Système

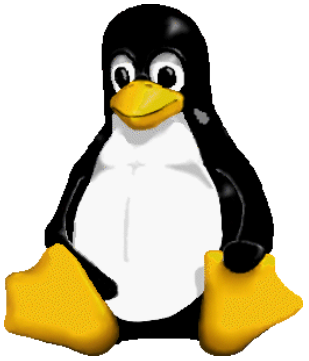
Cours n°3

Isis TRUCK,
Professeur des universités

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

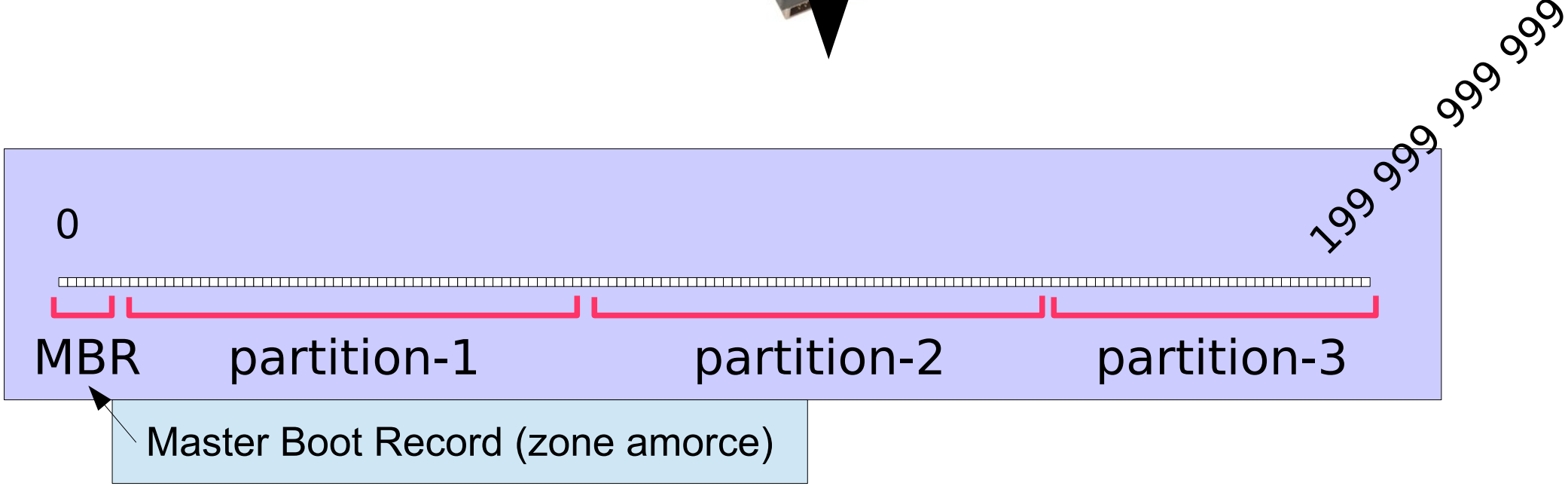
Plan

- partitions et systèmes de fichiers
- arborescence UNIX
- montage de disques
- répertoires et droits d'accès
- shell : configuration et scripts
- divers
- caractères spéciaux
- quelques commandes

partitions et systèmes de fichiers

partitions d'un disque dur (sorte de *mémoire de masse*)

disque dur
200 Giga Octets



comment gérer les fichiers?

- Au sein de chaque partition, il peut y avoir **conflit** entre les différents fichiers (chacun doit y avoir une place qui lui est propre, sinon, risque d'écrasement mutuel)

- C'est le rôle du SGF : système de gestion de fichiers

systèmes de fichiers (*file systems*)

GNU/Linux

ext4
ext3
...

