

Introduction aux systèmes d'exploitation 2/2



V. Berry

Polytech - Université Montpellier

Programme

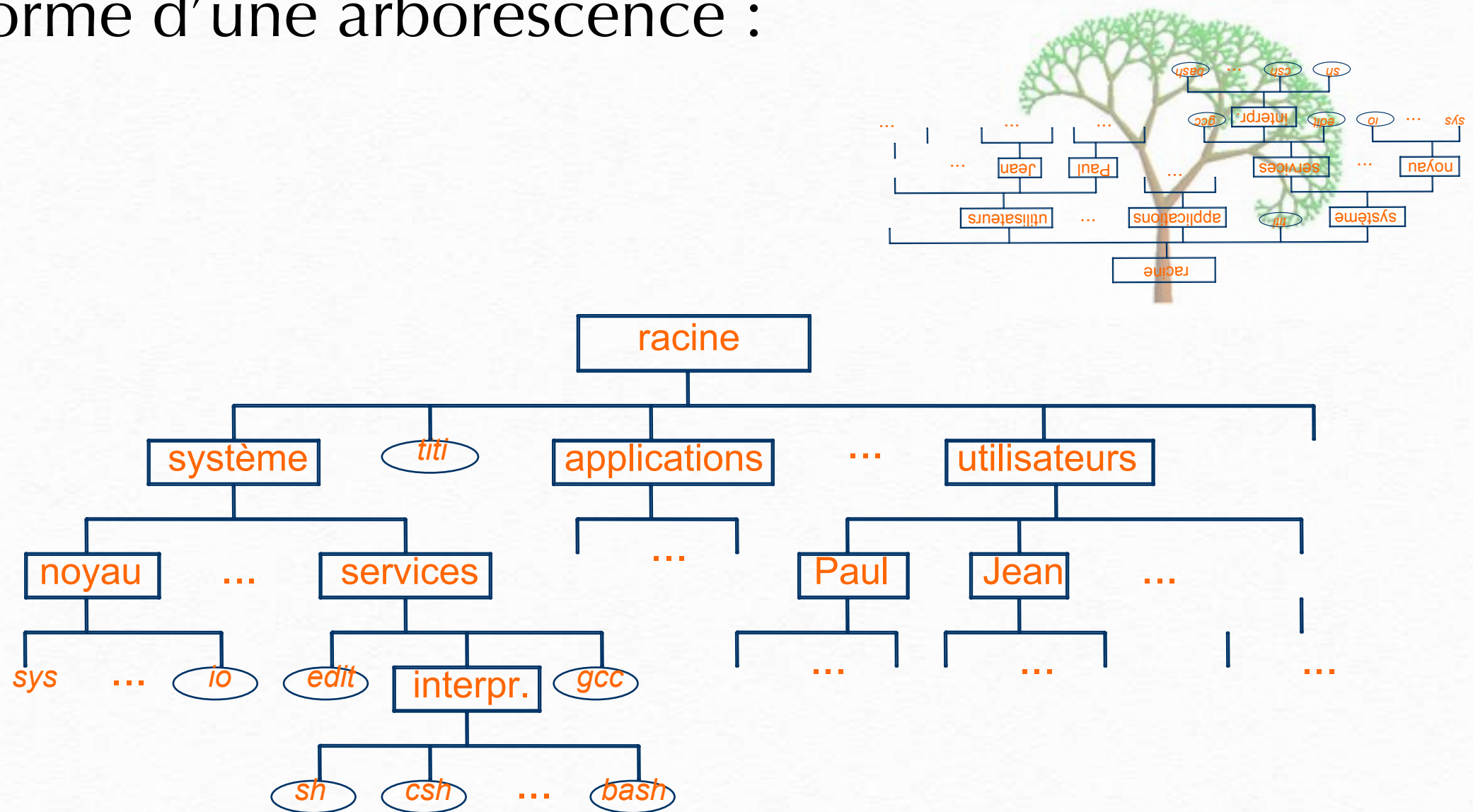
- ☒ Notion de système d'exploitation
- ☒ Architecture d'un système Unix
- ☐ Organisation des fichiers
- ☐ Commandes unix sur les fichiers
- ☐ Deux éditeurs de texte



Organisation des fichiers (Unix)

