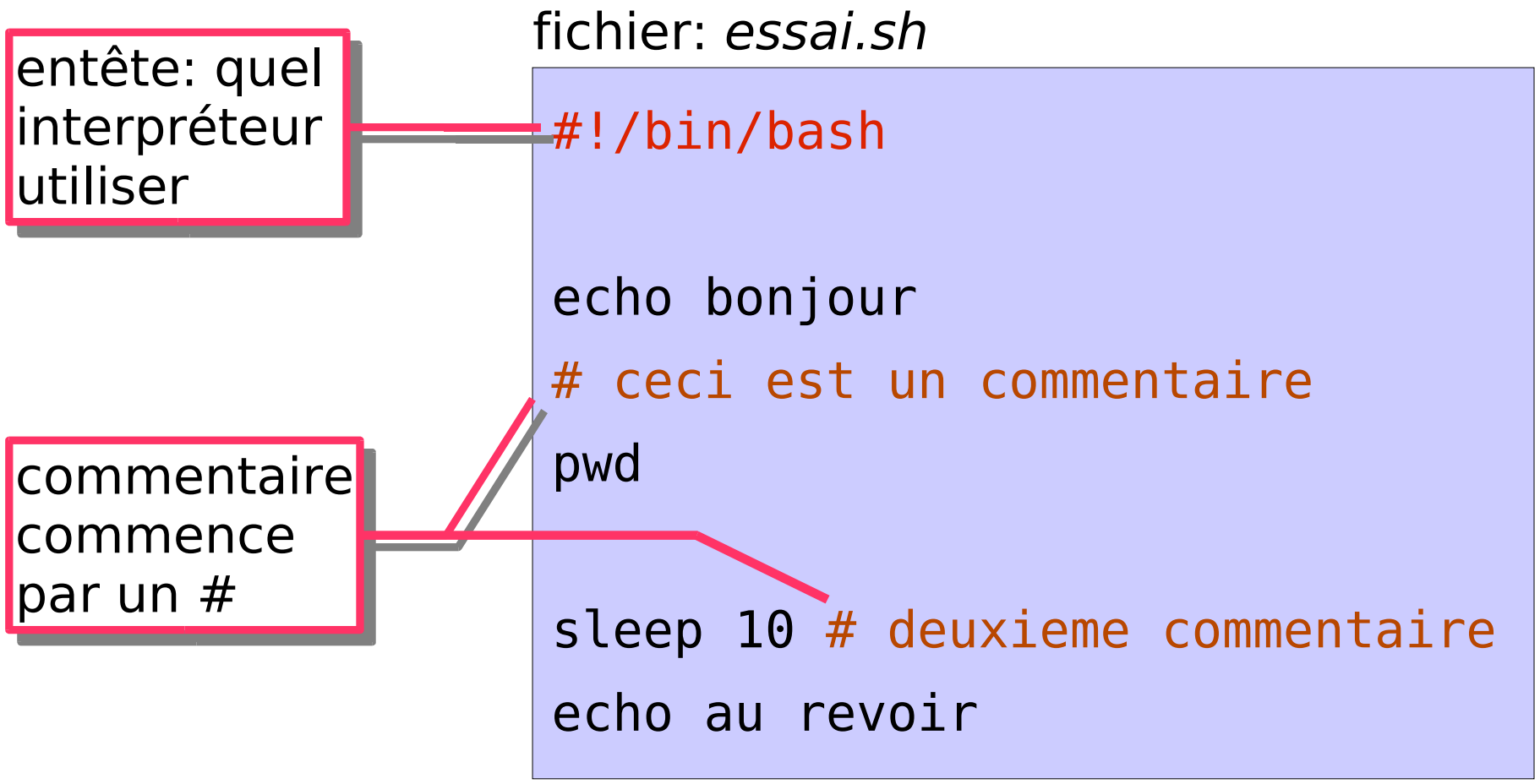
```
/usr/bin:/bin:/home/dupond/test
```

$PATH

```
/usr/bin:/bin:.
```

dangereux

script shell : entête, commentaires

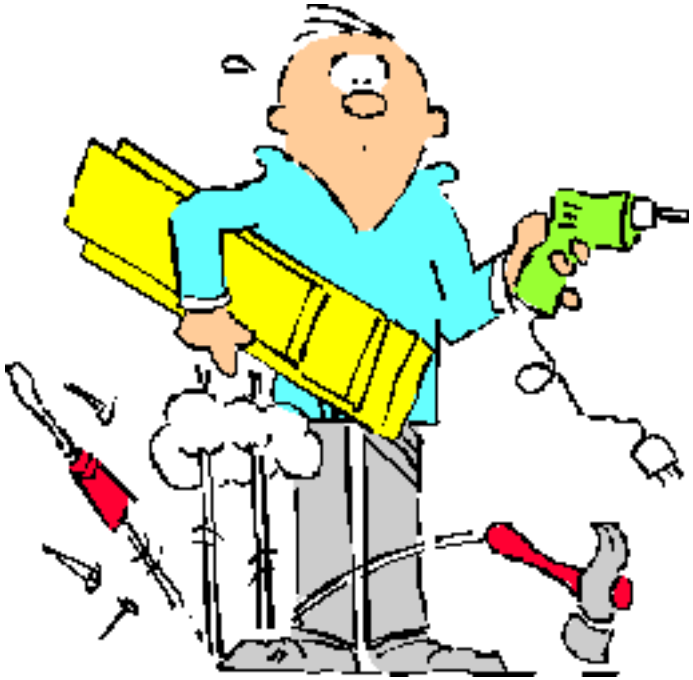


personnalisation du bash

fichiers: *~/.bashrc ou ~/.bash_profile*

```
# ajouter mon répertoire à PATH
export PATH=$PATH:$HOME/bin
# des raccourcis de commandes
alias la='ls -la'
alias grep='grep -i'
alias allps='ps -ef'
# changer la langue
export LANG="en_US.iso88591"
```

partie pratique



1ère partie

caractères spéciaux

le problème : exemple

nom de fichier: `mon fichier`

espace

```
[dupond@r10102 ~]# ls -l mon fichier
ls: mon:     Aucun fichier ou répertoire de ce type
ls: fichier: Aucun fichier ou répertoire de ce type
[dupond@r10102 ~]#
```

espace: séparateur entre arguments!

le problème : exemple

dollar: `echo ca coute $5`

$

```
[dupond@r10102 ~]# echo ca coute $5
ca coute
[dupond@r10102 ~]#
```

$5 : c'est une variable!

le problème : exemple

commande find: `find . -name *.png`


```
[dupond@r10102 ~]# find . -name *.png
find: les chemins doivent précéder l'expression