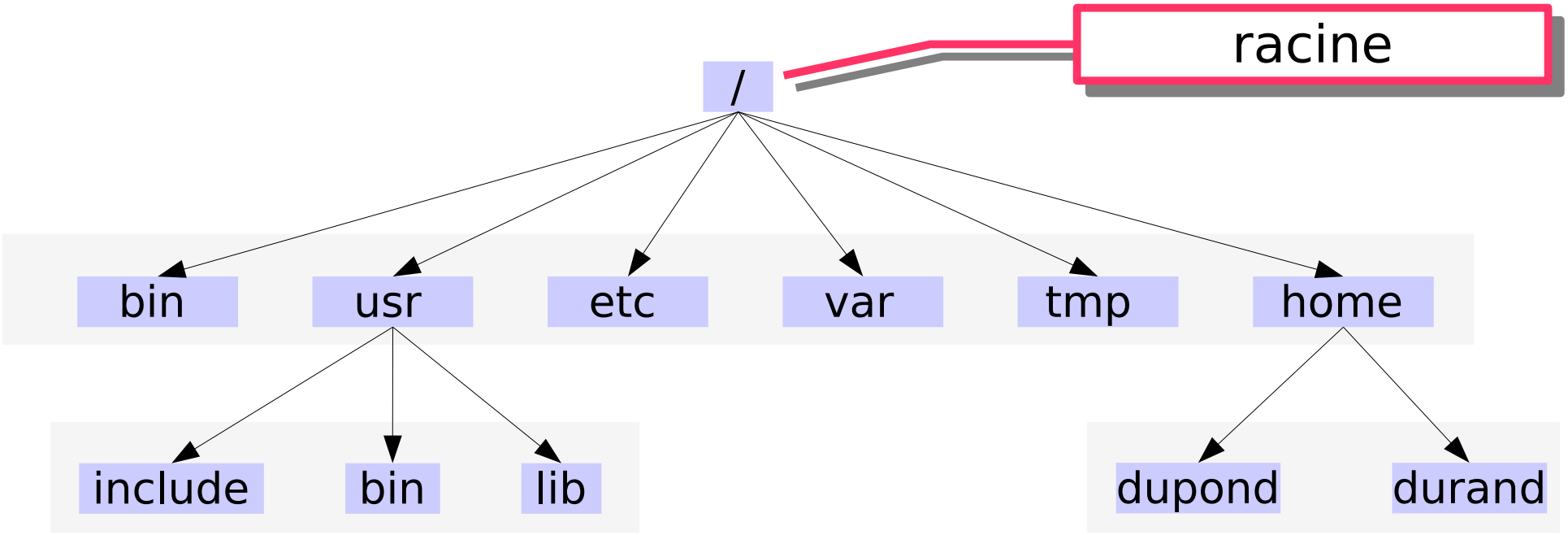
hfs plus

FAT
NTFS

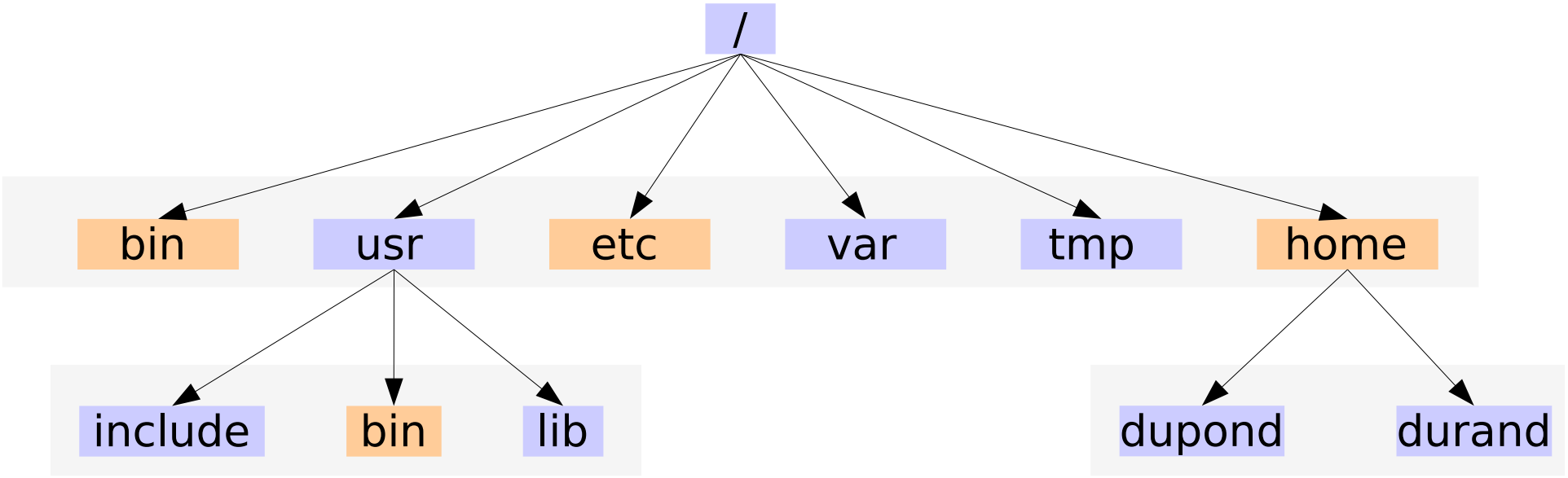
(partition swap => partition contenant la mémoire virtuelle nécessaire, entre autre, au processeur lorsque sa mémoire vive n'est plus suffisante)