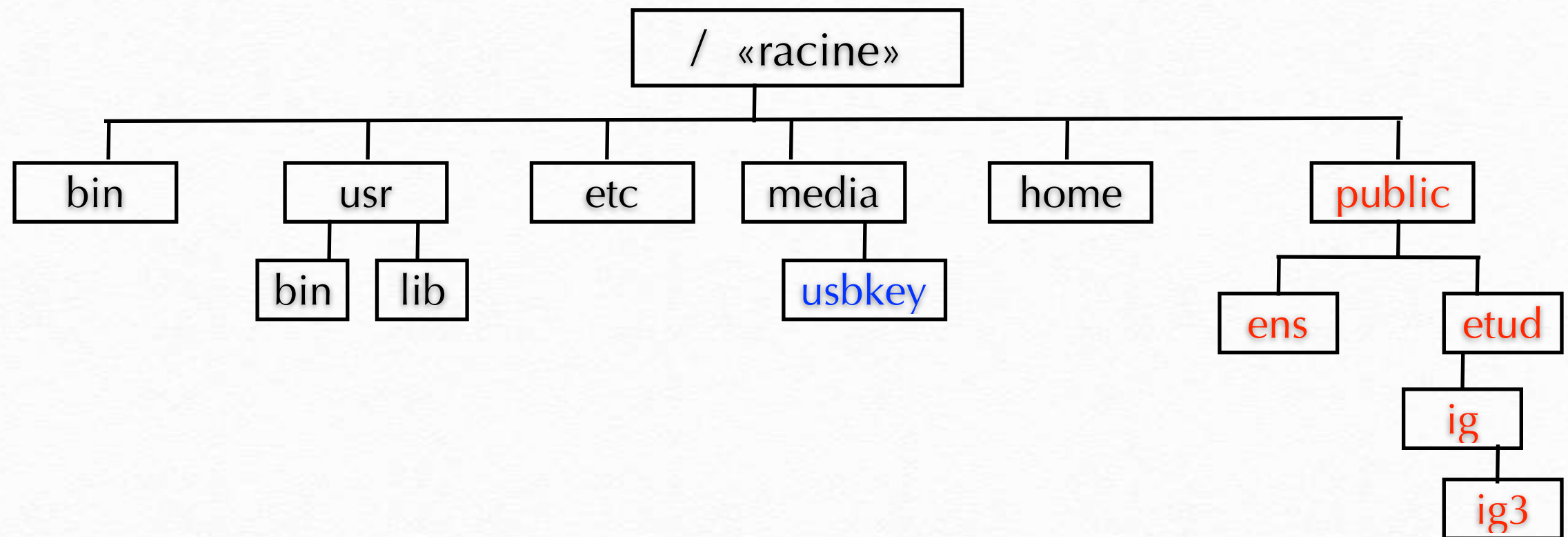
Organisation hiérarchique

Les **répertoires** & **fichiers** sont organisés sous la forme d'une arborescence :



Montages

Les **supports amovibles** (zip, cd,dvd,clé USB) et les **disques réseaux** contiennent aussi une arborescence de fichiers qui est « greffée » sur l'arborescence générale



Tout utilisateur a une **arborescence personnelle**, montée depuis le **réseau**, greffée dans l'arborescence du système Linux présent sur la machine

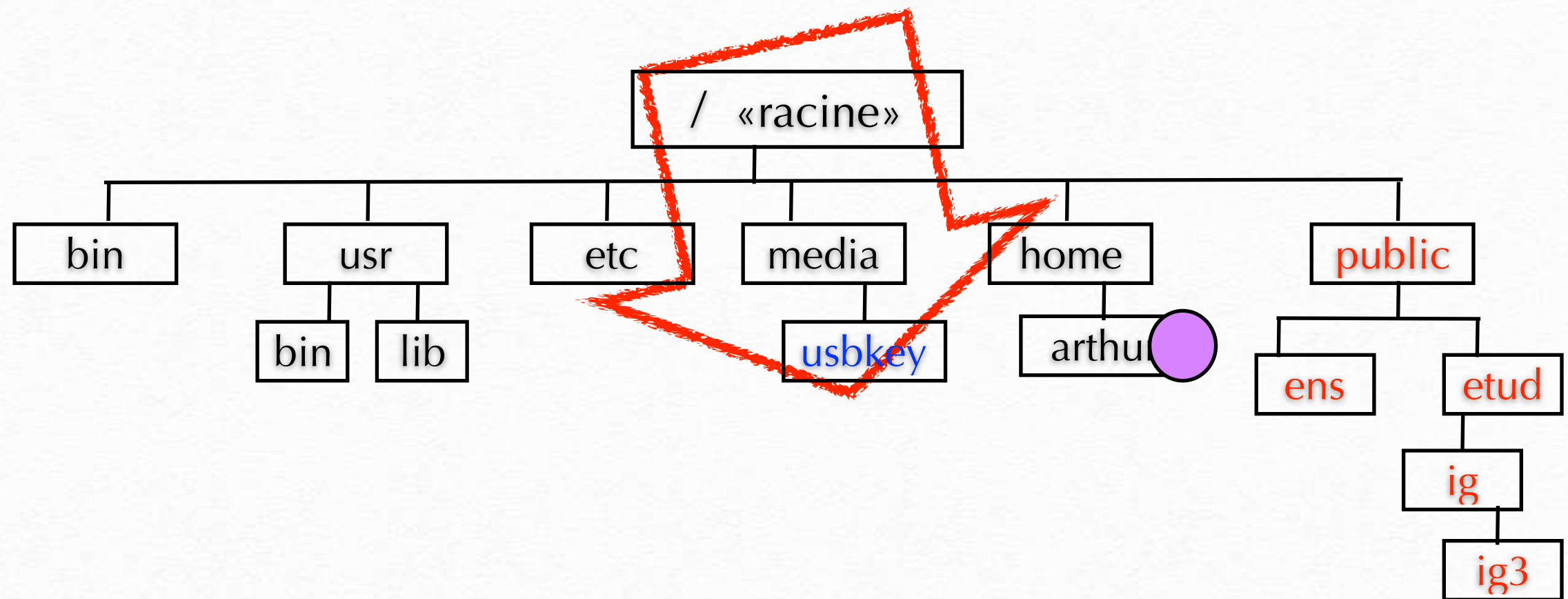
/public ou **/etudiant**

Désignations



Chaque fichier et répertoire peut-être désigné

- par un **chemin absolu** (depuis la racine, notée «/»)



