Formation “Unix et GNU/Linux”

Niveau intermédiaire

Partie 2/2

A.-S. Mouronval

Juillet 2011

Au sommaire…

Partie 2

- Système de fichiers (suite)
  - Commandes complémentaires
  - Espace disque
- Gestion des processus
  - Caractéristiques
  - Lancement
  - Contrôle
- Environnement de travail : le shell
  - Types de shells
  - Variables
  - Alias
  - Fichiers d'environnement

Commandes : notes sur les exemples

Nota : - dans la présentation des `commandes`, les '`[ ]`' indiquent des éléments facultatifs
- les exemples présentés respectent la forme suivante :

```
[formation@rose]$ cal -m 10 2009
    October 2009
Mo Tu We Th Fr Sa Su
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31


[formation@rose]$  Cal -m 10 2009
Cal: Command not found.


[formation@rose]$ man cal
CAL(1)                  BSD General Commands Manual                  CAL(1)
NAME
     cal - displays a calendar
SYNOPSIS
     cal [-smjy13] [[month] year]

(…)
```

En gras, la commande tapée par l'utilisateur

Résultat de la commande

En italique précédé de "#", commentaires relatifs à l'exemple

 # Erreur : Cal au lieu de cal

Prompt

indique que le résultat de la commande n'est pas complètement reporté

Compléments fichiers : `file`

- Connaître le type d'un fichier : `file [-L] filenames`
  - Type : fichier ASCII, exécutable, lien symbolique (*cf.* annexes), répertoire…
  - Utile par exemple pour les exécutables et les bibliothèques (pour quelle architecture ?)
  - Option `-L` : suit les liens

```
mouronv@service0:~> file license.dat TESTS
license.dat: ASCII text
TESTS/: directory

mouronv@service0:~> file commande
commande: Bourne-Again shell script text

mouronval@service0:~> file /opt/san/applications/abaqus/Commands/abaqus
/opt/san/applications/abaqus/Commands/abaqus: symbolic link to
`/opt/san/applications/abaqus/Commands/abq6102'

mouronval@service0:~> file -L /opt/san/applications/abaqus/Commands/abaqus
/opt/san/applications/abaqus/Commands/abaqus: ELF 64-bit LSB executable, x86-64,
version 1 (SYSV), for GNU/Linux 2.6.4, statically linked, not stripped
# ELF = Executable and Link Format

europa:~mouronval$ file sphere      # exécutable obtenu après compilation sur Mac OS X
sphere: Mach-O executable i386      # Mach-O = Mach Object (équivalent de ELF pour Mac OS X)
```

Compléments fichiers : `touch`

- Modification de la date de dernier accès d'un fichier : `touch`
  - Utile pour les Makefiles
  - Permet de créer rapidement un fichier vide

```
europa:~ mouronval$ ls -l essai.tex fichier_non_existant.txt
ls: fichier_non_existant.txt: No such file or directory
-rw-r--r--   1 mouronva  anne-sop  420 May 29  2008 essai.tex

europa:~ mouronval$ touch essai.tex fichier_non_existant.txt

europa:~ mouronval$ !l
ls -l essai.tex fichier_non_existant.txt
-rw-r--r--   1 mouronva  anne-sop  420 Sep 23 17:09 essai.tex
-rw-r--r--   1 mouronva  anne-sop    0 Sep 23 17:09 fichier_non_existant.txt
```

Trouver un fichier exécutable : `which`

- Localiser un fichier exécutable : `which [-a] programnames`
  - Recherche au sein des répertoires renseignés par la variable du shell « PATH » (affichable par la commande `echo $PATH`)
  - Retourne le chemin complet si le fichier est trouvé et "`Command not found`" (ou rien, suivant le shell) dans le cas contraire
  - Option `-a` : *All*, affiche toutes les occurrences (par défaut, si plusieurs programmes partagent le même nom, `which` n'indique que le 1er selon l'ordre du PATH)

```
europa:~ mouronva% which who ensight8 matlab                    # recherche de 3 exécutables
/usr/bin/who
/Applications/CEI/bin/ensight8
matlab: Command not found.

europa:~ mouronva% file /Applications/MATLAB_R2007b/bin/matlab
/Applications/MATLAB_R2007b/bin/matlab: Bourne shell script text executable
# matlab est bien un exécutable mais…

europa:~ mouronva% echo $PATH
/Applications/CEI/bin:/sw/bin:/sw/sbin:/bin:/sbin:(…)
# … Applications/MATLAB_R2007b/bin  ne fait pas partie du PATH
```