Usage: find [-H] [-L] [-P] [CHEMIN...] [EXPRESSION]
[dupond@r10102 ~]#
```

* : interprété par bash **AVANT** d'être passé à find !

échappement

quelques caractères spéciaux

```
espace
$
*
!
;
[
(
)
&
#
etc.
```

bash interprète certains caractères

échapper à l'interprétation

```
" : mon fichier → "mon fichier"

' : mon fichier → 'mon fichier'

\ : mon fichier →  mon\ fichier
```

échappement

échapper à l'interprétation

`"xxx"`	échappement partiel
`'xxx'`	échappement complet
`\x`	échappement d'un seul caractère

échappement : exemples

```
[dupond@r10102 ~/echap]# ls
a.png b.png
[dupond@r10102 ~/echap]# echo exemple: *.png
exemple: a.png b.png
[dupond@r10102 ~/echap]# echo "exemple: *.png"
exemple: *.png
[dupond@r10102 ~/echap]# echo 'exemple: *.png'
exemple: *.png
[dupond@r10102 ~/echap]# echo exemple: \*.png
exemple: *.png
[dupond@r10102 ~/echap]#
```

échappement : exemples

échappement partiel / complet

```
[dupond@r10102 ~/echap]# cher=500
[dupond@r10102 ~/echap]# echo tres $cher
tres 500
[dupond@r10102 ~/echap]# echo "tres $cher"
tres 500
[dupond@r10102 ~/echap]# echo 'tres $cher'
tres $cher
```