- ou par un **chemin relatif** (depuis un répertoire de travail)



chemin relatif au répertoire de départ

Protection des informations

- Le problème : permettre un **partage** de fichiers entre utilisateurs mais aussi une **protection**
- L'ensemble des opérations autorisées sur un objet constitue ses **droits d'accès** (donnés par le propriétaire)
- Les droits peuvent **évoluer** dans le temps (*ex : un corrigé lisible par les étudiants*)



Les droits d'accès

Tout fichier ou répertoire a un **utilisateur propriétaire** et un **groupe propriétaire**

(par défaut, le créateur du fichier et le groupe auquel appartient le créateur du fichier)

Les droits d'accès à un fichier (ou répertoire) sont paramétrables par le propriétaire du fichier pour trois classes d'utilisateurs (**ugo**)

- lui-même = le propriétaire du fichier (**user**)



- les membres du groupe propriétaire (**group**)



- les autres utilisateurs du système (**others**)





Les droits d'accès

Trois **types d'accès** sont possibles (**rwX**). Ces accès ont un sens différents suivant qu'ils s'appliquent à un **répertoire** ou un **fichier**



	fichier	dossier
r	permet	permet
w		
x		

Les droits d'accès

Les droits des fichiers et répertoires peuvent être affichés par une commande Linux particulière (*cf partie sur le Terminal*).

```
-rw-rw-r-- . . . . . fic1.txt
```

10 caractères sont associés à chaque fichier et répertoire :

- le 1^{er} = nature du fichier (**d** pour répertoire et **tiret (-)** pour fichier simple)
- les 9 suivants = **droits** attribués à ce fichier ou rép.
(3 pour le propriétaire, 3 pour le groupe, 3 pour les autres)

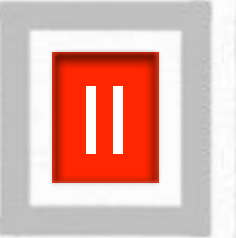
Chaque groupe de 3 précise **dans l'ordre** :

les droits en lecture, en écriture, enfin en exécution

Pour chaque droit, si **sa lettre apparaît** = le droit est donné,

si il y a un tiret (-) = le droit est refusé.

Les droits d'accès



Exemple :

Type droits		prop.	groupe	taille	date_modif.	nom
<code>-rw-rw-r--</code>	1	berry	ens	502	sept 2 9:40	fic1.txt
<code>-rwx--x--x</code>	1	igor	ig	205	aout 3 8:03	proj.tar
<code>drwxr-x---</code>	2	fiorio	dir	4096	sept 1 7:35	UNIX

- *fic1.txt* est
- *proj.tar* est
- *UNIX* est un répertoire qui est


```
Terminal — bash — 79x24
ls /usr/local/
ite      mysql
        mysql-5.1.28-rc-osx10.5-x86
lude     share
        texlive

find /usr/local/share -name README -print 2>/dev/null
r/local/share/ghostscript/8.57/doc/README
r/local/share/ghostscript/fonts/README
r/local/share/taxomanie/ol-taxo/README
r/local/share/taxomanie/taxomanie-1.0/README
grep -E ".htm" /usr/local/share/ghostscript/**/*.README
Deprecated.htm
Devices.htm
  Helpers.htm
Humor.htm
Ps2epsi.htm
  Ps2pdf.htm
Ps2ps2.htm
Readme.htm
Unix-lpr.htm
Use.htm
Changes.htm
Commprod.htm
Fonts.htm
```