- Commande similaire (binaires, sources et manuel d'un programme) : `whereis`

Trouver un fichier : `find`

Trouver des fichiers ou répertoires à partir de critères : `find dir expr command`

- Recherche récursive dans une arborescence dont la racine est le répertoire `dir`
- `expr` désigne des critères de recherche (combinables) dont :
  - `-name` et `-iname` : nom (sensible ou non à la casse MAJ/min)
  - `-user` et `-group`: utilisateur et groupe
  - `-atime` et `-mtime` : date dernier accès/modification (utile pour nettoyage)
  - `-size` : taille (utile pour nettoyage)
  - `-type` : type de fichier (`D` : directory)
  - `!` : inverse la recherche
- Niveau de recherche dans l'arborescence : critère `-maxdepth`
- `command` :
  - `-print` : affiche le résultat à l'écran (important car certaines implémentations n'affichent pas par défaut le résultat de la recherche !)
  - `-exec cmd {} \;` : applique `cmd` sur les fichiers trouvés (indiqués par `{}`)

Trouver un fichier : `find` (exemples)

Exemples d'utilisation

1/ Rechercher les fichiers ppt dont le nom contient UNIX (en min ou MAJ) à partir du répertoire courant (`.`)

```
europa:~ mouronval$ find . -iname "*UNIX*.ppt" -print
./Formation_Unix/Unix-1/formation_unix-1.ppt
./Formation_Unix/Unix-2/formation_unix-2.ppt
./Formation_Unix/Unix-intro/formation_unix.ppt
./Formation_Unix/Unix-Linux.ppt
```

2/ Idem que 1/ mais en limitant la recherche aux répertoires et aux sous-répertoires

```
europa:~ mouronval$ find . -maxdepth 2 -iname "*UNIX*.ppt" -print
./Formation_Unix/Unix-Linux.ppt
```

3/ Rechercher tous les fichiers "core" à partir du répertoire `CODES_FORTRAN`

```
europa:~ mouronval$ find CODES_FORTRAN/ -name "core" -print
CODES_FORTRAN/RUN1/core
CODES_FORTRAN/RUN2/core
```

4/ Idem que 3/ mais les effacer au lieu d'afficher le résultat

```
europa:~ mouronval$ find CODES_FORTRAN/ -name "core" -exec rm -f {} \;
```

Espace disque : `df` et `quota`

- Occupation des FS montés (*Display Free disk*) : `df -h [filesystems]`
  Option `-h` : *Human readable format,* facilite la lecture de la sortie

```
[formation@rose]$ df -h
Filesystem     Size  Used Avail Use% Mounted on
/dev/sda3       15G  6.2G  7.6G  45% /
/dev/sda6       68G  203M   65G   1% /tmp2              # répertoires locaux de calcul
/dev/sda2       63G   55M   60G   1% /tmp1              # répertoires locaux de calcul
/dev/sda1       99M   18M   77M  19% /boot
tmpfs         1013M  192K 1013M   1% /dev/shm
thuya:/MSS    1008G  449G  509G  47% /mnt/thuya/MSS   # contient les répertoires utilisateurs (NFS)
```

- Afficher son quota : `quota`
  - 2 quotas : disque (blocs de 1 Ko) et nb de fichiers (inodes, 0 = pas de quota)
  - Limites imposées par l'administrateur de la machine :
    Limites "*soft*" (colonnes "`quota`") dépassables pendant une certaine durée (sursis : colonne "`grace`")
    Limites "*hard*" (colonnes "`limit`") non-dépassables
  - A vérifier en cas de problème d'écriture ou de sauvegarde !

```
[formation@rose]$ quota
   Disk quotas for user formation (uid 688):
    Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
   thuya:/MSS     3028  4500000 5000000             310       0       0
```

Espace disque : `du`

Espace disque occupé par un répertoire (*Disk Usage*) : `du [-skh] [dirnames]`

- Par défaut, espace généralement donné en nombre de blocs occupés sur le disque
  Taille des blocs variable suivant commandes, FS… (512 ou 1024 octets…) : `man du`
  ⇒ Options conseillées : `-k`, *Kilobytes* ou `-h`, *Human readable format*