arborescence UNIX

répertoires UNIX importants



répertoires UNIX importants

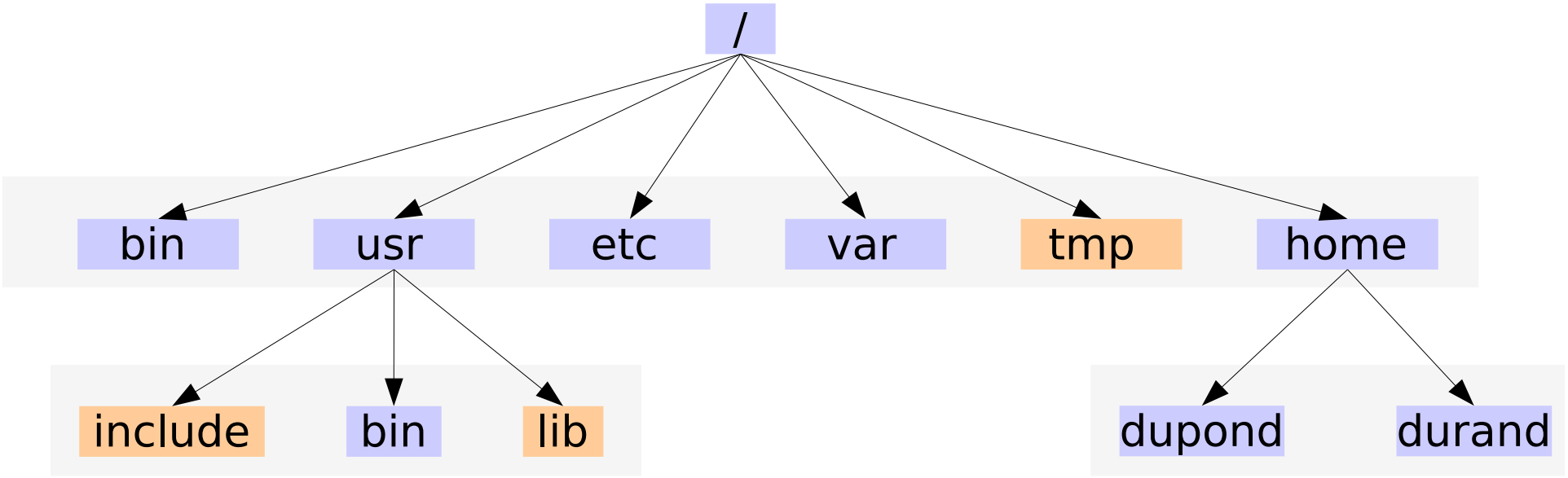


/bin et /usr/bin : programmes exécutables

/etc : configuration du système

/home : répertoires personnels des utilisateurs

répertoires UNIX importants



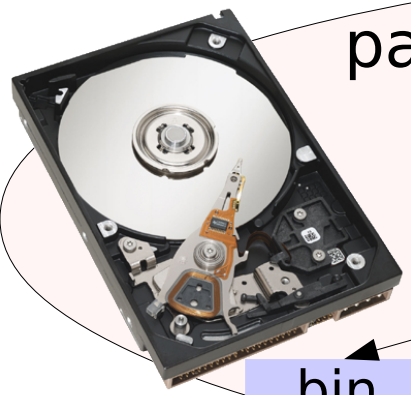
/usr/include : entête (header) programme C/C++

/usr/lib : librairies

/tmp : fichiers temporaires

montage de disques

emplacement logique



partition 1
(sda1)

/

bin

usr

etc

var

tmp

home

include

bin

lib

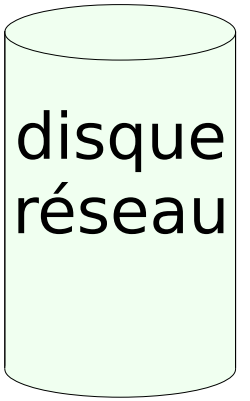
dupond

durand

partition 2
(sda2)

disque
réseau

emplacement physique des fichiers



hdc1

sda1

serveur:/share

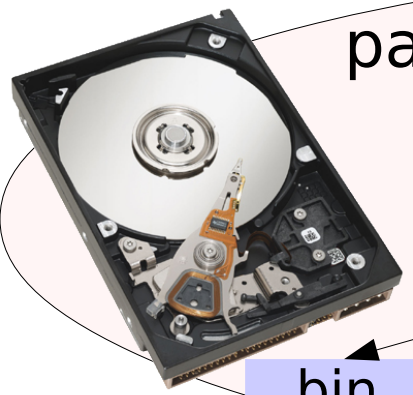
partitions 1,2,3:
sda1,sda2,sda3

noms des périphériques

sd = storage drive (SATA ou SCSI ou USB disk)
a = 1er disque enregistré
hd = hard drive (en IDE) : auparavant, utilisé pour les disques durs uniquement, maintenant pour tous les drives connectés en IDE

comment y accéder?