Commandes sur les fichiers

Commandes sur les fichiers

source -> destination

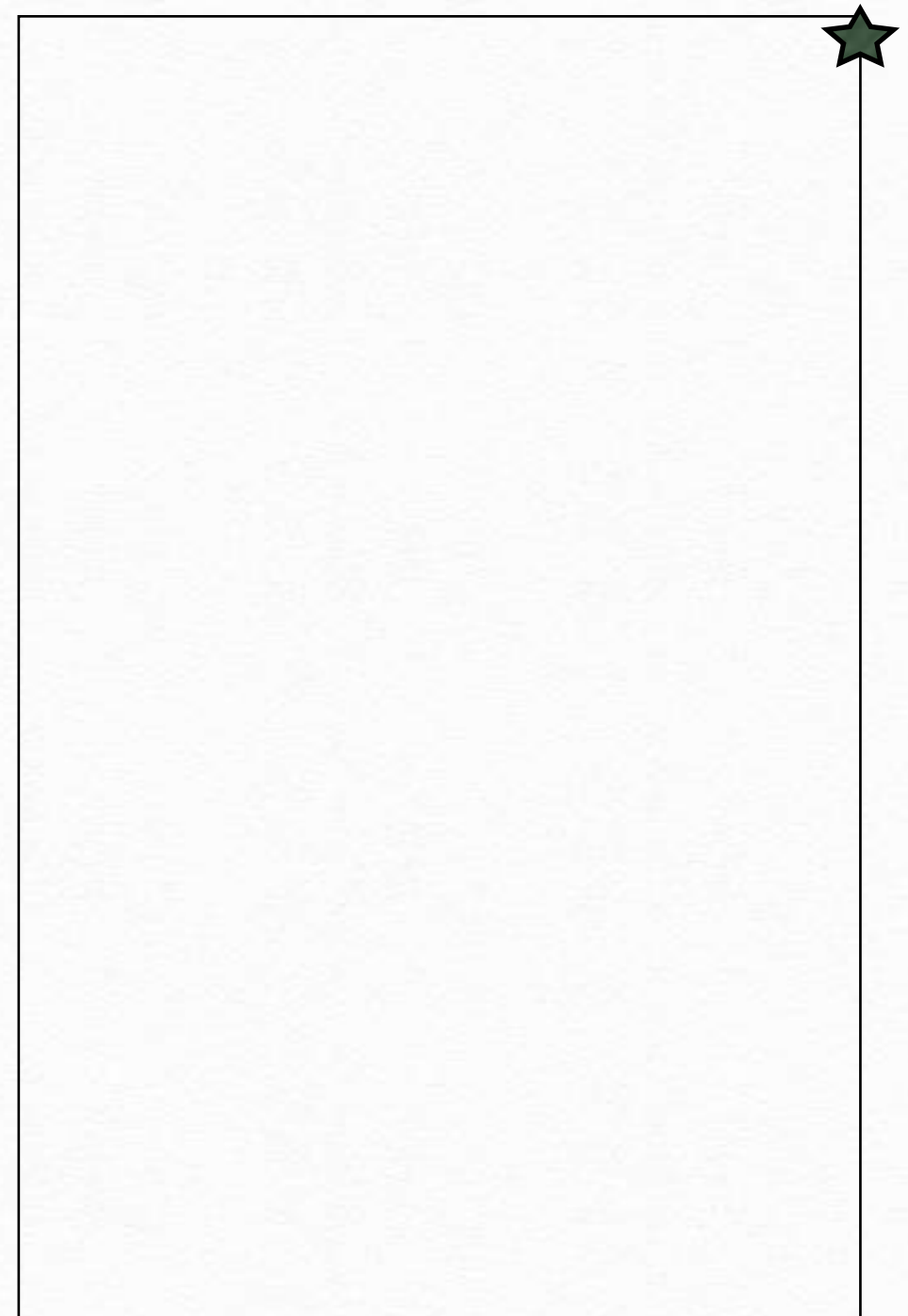
copier

déplacer/renommer

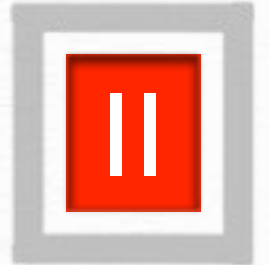
effacer

afficher le contenu

ou



Commandes sur les fichiers



Changer les droits sur un fichier : **chmod**

chmod <classe op perm, ...>|nnn <fic>

classe :

u : user
g : group
o : others
a : all

op :

= : affectation
- : suppr.
+ : ajout

perm :

r : lecture
w : écriture
x : exécution

chaque droit = une valeur = l'addition de

r	4
w	2
x	1
aucun	0

Exemples :

- Sur le fichier **tp1.txt** donner tous les droits à l'utilisateur, et lecture+exécution aux autres entités :

- Ajouter les droits d'exécutions à toutes les entités sur le fichier **script.py** :