- Option `-s` : *Summarize*, affiche uniquement l'espace disque occupé par les répertoires donnés en argument

- Option `-a` : *All*, affiche l'espace occupé pour tous les fichiers contenus

- Souvent utilisée avec la commande `sort -nr` (affichage par taille décroissante)

- Utilisable aussi pour les fichiers (équivalent de `ls -s filename`)
  mais préférez `ls -l filename` (taille réelle, *cf.* annexes)

Espace disque : du (exemples)

```
europa:~ mouronval$ du GROUPE_TD1          #espace disque en nb de blocs
104     GROUPE_TD1/C_exemples              # (Mac OS X 10.4 man du : 1 bloc = 512 octets)
72      GROUPE_TD1/Fortran_exemples
176     GROUPE_TD1

europa:~ mouronval$ du -h GROUPE_TD1 | sort -nr # idem au format "human readable" avec tri
 88K    GROUPE_TD1
 52K    GROUPE_TD1/C_exemples
 36K    GROUPE_TD1/Fortran_exemples

europa:~ mouronval$ du -sh GROUPE_TD1                    # résumé au format "human readable"
88K    GROUPE_TD1

europa:~ mouronval$ du -ah GROUPE_TD1 # espace occupé par fichier au format « human readable »
4.0K    GROUPE_TD1/C_exemples/boucle_do_while.c
8.0K    GROUPE_TD1/C_exemples/boucle_for
(...)
 52K    GROUPE_TD1/C_exemples
 12K    GROUPE_TD1/Fortran_exemples/concat
(...)
 36K    GROUPE_TD1/Fortran_exemples
 88K    GROUPE_TD1

# affichage des 2 plus gros fichiers de C_exemples
europa:~ mouronval$ du -ah GROUPE_TD1/C_exemples | sort -nr | head -3
52K     GROUPE_TD1/C_exemples/
8.0K    GROUPE_TD1/C_exemples/surface_cercle
8.0K    GROUPE_TD1/C_exemples/boucle_for
```

Gestion des processus : définitions

- Processus (*process*) : **instance d'un programme** en train de s'exécuter (tâche) et les ressources qui lui sont rattachées (mémoire, fichiers, données…)

  Exemple : un shell en fonctionnement

- 2 types de processus
  - processus système
  - processus utilisateur

- Processus actifs en permanence s'exécutant en arrière-plan: **démons** (*daemons*)