2ème partie

quelques commandes

la commande: **top**

afficher la liste de processus interactivement

```
  PID USER      NI    RES S %CPU %MEM     TIME+  COMMAND
 3848 root       0   88m S 44.5  4.4   9:29.61 XFree86
 4401 bosc       0   13m S  2.9  0.7   0:18.05 gnome-termi
 4487 bosc       0  133m S  0.6  6.6  12:33.99 soffice.bin
 3372 root       0  2620 S  0.2  0.1   0:12.40 cupsd
 4408 bosc       0  8280 S  0.2  0.4   0:07.54 clock-applet
20459 bosc       0   55m S  0.2  2.8   0:06.43 acroread
20881 bosc       0  1092 R  0.2  0.1   0:00.45 top
    1 root       0   512 S  0.0  0.0   0:00.60 init
    2 root       0     0 S  0.0  0.0   0:00.00 migration/0
    3 root      19     0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root     -10     0 S  0.0  0.0   0:03.30 events/0
    5 root     -10     0 S  0.0  0.0   0:00.00 khelper
    6 root     -10     0 S  0.0  0.0   0:00.00 kacpid
   60 root     -10     0 S  0.0  0.0   0:00.21 kblockd/0
```

la commande: **cat**

afficher un ou plusieurs fichier

très simple!

essai.txt

```
bonjour toto
bla bla bla
```

```
[dupond@r10102 ~/essai]# cat essai.txt
bonjour toto
bla bla bla
[dupond@r10102 ~/essai]# cat essai.txt | grep bla
bla bla bla
[dupond@r10102 ~/essai]#
```

la commande: **find**

rappel

rechercher des fichiers dans une arborescence

rechercher les fichiers terminant par .jpg

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

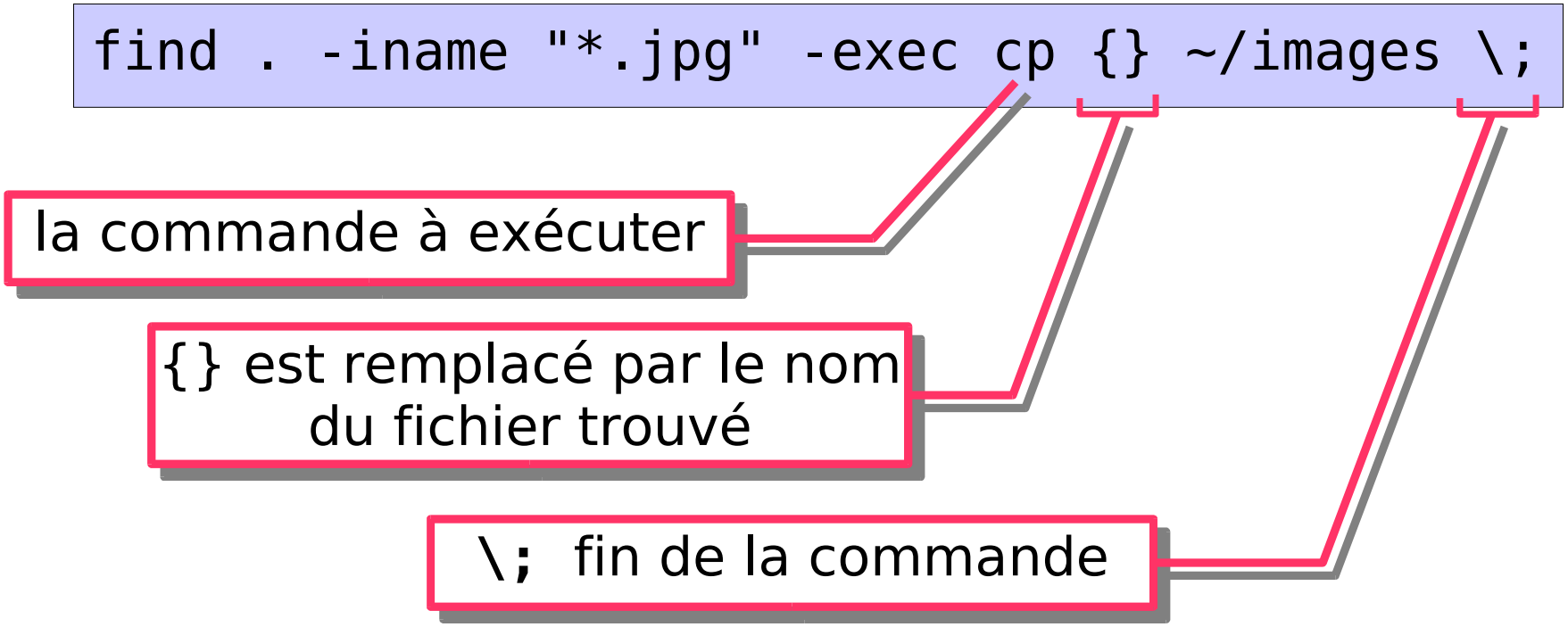
rechercher les fichiers .h contenant "color" dans /usr/include