Commandes sur les répertoires

le répertoire d'accueil : **~**

le répertoire courant : **.**

le répertoire parent : **..**

connaître le rép. courant : **pwd**

changer de répertoire : **cd**

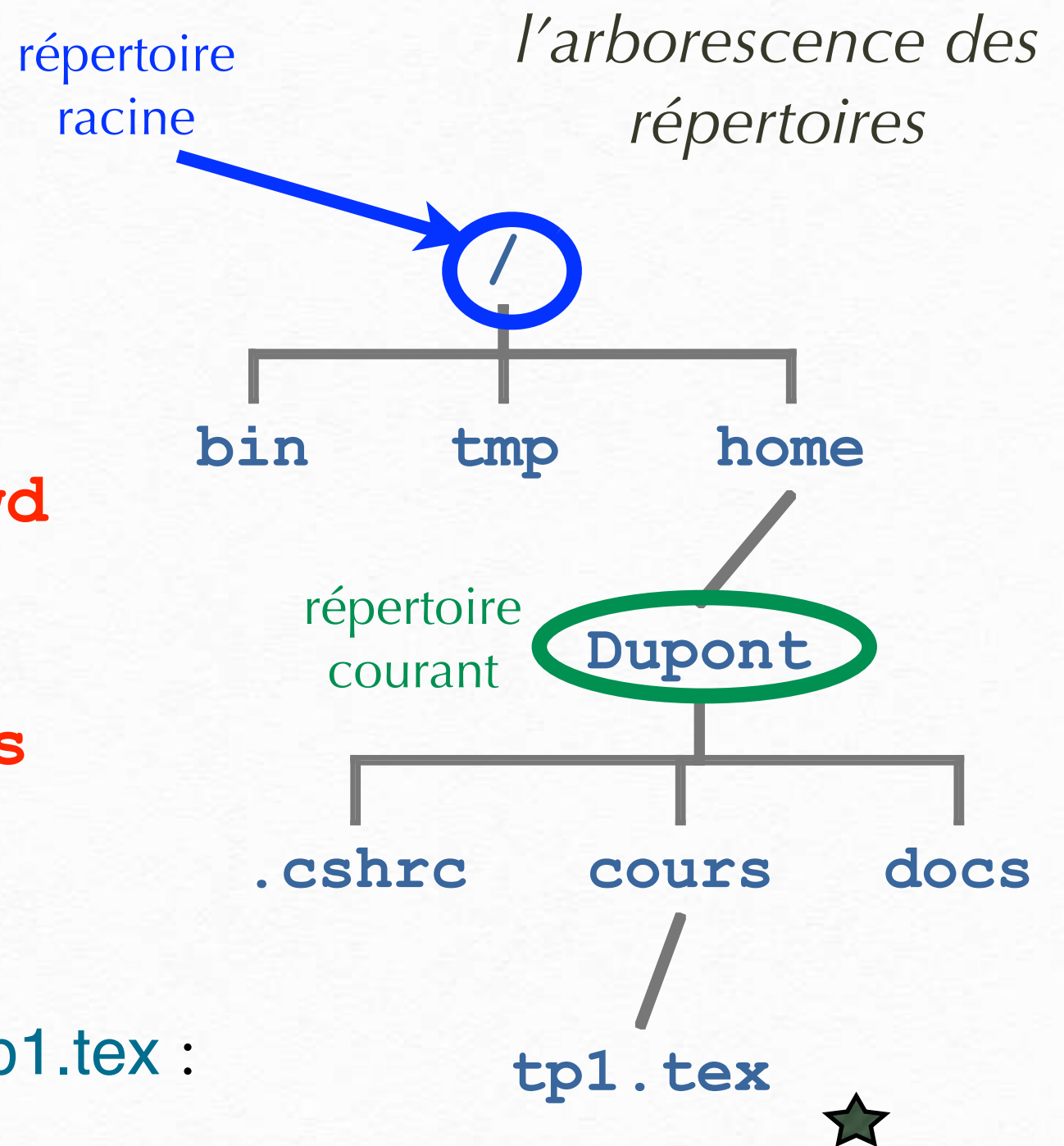
lister le contenu d'un rép. : **ls**

QUIZZ

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

relatif (à la racine):

absolu :



Méta-caractères du shell

Ce sont des caractères qui ont un sens spécial :

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

★ L'**espace** sert à séparer le **nom de commande** de ses **options** et **arguments**

★ L'astérisque ou étoile: *

- ▶ interprété comme toute suite de caractères alphanumériques
- ▶ utiliser **avec précaution** (commande rm par ex...)

★ Le point d'interrogation: ?

- ▶ remplace un seul caractère alphanumérique (ex: dans des noms de fichiers)

▶ Le point-virgule: ;

- ▶ Séparateur de commandes

★ L'anti-slash: \

- ▶ Inhibe la signification du méta-caractère qui suit

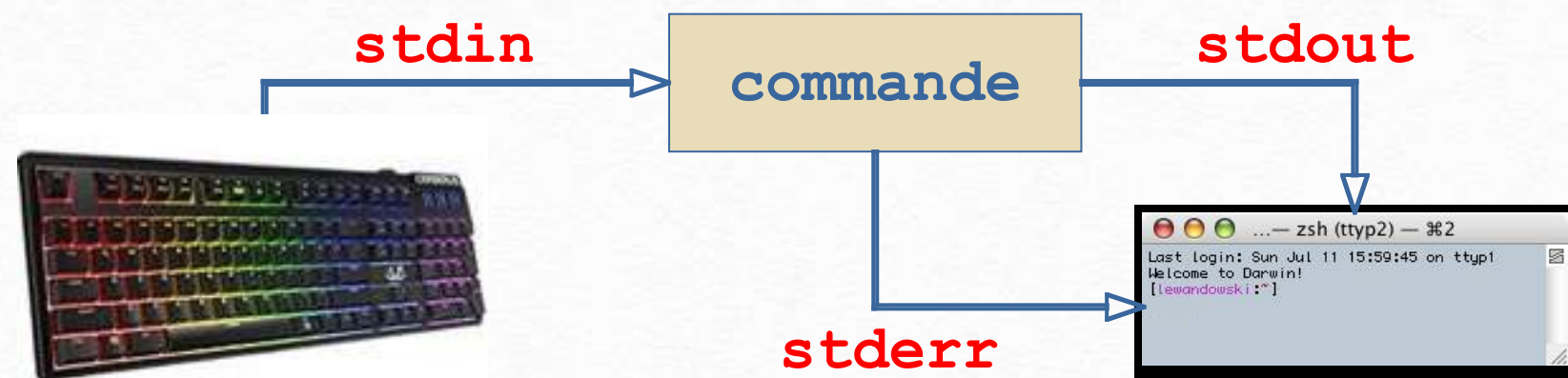
▶ Exemple: `ls Metallica\ Sonate\ 9\ si\ majeur.mp3`

\ suivi de **espace**

- ▶ Texte entre ' ' (simples **quotes**): le texte n'est pas interprété par le shell (donc peut contenir des caractères spéciaux), il est considéré comme **un seul mot**

Les re-directions

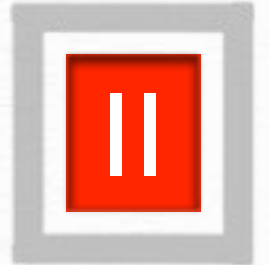
- Une commande utilise 3 *descripteurs* de fichiers ; par défaut :



- Une *redirection* consiste à remplacer un de ces canaux par défaut par/vers une autre commande ou un fichier

★ <f1	redirige le fichier f1 vers l'entrée standard
>f2	redirige la sortie standard dans le fichier f2
>>f2	redirige la sortie standard à la fin du fichier f2
c1 c2	redirige la sortie standard de la commande c1 vers l'entrée de c2
2>f1	redirige la sortie d'erreur dans le fichier f1
&>f2	redirige la sortie standard et la sortie d'erreur dans f2