montage de disques



partition 1
(sda1)

/

bin usr etc var tmp home

include bin lib

partition 2
(sda2)

```
[root@r101 ~]# mount /dev/sda2 /usr
[root@r101 ~]#
```

shell: configuration et scripts

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

variables d'environnement

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

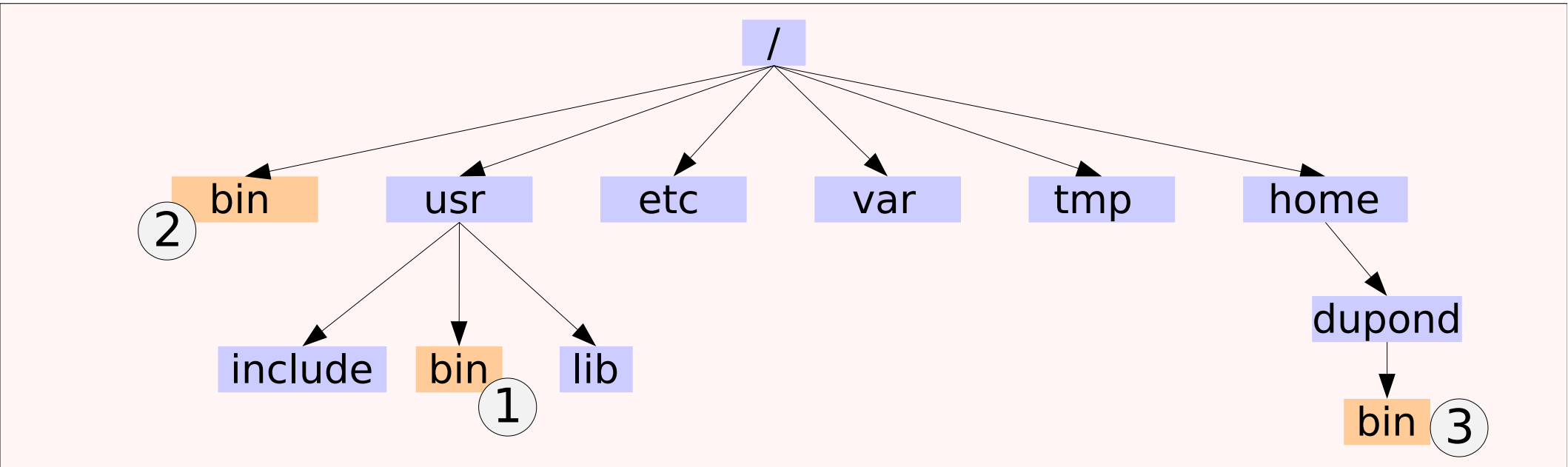
L'ordre est important!

liste de répertoires où le système cherche des programmes et commandes

séparateur ":"

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

1 2 3



la variable PATH

la commande "which"

indique le chemin absolu du fichier en argument

```
[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
```

Variables d'environnement : principes de fonctionnement

- **Portée des variables : locale**
  - Leur valeur est spécifique au processus dans lequel ou pour lequel elles ont été définies
  - Si on ouvre 2 terminaux différents (2 processus **bash** différents), et que l'on change la valeur d'une variable d'environnement dans un terminal, il n'y aura **pas d'impact** dans l'autre terminal.
- **Héritage**
  - Lorsqu'un processus enfant est créé à partir d'un processus parent, le processus enfant hérite de toutes les variables du processus parent, avec leurs valeurs. Par exemple, si on lance **gedit** depuis un terminal (**bash)**, alors **gedit** héritera des valeurs des variables d'environnement connues dans bash
  - Exemple : si l'on définit la valeur de la variable d'environnement « LANG » dans un terminal, et qu'on lance **gedit** depuis le même terminal, **gedit** héritera de la nouvelle valeur de la variable LANG, et s'affichera donc dans une langue différente du reste des applications du système.
  - NB : du fait de la portée des variables, une fois le processus **gedit** lancé, les modifications de variables dans le processus parent ne seront pas répercutées sur le processus enfant et vice-versa.

script shell : exécution

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

solutions:

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

```
./essai.sh
```

$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.utf8"
```

divers

virtualisation

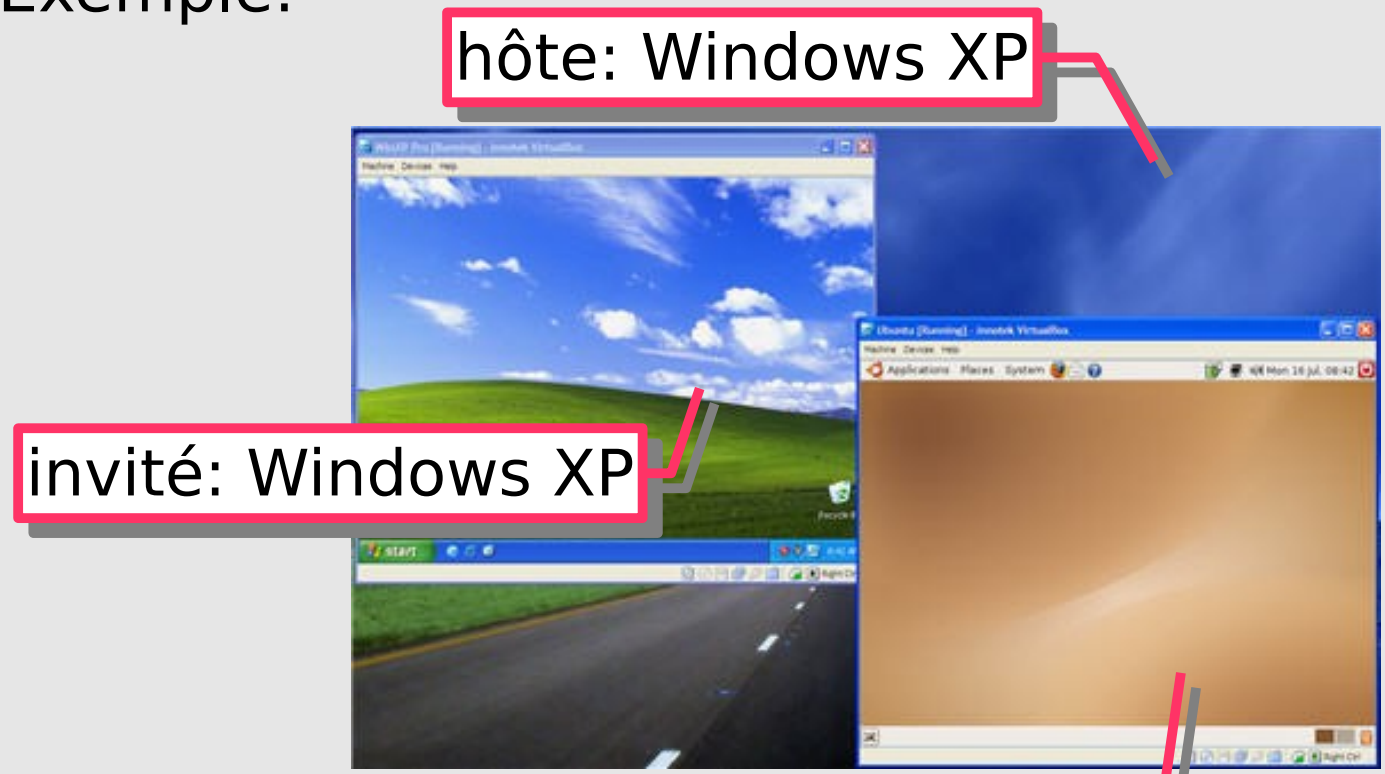
Virtualisation : exécuter un ou plusieurs systèmes **invités** dans un système d'exploitation **hôte**

très pratique! 

logiciels:

- VirtualBox (libre!)
- Parallels
- VMware

Exemple:



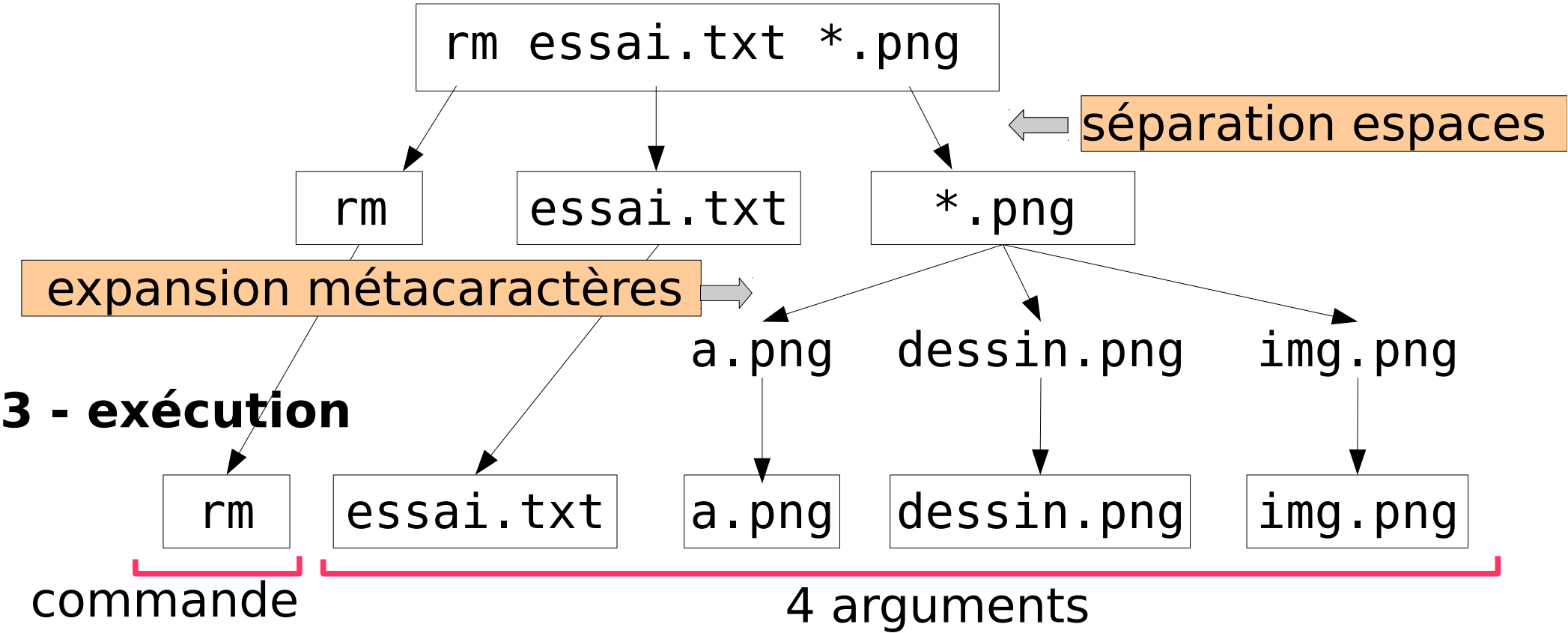
étapes de traitement

1 - saisie de la commande