  Exemple (services) : `crond`, `ftpd`, `httpd`…

Gestion des processus : caractéristiques

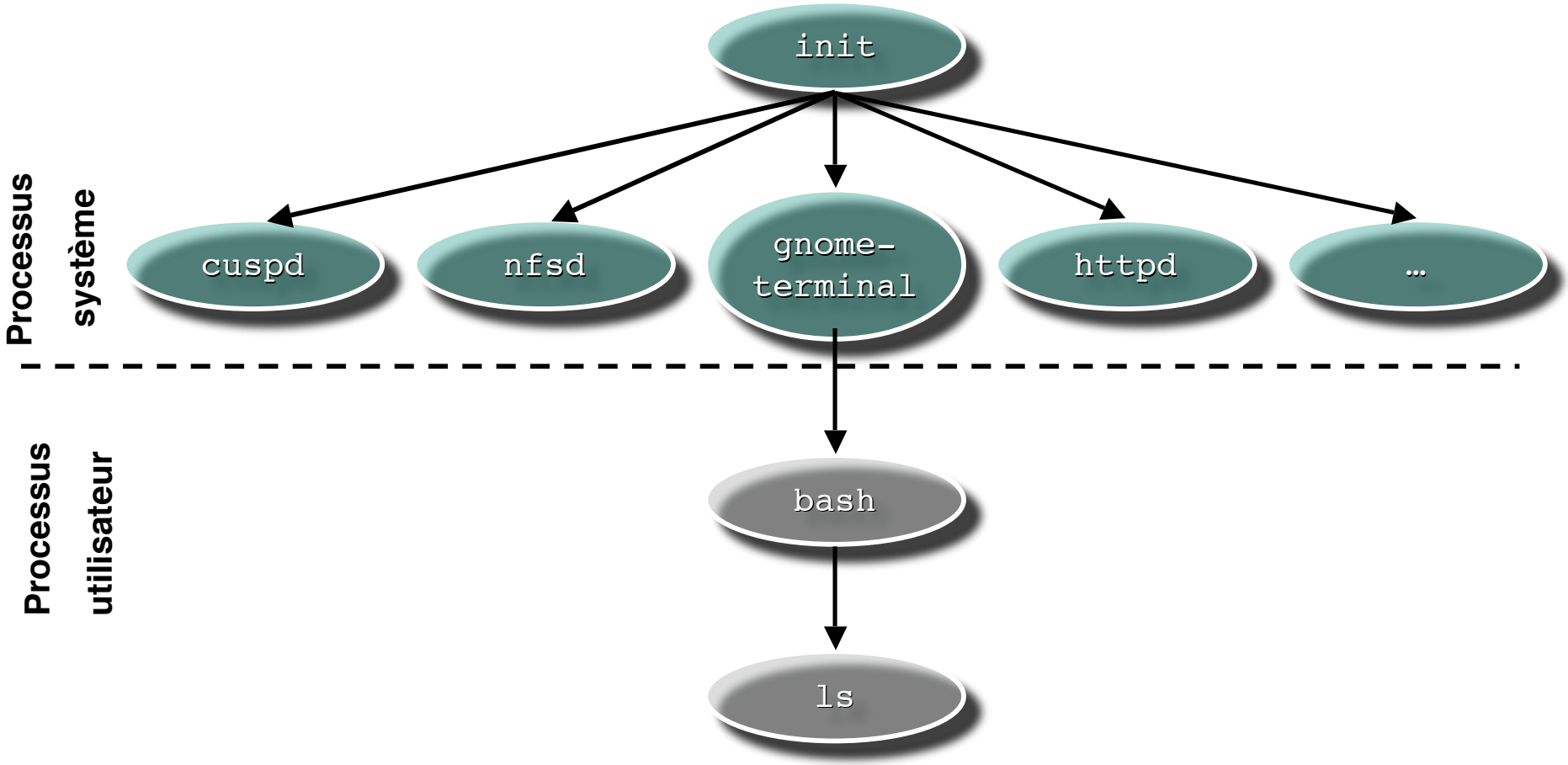
Chaque processus possède

- Un **PID** (*Process IDentification*) : entier, identifiant unique
- Un PPID (*Parent Process IDentification*) : PID de son processus parent

⇒ **hiérarchie de processus** parents (pères) et enfants (fils) à partir de `init` (PID 1)

- Un propriétaire et un groupe identifiés par leur UID et GID
- Un terminal de contrôle TTY (indéfini pour les démons)
- Une valeur de priorité (NICE VALUE)

Gestion des processus : exemple simplifié de hiérarchie



`pstree` permet de visualiser cette arborescence (pas installée par défaut sur tous les O.S.)

Gestion des processus : état d'un processus

Différents états possibles entre la création et la fin d'un processus :

- En cours d'exécution (*Running*) ou prêt à l'exécution (*Runnable*)
  Processus actif disposant du CPU ou en attente qu'un processus actif laisse sa place

- Bloqué (*Blocked*), en attente (*Waiting*), endormi (*Sleeping*), suspendu (*sTopped*)…
  Processus ne disposant pas du CPU : en attente d'une ressource, entrée clavier *etc*

- Zombie (*Zombie*)
  Processus terminé dont le père n'a pas encore connaissance de la mort

Gestion des processus : lancement

- Lancement interactif au prompt : `command [arguments]`

- Ordre de recherche des commandes

  1/ alias

  2/ commande interne (`cd`… , *cf.* annexes)

  3/ commande dans un répertoire de la variable PATH

```
europa:~ mouronval$ alias                    # affichage des alias
                                             # aucun alias ici

europa:~ mouronval$ echo $PATH
/sw/bin:/sw/sbin:/Applications/CEI/bin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin

europa:~ mouronval$ gcc -o hello hello.c

europa:~ mouronval$ hello
-bash: hello: command not found

europa:~ mouronval$ ./hello                  # chemin relatif : . fait référence au répertoire courant
hello, world !

europa:~ mouronval$ /Users/anne-sophiemouronval/hello        # chemin absolu
hello, world !
```