Les re-directions



<	redirige l'entrée standard
>	redirige la sortie standard
>>	concatène la sortie standard
	redirige la sortie standard vers l'entrée d'une autre cmde
2>	redirige la sortie d'erreur
&>	redirige la sortie standard et la sortie d'erreur



Exemples :

```
ls > liste
```

```
ls .. >> liste
```

```
wc -l < liste
```

Effet

écrit (écrase) dans le fichier `liste`
la liste des fichiers et
répertoires du rép. courant

ajoute à la fin du fichier `liste`
le contenu du rép. parent

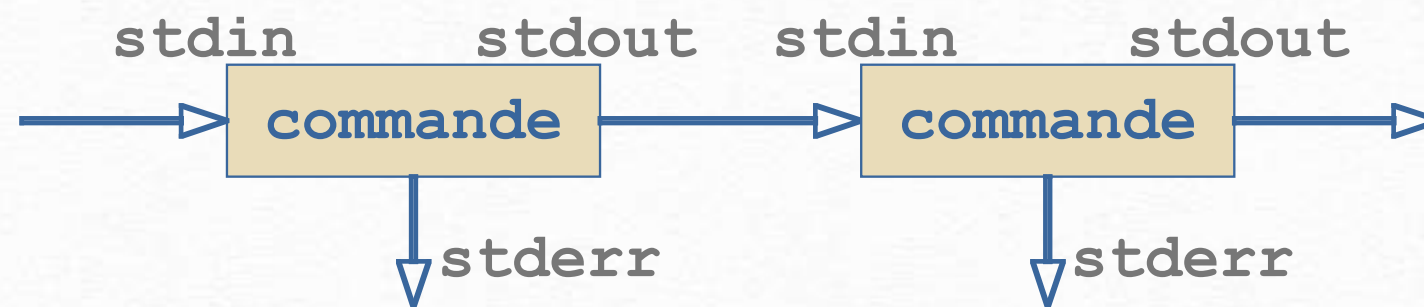
compte le nb de lignes du fich.
`liste`

Les re-directions



Le tube (*pipe*) permet de réutiliser le résultat d'une commande comme entrée d'une autre commande

	redirige la sortie standard vers l'entrée d'une autre cmde
--	--



QUIZZ

Exemples : «combien de fichiers dans le rép. courant ?»

sans pipe:

avec un pipe:



Regroupement de fichiers en
archives

Archivage et compression

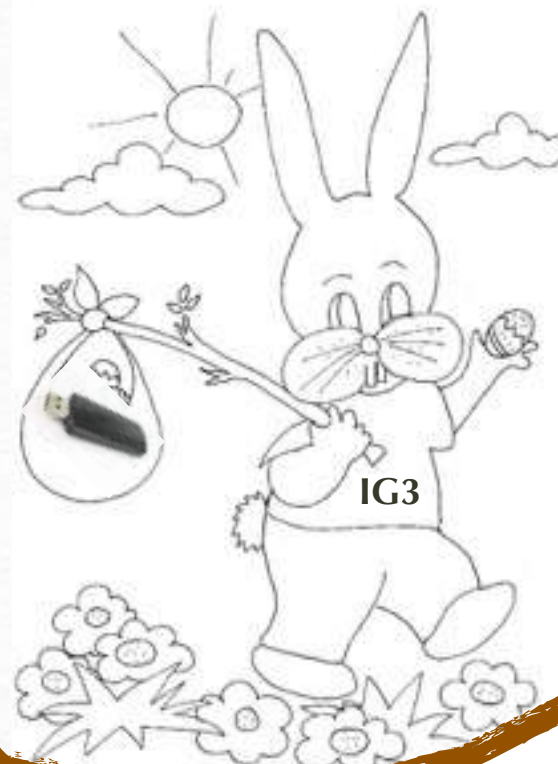
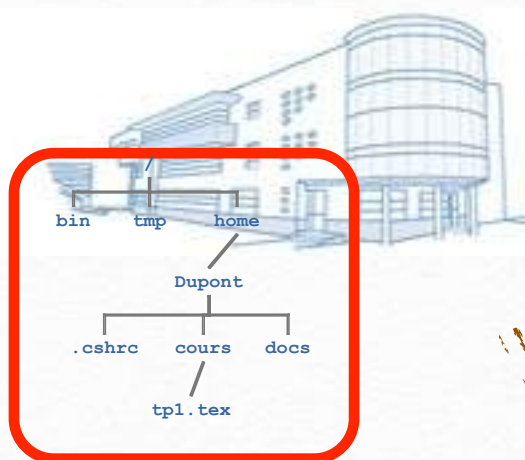
Définitions

- La **compression** consiste à regrouper un ensemble de fichiers et à réduire leur taille le plus possible afin de faciliter leur transfert.
- Le fichier contenant l'ensemble des autres fichiers compressés est généralement appelé fichier **archive**.
- Il existe plusieurs formats pour les archives :
 - sous **Windows** : **zip** et **rar** sont les plus répandus
 - sous **Unix** : **.tar.gz** et **tar.bz2**

Archivage et compression

Les raisons

- **Archivage** : simplifie le transfert (clef USB ou mél) : un seul fichier plutôt que plusieurs.
- **Compression** : réduit la quantité de données à transférer, à sauvegarder (on peut en mettre plus sur sa clé USB, ou en attache d'un courriel.)