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

commande très utile!

la commande: **find**

-exec : exécuter une commande à chaque fichier trouvé



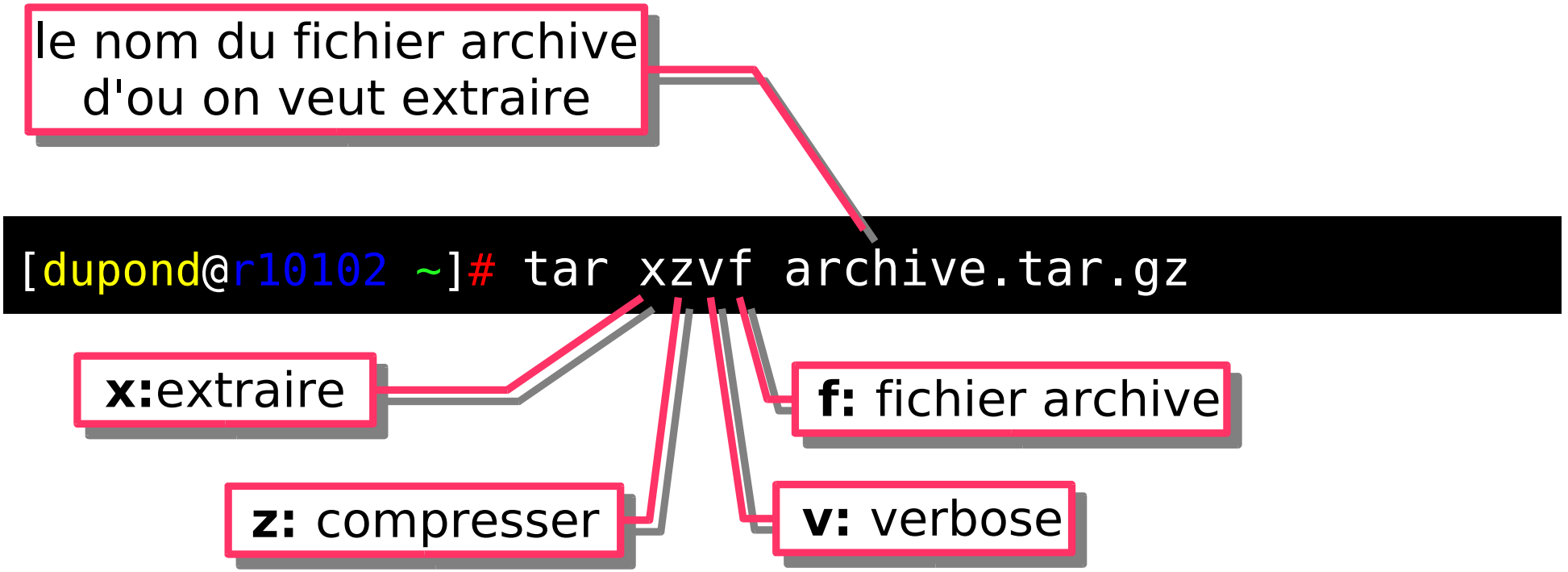
exemple:
« find » trouve deux fichiers gnome.jpg et tux.jpg :

```
cp gnome.jpg ~/images
cp tux.jpg   ~/images
```