Gestion des processus : lancement (suite)

- En cas de '`Permission denied`' au lancement (fichiers script…) :

  1/ Vérifier les droits : `ls -l filename`

  2/ Accorder le droit d'exécution si besoin : `chmod u+x filename`

Remarque : droits accordés automatiquement aux exécutables créés par compilation

```
europa:~ mouronval$ cat script_exemple
#!/bin/bash
echo "Bonjour"

europa:~ mouronval$ ./script_exemple
bash: ./script_exemple: Permission denied

europa:~ mouronval$ ls -l script_exemple
-rw-r--r--   1 mouronva  anne-sop  27 Jun 12 23:12 script_exemple

europa:~ mouronval$ chmod u+x script_exemple

europa:~ mouronval$ ./script_exemple
Bonjour
```

Gestion des processus : interruption (cas simple)

- Interruption du calcul, envoi d'un signal via le clavier : `CTRL c`

Exemple de calcul sans interruption

```
europa:~ mouronval$ ./sphere
1
 Volume de la sphere =    4.1887903
europa:~ mouronval$
```

Exemple de calcul avec interruption

```
europa:~ mouronval$ ./sphere
2
^C                              # interruption par CTRL c
europa:~ mouronval$
```

Gestion des processus : liste des processus, `jobs`

Connaître la liste et l'état des processus attachés au terminal : `jobs`

- « + » et « – » : dernier et avant-dernier processus manipulé
- Numéros de jobs indiqués uniques pour chaque terminal
- Option `-l` : *long*, indique en plus les PIDs
- Utilisé surtout lors des passages foreground / background

```
europa:~ mouronval$ jobs -l
[1]-   640 Running   ../../exe/macx32/release_dynamic/oofe G2Hr.oofe > sortie.txt &
[2]+   641 Running   ../../exe/macx32/release_dynamic/oofe G2Hr2.oofe > sortie2.txt &
```

Gestion des processus : liste des processus, `ps`

Lister les processus *Process Status* : `ps`

- Attention peu standardisé (BSD vs System V) : `man ps` !!
- Principales options BSD (Mac OS X, FreeBSD…) : `-aux` ou `aux`
  - par défaut : processus de l'utilisateur lancés depuis un terminal
  - `-a` : *All*, affiche en plus les processus des autres utilisateurs
  - `-u` : affiche les champs suivants
    `pid`
    `%cpu`
    `%mem`, `vsz` et `rss` (infos mémoire)
    `tt` (terminal de rattachement)
    `stat` (état)
    `started` (heure de début) et `time` (temps d'exécution)
    `command`
  - `-x` : affiche les processus non-attachés à un terminal
- Options équivalentes Sys V : `-edf`
- Souvent utilisé avec `grep` pour filtrer les informations

Gestion des processus : liste des processus, `ps` (exemple)

Exemple sur une machine sous Mac OS X (implémentation de `ps` de type BSD)

```
europa:~ mouronval$ man ps                    # verification des options
PS(1)                    BSD General Commands Manual                    PS(1)
(…)

europa:~ mouronval$ ps                        # processus de mouronval  rattachés à un terminal
  PID  TT  STAT      TIME COMMAND
  671  p1  S      0:00.03 -bash
  695  p1  R      1:28.95 code_c
  699  p2  S+     0:00.02 -bash


europa:~ mouronval$ ps -a                     # tous les processus rattachés à un terminal
  PID  TT  STAT      TIME COMMAND
  670  p1  Ss     0:00.01 login -pf mouronval
  671  p1  S      0:00.03 -bash
  695  p1  R      1:33.22 code_c
  736  p1  R+     0:00.00 ps -a
  698  p2  Ss     0:00.01 login -pf mouronval
  699  p2  S+     0:00.02 -bash
```