```
[dupond@r10102 ~/echap]# rm essai.txt *.png
```

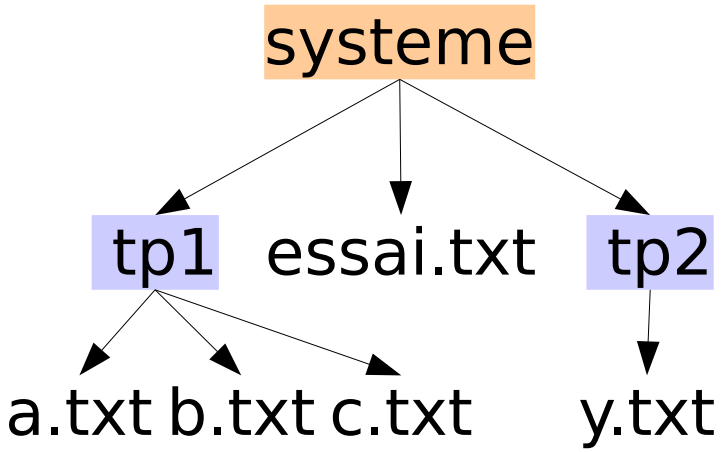
-> touche "entrée"

2 - interprétation de la saisie par le shell



3 - exécution

métacaractère "*"