la commande: **tar**

manipuler des archives

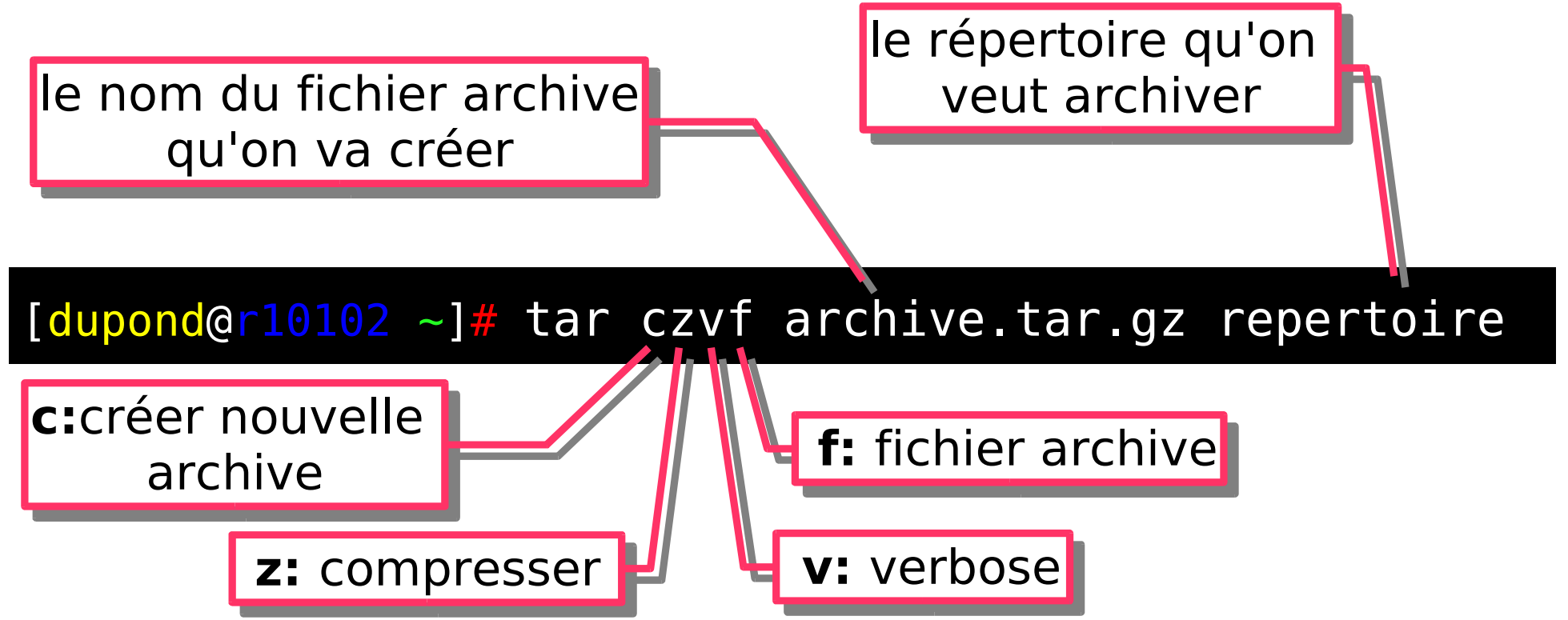
désarchiver



la commande: **tar**

manipuler des archives

créer une archive



la commande: **sort**

trier des données

texte.txt

```
bonjour
au revoir
courir
```

numeros.txt

```
100 /home
30  /usr
40  /var
```

```
[dupond@r10102 ~/essai]# sort texte.txt
au revoir
bonjour
courir
[dupond@r10102 ~/essai]# sort -n numeros.txt
30  /usr
40  /var
100 /home
[dupond@r10102 ~/essai]#
```

tri alphabétique

n: tri numérique

la commande: **cut**

afficher certains champs d'une ligne

numeros.txt

```
100 /home,x
30  /usr ,y
40  /var ,z
```

d: délimiteur entre champs

f: numéro du champs

```
[dupond@r10102 ~/essai]# cut -d ' ' -f 1 numeros.txt
100
30
40
[dupond@r10102 ~/essai]# cut -d ',' -f 2 numeros.txt
x
y
z
[dupond@r10102 ~/essai]#
```

ce document est distribué librement :

- sous licence GNU FDL :

  http://www.gnu.org/copyleft/fdl.html

- les originaux sont disponibles aux formats OpenOffice et powerpoint

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

Marcel.Bosc@iutv.univ-paris13.fr