Archivage et compression

Les outils

Il existe de nombreuses applications qui permettent d'archiver-et-compresser :

- Pour **Windows** : il s'agit principalement en partie d'outils réalisés en complément du SE :
 - *WinZip* ou *Winrar*
 - *PowerArchiver* (shareware)
 - *7-zip* (**freeware**)
 - Pour **Unix** : on peut distinguer **l'assemblage** d'un ensemble de fichiers/répertoires en un seul fichier (*archive*)
 - commande **tar**
- ... et la phase de **compression** de l'archive :
- commandes **zip**, **gzip**, **bzip2**, **compress**, ...
- Bien que **tar** puisse faire les deux en fait

Archivage et compression

La commande **tar** :

L'essentiel de la commande :

```
tar [c|t|x] [z] [v] [f device] [fichier(s) |  
répertoire(s)]
```

- c** (*create*) crée une nouvelle archive. (efface d'éventuels fichiers/réps présents dans l'archive auparavant)
- t** (*list*) affiche le contenu de l'archive
- x** (*extract*) extrait le(s) fichier(s) désigné(s) de l'archive et le(s) restaure en local sur le système. Désigner un répertoire par son nom aboutit à désarchiver son contenu récursivement.
- z** pour une archive compressée (par *gzip*)
- v** affiche d'informations lors de la manipulation de l'archive
- f *device*** force `tar` à utiliser l'argument suivant comme nom d'archive



les fichiers désarchivés appartiennent à celui qui les désarchive (peut importer l'endroit (répertoire) ou le créateur de l'archive)

Archivage et compression

La commande **gzip** (voir aussi **bzip2**) :

gzip [-d] [-l] [-r] *fichier(s)*

- d décompresse le fichier (mais ne désassemble pas dans le cas d'une archive *.tar*)
- l donne une liste d'information sur chaque fichier :
taille quand compressé et non compressé,
- r procède récursivement, i.e. traite (séparément) tous les fichiers trouvés dans la sous-arborescence

Exemples : «combien de fichiers dans le rép. courant ?»

créer un fichier compressé :

gzip *uneArchive.tar*

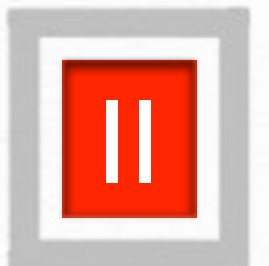
décompresser un fichier :

gzip -d *uneArchive.tar.gz*

Par la pratique



1. Créez un dossier `UNIX` contenant un dossier `COURS`
2. Créez deux fichiers vides (`touch`) `f1` (dans `UNIX`) et `f2` (dans `COURS`)
3. Vérifiez que tout est en place par `ls -R` depuis `~`
4. Créez une archive `unix.tar` contenant tout `UNIX` dans `~`
5. Détruisez le dossier `UNIX` et tout ce qu'il contient
6. Extrayez de l'archive uniquement le fichier `f1`
7. Où a-t-il été placé par défaut ?





Les filtres

Les filtres

Un **filtre** est une commande capable de traiter un flux de données et d'en effectuer un affichage formaté et/ou sélectif

cat	affiche le contenu des fichiers passés en paramètres (par défaut, stdin)
more	affiche <i>page par page</i> le contenu des fichiers passés en paramètres (par défaut, stdin)
grep	affiche les lignes du fichier passé en paramètre qui vérifient une expression régulière donnée
cut	sélectionne uniquement certaines colonnes du fichier passé en paramètre

on gagne à utiliser la cmde less plutôt que la cmde more
(moins user-friendly)

Les filtres

S

«afficher le contenu du fichier `MonAppli.cpp` en montrant aussi les caractères non visibles et les fins de lignes» :

```
cat -v MonAppli.cpp
```

«afficher le contenu du fichier `essai.txt` en groupant les lignes blanches (-s) » :

```
more -s essai.txt
```

- ▶ '=' pendant la visualisation permet de connaître le numéro des lignes affichées
- ▶ '/' pendant la visualisation permet de chercher les occurrences d'un mot dans le texte (mise en surbrillance)
- ▶ Contrairement à ce qu'indique son nom, la commande **less**, fait plus que la commande **more** (même syntaxe, retours arrières, meilleure gestion des gros fichiers).



Les filtres

Commandes de **sélection de parties** d'un fichier :

egrep *motif* [**fichier**]

recherche, dans le fichier passé en paramètre, les lignes vérifiant un motif donné.

Les motifs recherchés sont parfois composés de caractères non connus à l'avance.

Exemple (extrait de la commande `last`) : «quels sont les utilisateurs qui se sont connectés le 21 septembre avant 10h ?»

vberry	pts/2	tp5-pc01	Fri	Sep	22	15:15	still	logged	in
mcvill	pts/0	tp3-pc12	Fri	Sep	22	08:01	still	logged	in
fiorio	pts/0	dir-pc02	Thu	Sep	17	19:58 - 00:32	(04:33)		
pochard	pts/2	dir-pc01	Thu	Sep	11	09:21 - 10:21	(01:00)		
fiorio	pts/0	tp3-pc13	Thu	Sep	17	10:57 - 18:09	(07:12)		

CLASSIFIED



Une **expression régulière** permet de décrire un motif cherché de façon flexible en utilisant des méta-caractères :



egrep est une variante étendue de *grep*, fournissant une plus grande puissance d'expression pour définir les motifs cherchés

Les filtres

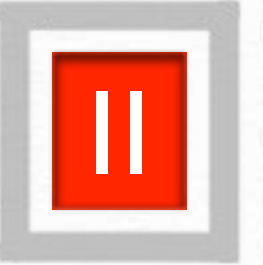
egrep

Commandes de **sélection de parties** d'un fichier.