Gestion des processus : liste des processus, `ps` (exemple)

```
europa:~ mouronval$  ps -au        # tous les processus rattachés à un terminal avec plus d'infos
USER       PID %CPU %MEM      VSZ    RSS  TT  STAT STARTED      TIME COMMAND
mouronva   695  25.5 -19.0   431212 398860  p1  S     2:57PM  code_c
root       774   1.5 -0.0    27312    432  p1  R+    3:05PM   0:00.01 ps -au
root       670   0.0 -0.0    27576    524  p1  Ss    2:56PM   0:00.01 login -pf
    mouronval
mouronva   671   0.0 -0.0    27728    748  p1  S     2:56PM   0:00.06 -bash
root       698   0.0 -0.0    27576    532  p2  Ss    2:57PM   0:00.01 login -pf
    mouronval
mouronva   699   0.0 -0.0    27728    744  p2  S+    2:57PM   0:00.02 -bash


europa:~ mouronval$  ps -aux       # tous les processus rattachés ou non (tty=??)
                                   # à un terminal  avec plus d'infos
USER       PID %CPU %MEM      VSZ    RSS  TT  STAT STARTED      TIME COMMAND
mouronva   695  100.0 -18.8   427164 395296  p1  R     2:57PM   8:47.47 code_c
root       670   0.0 -0.0    27576    524  p1  Ss    2:56PM   0:00.01 login -pf
    mouronval
mouronva   671   0.0 -0.0    27728    740  p1  S     2:56PM   0:00.06 -bash
root       698   0.0 -0.0    27576    532  p2  Ss    2:57PM   0:00.01 login -pf
    mouronval
mouronva   699   0.0 -0.0    27728    744  p2  S+    2:57PM   0:00.02 -bash
root       776   0.0 -0.0    27312    432  p1  R+    3:06PM   0:00.00 ps -aux
root         1   0.0 -0.0    28356    408  ??  S<s   9:41AM   0:00.17 /sbin/launchd
root        23   0.0 -0.0    27268    164  ??  Ss    9:41AM   0:00.04
    /sbin/dynamic_pager -F /private/var/vm/swapfile
root        27   0.0 -0.0    28296    144  ??  Ss    9:41AM   0:01.15 kextd
(...)
```

Gestion des processus : liste des processus, `top`

Afficher **en continu** la liste des processus : `top`

- Attention peu standardisé : `man top` !!

- Diverses informations globales (variables suivant implémentation)
  - Nombres de processus sur la machine
  - Etat des cpus : % user, sys, idle…
  - Mémoire vive utilisée, libre…
  - …

- Informations diverses pour chaque processus (variables suivant implémentation)
  Exemples : `PID, USER, %CPU, STAT, COMMAND`

- Informations concernant l'utilisation mémoire délicates à interpréter
  Dénomination et calcul différents suivant les implémentations :
  Ram ? Ram + swap ? Ram utilisée, réservée… ?
  Prise en compte des bibliothèques partagées ?
  http://www.jacquet80.eu/blog/post/2010/03/Memoire-processus-Linux
  http://loligrub.be/contrib/tlepoint/BASE/node104.html
  http://www.mac4ever.com/articles/macos/263/la_gestion_de_la_memoire_vive_sous_mac_os_x/

Gestion des processus : liste des processus, `top` (suite)

- Commandes en mode interactif pour `top`
  - `h` : affiche l'aide
  - `q` : quitte `top`
  - `M` : trie les processus par utilisation décroissante de la mémoire
  - `P` : trie les processus par utilisation décroissante du cpu
    (comportement par défaut pour la majorité des implémentations)
  - `H` : affiche tous les threads (utile pour les codes utilisant OpenMP)
  - `u` : trie les processus par utilisateur
  - `s` : change la durée de rafraîchissement
  - `k` : envoie un signal à un processus

Nota : sur certaines implémentations, ces commandes diffèrent

Exemple Mac OS X : passer en “compatibility mode” en tapant `x` puis `?` pour afficher la liste

Gestion des processus : liste des processus (top)