```
rm *.txt
```

⇩

```
rm essai.txt
```

```
rm tp1/*.txt
```

⇩

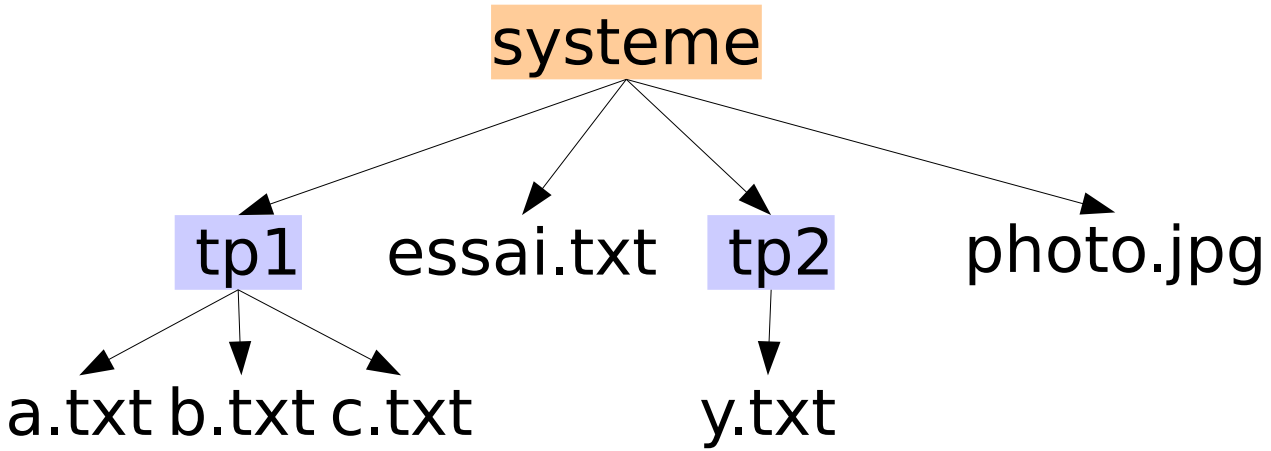
```
rm tp1/a.txt tp1/b.txt tp1/c.txt
```

```
rm */*.txt
```

⇩

```
rm tp1/a.txt tp1/b.txt tp1/c.txt tp2/y.txt
```

métacaractère "*" - suite



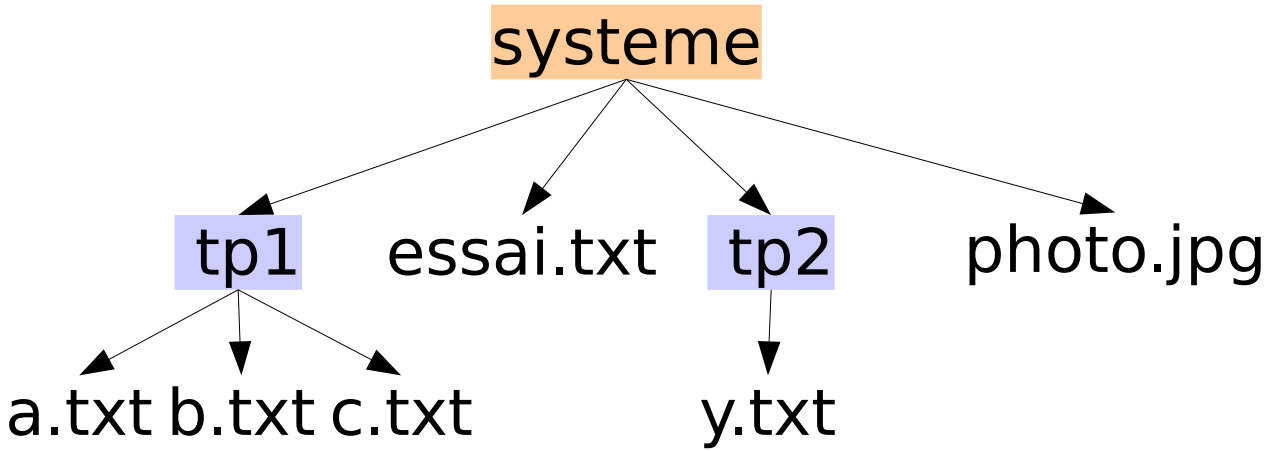

```
cp *.txt tp1/*.txt photo.jpg tp2
```

```
cp essai.txt tp1/a.txt tp1/b.txt tp1/c.txt photo.jpg tp2
```

sources

destination

métacaractère "*" - suite