Méta caractères utilisables dans une expression régulière :

- un caractère quelconque
- + une ou plusieurs occurrences
- * zéro, une ou plusieurs occurrences
- [<liste>] au choix parmi les possibilités indiquées
- [^<caractère>] tout sauf un certain caractère
- { x } exactement x fois le caractère précédent
- { x , y } entre x et y fois le caractère précédent
- ^ (en début d'expression) = début de la ligne
- \$ fin de la ligne
- \ pour *dé-spécialiser* un méta-caractère

Les filtres



Exemples de **sélection de parties** d'un fichier :

QUIZZ

- «quelles sont les lignes du fichier **documentation.txt** qui contiennent le mot **options** ?» :
 - «quelles sont les lignes du fichier **HOWTO** qui commencent par une minuscule ?»
- «combien de sous-répertoires du répertoire **Musiversal** contiennent un fichier **readme.txt** ?»

Les filtres



QUIZZ

Commandes de **sélection de parties** d'un fichier.

Exemple : «**quels sont les utilisateurs qui se sont connectés le 11 septembre avant 10h ?**»



CLASSIFIED

vberry	pts/2	tp5-pc01	Fri Sep 22 15:15	still logged in
mcvill	pts/0	tp3-pc12	Fri Sep 22 08:01	still logged in
fiorio	pts/0	dir-pc02	Thu Sep 17 19:58 - 00:32	(04:33)
pochard	pts/2	dir-pc01	Thu Sep 11 09:21 - 10:21	(01:00)
fiorio	pts/0	tp3-pc13	Thu Sep 17 10:57 - 18:09	(07:12)



Pour d'autres méta-caractères, voir le TP

Les filtres



Attention : à la différence entre `grep` et `grep -E` (`egrep`)

Dans les reg-exp basiques (`grep`), les meta-characters

'{', '}', '(', ')', '|', '+', '?' perdent leur sens particulier et sont considérés comme des caractères normaux. Pour faire appel à leur sens spécial, il faut les faire précéder d'un backslash (\)

Exemple : «**quelles sont les utilisateurs qui se sont connectés pour plus de 3 jours ?**»

vberry	ttys002	Thu Sep 14 10:51 - 10:52	(00:00)
cfiorio	ttys001	Tue Sep 12 16:38 - 17:27	(1+00:49)
cseguin	ttys000	Mon Sep 11 09:05 - 19:33	(3 +10:28)


```

      iii
iLE98Dj. :jD888888Dj:
.LGitiE888D.f8GjjjL8888E;
iE :8888Et. .G8888.
;i  E888, ,8888,
    D888, :8888:
    D888, :8888:
    D888, :8888:
    D888, :8888:
    888W, :8888:
    W88W, :8888:
    W88W, :8888:
    DGGD: :8888:
          :8888:
          :W888:
          :8888:
          E888i
          tW88D

```



L'éditeur nano

L'éditeur nano

un éditeur présent
sur la plupart des
serveurs

auxquels on
accède à
distance

nano [options] fichier.ext

L'essentiel :

- ⌘ CTRL + G : *get help*, liste des fonctions / raccourcis
- ⌘ CTRL + O : **enregistrer** le fichier ('Y' : sauver les changements)
- ⌘ CTRL + R : **charger** un fichier
- ⌘ CTRL + X : **quitter** Nano.



Explorateur de fichier rudimentaire mais utile

- ⌘ (disponible avoir effectué CTRL+O ou CTRL+R) : **CTRL-T**

L'éditeur nano

`nano` [options] fichier.ext

Sélection d'un morceau de texte :






- ⌘ Ctrl + ^ (accent citronfraise) : commencer à **sélectionner** un morceau de texte
- ⌘ Meta + ^ : **copier** le texte sélectionné *
- ⌘ Ctrl + K : **couper** le texte sélectionné (sinon la ligne en cours) --> dans le presse-papier) ;
- ⌘ Ctrl + U : **coller** la ligne de texte que vous venez de couper ;

* Dans nano sur Mac, la touche Meta est Esc

NB : pour se déplacer dans le texte, utiliser les touches flèches du clavier (pas la souris)

L'éditeur nano

Autres fonctions utiles

-  **Undo** : après avoir utilisé l'option **-u** au lancement :
M-U (undo) et M-E (redo) **!! expérimental et > v2.2.6 !!**
-  Ctrl + W : **rechercher** dans le fichier :
 -  sur la ligne du bas, indiquez le mot que vous cherchez
 -  la touche «entrée» sans indiquer de mot recherche l'**occurrence suivante** du mot que vous venez de chercher
 -  les flèches (haut et bas) vous permettent d'accéder aux **recherches précédentes**

L'éditeur nano

Pour la programmation

- ⌘ Ctrl + C : afficher la **position** (ligne / colonne) de votre curseur (utile pour le déboggage d'un fichier de code / configuration)
- ⌘ M + X : gagne de la place en masquant les lignes d'aides en bas de l'écran
- ⌘ option -i = **indentation automatique** : quand on crée une nouvelle ligne (contexte de programmation)
- ⌘ **Coloration syntaxique** : inclure les bons fichiers dans le fichier **.nanorc**