Exemple sur un des noeuds de 16 processeurs et 48 Gos de Ram de l'ancien calculateur SGI (tourab)

```
[mouronval@wayra mouronval]$ top
16:14:45  up 355 days,  5:30,  5 users,  load average: 5.77, 5.77, 5.85
188 processes: 184 sleeping, 3 running, 1 zombie, 0 stopped
CPU states:  cpu    user    nice  system    irq  softirq  iowait    idle
           total  728.0%    0.0%    1.6%   0.0%     0.0%    0.0%  867.2%
           cpu00    0.1%    0.0%    0.3%   0.0%     0.0%    0.0%   99.4%
           cpu01   96.3%    0.0%    0.1%   0.0%     0.0%    0.0%    3.4%
           cpu02   59.4%    0.0%    0.3%   0.0%     0.0%    0.0%   40.1%
           cpu03   59.5%    0.0%    0.2%   0.0%     0.0%    0.0%   40.1%
          (…)
           cpu15   38.0%    0.0%    0.0%   0.0%     0.0%    0.0%   61.8%
Mem:  48012288k av, 33634496k used, 14377792k free,       0k shrd,     128k buff
      12818672k active,            12613232k inactive
Swap: 62974880k av,   58272k used, 62916608k free                 10495840k cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
24139 dupont     25   0 3000M 2.9G 2991M R    99.9  6.2   5:05   7 cmp_trs
23455 dupont     25   0 3010M 2.9G 2997M R    96.4  6.2 392:45   1 cmp_trs
24224 mouronva   15   0 38384 2688 38160 R     0.3  0.0   0:00   0 top
14445 dupont     15   0 59120  18M 58672 S     0.1  0.0   2:30   4 emacs
    1 root       15   0  3008 1152   176 S     0.0  0.0  51:50  11 init
    (…)
```

Total cpu (user) = 728% alors que 2 cpus semblent utilisés seulement.
Taper `H` en mode interactif ⇒ `cmp_trs` utilise 12 cpus et non 2 (code OMP lancé 2 fois avec 6 threads). Attention à la visibilité des threads avec `top` !

Gestion des processus : liste des processus (top)

Exemple sur une machine de 2 coeurs et 2 Gos de Ram sous Mac OS X

Présentation différente de l'exemple précédent !

```
europa:~ mouronval$ top
Processes:  57 total, 3 running, 54 sleeping... 182 threads        13:59:00
Load Avg:  0.64, 0.35, 0.22     CPU usage:  50.9% user, 2.4% sys, 46.7% idle
SharedLibs: num =  155, resident = 35.4M code, 5.27M data, 8.27M LinkEdit
MemRegions: num =  6647, resident =  605M + 13.6M private,  221M shared
PhysMem:   134M wired,  230M active,  869M inactive, 1.21G used,  813M free
VM: 8.08G +  108M   34154(0) pageins, 0(0) pageouts

  PID COMMAND      %CPU   TIME    #TH #PRTS #MREGS RPRVT  RSHRD  RSIZE  VSIZE
  565 sphere       0.0%  0:00.00   1    13    26   148K  1.08M   528K  31.8M
  563 top          5.5%  0:02.17   1    19    20   896K   840K  1.36M  27.0M
  562 oofe        99.9%  0:36.49   1    13   699   366M  6.00M   370M   398M
  528 bash         0.0%  0:00.03   1    14    16   228K  1.21M   852K  27.1M
  527 login        0.0%  0:00.00   1    16    40   172K   976K   656K  26.9M
  495 bash         0.0%  0:00.03   1    14    16   220K  1.21M   856K  27.1M
  494 login        0.0%  0:00.00   1    16    40   172K   976K   648K  26.9M
  492 Terminal     0.4%  0:01.81   7   137   184  3.27M  11.9M  10.5M   366M
(...)
```

Gestion des processus : &, `fg` et `bg`

- Tâches interactives exécutées par défaut en “avant plan” (fg, *foreground*)
  Blocage de la ligne de commande

⇒ attendre la fin de la commande, l’arrêter (`CTRL c`), ouvrir un autre terminal ou basculer en arrière plan (*cf.* `bg`)

```
europa:~ mouronval$ ./sphere
1
 Volume de la sphere =    4.1887903
europa:~ mouronval$                          # on récupère le prompt à la fin du calcul
```

- Lancement en “arrière plan” ou “tâche de fond” (bg, *background*) : `command &`
  Permet de « récupérer la main »

  Nota : le « signal clavier » `CTRL c` ne permet plus son arrêt (utiliser d’autres signaux)

```
europa:~ mouronval$ ./sphere < entree.txt > resultat.txt &
[1] 556                        #  numéro du job et PID
europa:~ mouronval$ ls         # on récupère le prompt avant la fin du calcul (suivi du calcul via ps)
europa:~ mouronval$ cd Formation
europa:~ mouronval$
[1]+  Done   ./sphere <entree.txt > resultat.txt   (wd: ~ mouronval
(wd now: ~Formation)                                    # annonce de la fin du calcul
```