```
cp *.txt tp1/*.txt photo.jpg tp2
```

```
cp essai.txt tp1/a.txt tp1/b.txt tp1/c.txt photo.jpg tp2
```

sources

destination

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!

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 : 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 « 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

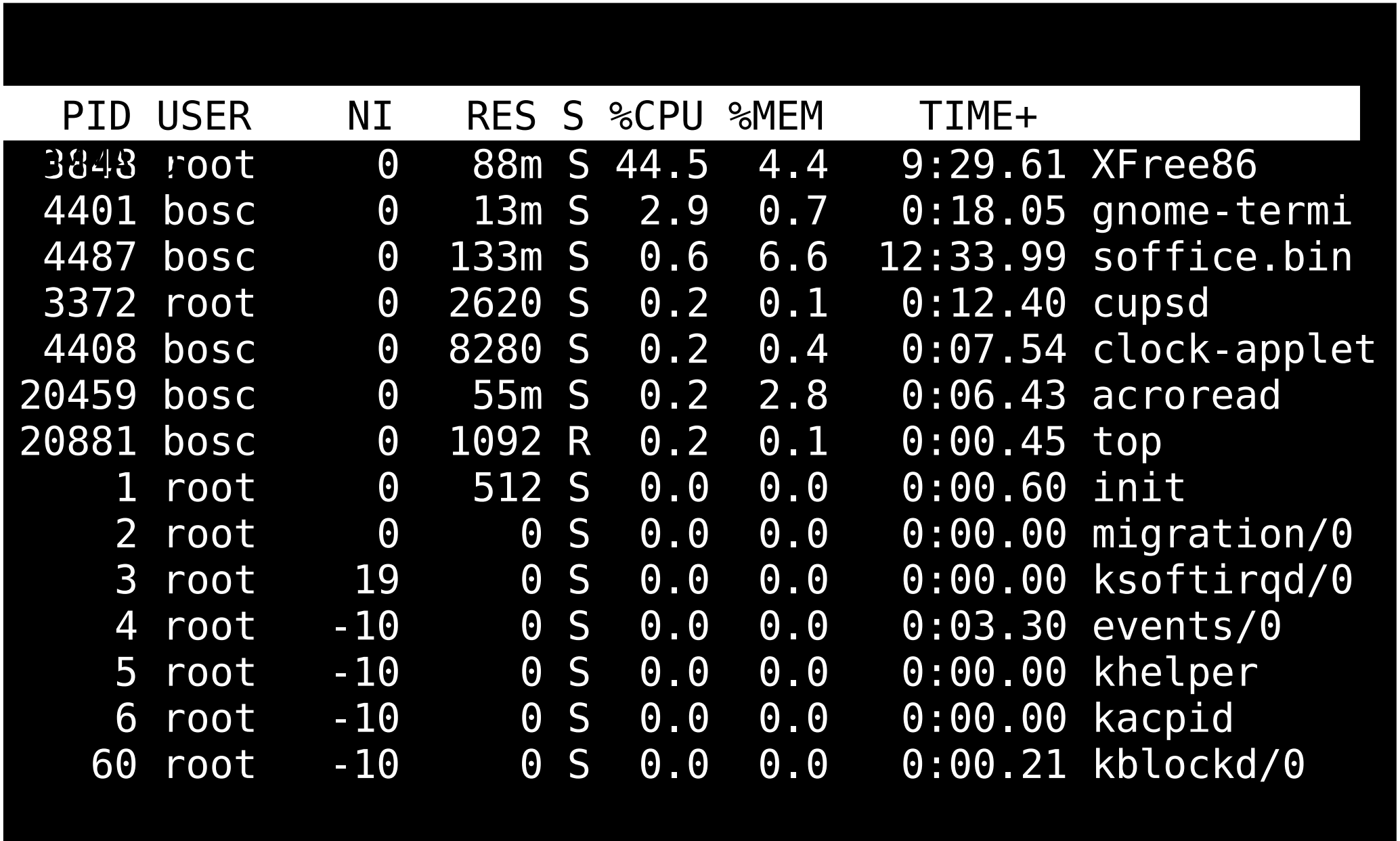
sda1: 14Go → /
sda3: 58Go → /home

sdb

sdb1:151Go → /disk2

la commande: **top**

afficher la liste de processus interactivement



PID	USER	NI	RES	S	%CPU	%MEM	TIME+	
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: **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: **sort** (suite)

trier des données

- `-d` trie alphabétiquement (*dictionary*)
- `-r` inverse le tri (*reverse*)
- On peut trier sur **un certain champ** (option `-k`) mais il faut évidemment préciser le délimiteur (option `-t`)
  - Exemple : `sort -k2n -t ':' fichierATrier.txt`
  - Ici, `-k2` signifie de prendre en compte tous les champs depuis le 2e jusqu'au dernier
  - Donc pour trier sur un seul champ, il faut écrire : `-k2,2` . Cela signifie d'utiliser les champs de 2 à 2, donc seulement le 2e champ.
- On peut trier sur **plusieurs champs** avec un ordre, en utilisant plusieurs fois l'option `-k`. Si l'on veut un tri sur un seul champ à chaque fois, on écrira :
  - Exemple : `sort -k3,3n -k1,1rd -t ':' fichierATrier.txt`

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 champ

```
[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]#
```

la commande: **cut** (suite)

- `-c` permet de sélectionner des colonnes (ou caractères), sans précision de délimiteur
- Exemples :
  - `cut -c2-5 fichierADecouper.txt` permet de sélectionner les colonnes 2 à 5
  - `-c14-` permet de sélectionner de la colonne 14 à la dernière
  - `-c1-3,14-18` permet de spécifier plusieurs plages de colonnes