Gestion des processus : &, `fg` et `bg`

- Conseils pour bien utiliser le mode “arrière plan” :
  - l’entrée standard (clavier) n’est plus disponible

```
europa:~ mouronval$ ./sphere &  # lancement en tache de fond
[1] 542
europa:~ mouronval$ 1       # impossible de faire l’entrée clavier (valeur du rayon) attendue par le code
-bash: 1: command not found                                  # … le code va rester en attente (Sleeping)
europa:~ mouronval$
```

⇒ utiliser une redirection (lecture à partir d’un fichier), voir page précédente

  - la sortie standard (écran) existe toujours mais peut perturber le travail de l’utilisateur sur la ligne de commande

```
europa:~ mouronval$ ./sphere < entree.txt &  # lancement en tache de fond avec redirection entrée
[1] 549
europa:~/Formation_Unix/Tests_commandes mouronval$ ls
carnet_adresse.txt   entree.txt      lib_oofe.tar.gz          sphere
europa:~ mouronval$ Volume de la sphere =    4.1887903        # sortie du code affichée
                                                              # à n’importe quel moment
[1]+  Done                    ./sphere <entree.txt europa:~ mouronval$
```

⇒ utiliser également une redirection vers un fichier, voir page précédente

Gestion des processus : &, `fg` et `bg` (suite)

- Basculer une tâche lancée en foreground en background : `CTRL z` puis `bg [NUM]`
  - Phase 1 : `CTRL z` suspend la tâche, elle n'utilise plus le CPU
  - Phase 2 : `bg [NUM]` relance le processus de numéro `NUM` en arrière plan
    Par défaut `bg` s'applique au processus courant (noté + par `jobs`)

- Basculer une tâche de background à foreground : `fg [NUM]`

```
europa:~ mouronval$ ./sphere < entree.txt > resultat.txt      # lancement en foreground
^Z                                                            # suspendre la tâche
[1]+  Stopped                 ./sphere < entree.txt > resultat.txt
europa:~ mouronval$ jobs -l                                   # vérification
[1]+   559 Suspended ./sphere < entree.txt > resultat.txt    #  la tâche est suspendue
europa:~ mouronval$ bg                                  # passage en background du processus courant
[1]+ ./sphere < entree.txt > resultat.txt &

europa:~ mouronval$ jobs -l                             # vérification : tâche running en background
[1]+   559 Running ./sphere < entree.txt > resultat.txt &

europa:~ mouronval$ fg                                  # retour en foreground du processus courant
./sphere < entree.txt > resultat.txt
^C                                                      # arrêt de la tâche par CTRL c
europa:~ mouronval$
```

Des signaux permettent de contrôler le déroulement des processus

- Un signal peut être désigné
  - par son nom (ex : SIGKILL)
  - par son numéro (ex : 9), attention celui-ci dépend de l'O.S.

  $\Rightarrow$ privilégier l'utilisation du nom (surtout pour SIGSTOP et SIGCONT)
- Certains signaux peuvent être ignorés (ex : SIGTERM) d'autres non (ex : SIGKILL)
- Un signal peut correspondre à une combinaison de touches au clavier (`CTRL c`, `CTRL z`)
- Le comportement d'un processus à la réception de certains signaux est modifiable

Principaux signaux (valeurs numériques : MAC OS X et GNU/Linux Fedora)

Signal	Valeur numérique	Description
SIGINT	2	Termine le processus. Interruption depuis le clavier par `CTRL c` (modifiable)
SIGKILL	9	Termine le processus. Signal « KILL ». Arrêt brutal. Signal ne pouvant être ignoré
SIGTERM	15	Termine le processus. Signal de fin (arrêt propre)
SIGSTOP	17 / 19	Suspension du processus. Signal ne pouvant être ignoré
SIGTSTP	18 / 20	Suspension invoquée depuis le clavier par `CTRL z` (modifiable)
SIGCONT	19 / 18	Continue le processus si suspendu. Signal ne pouvant être ignoré
SIGSEGV	11	Termine le processus. Référence mémoire invalide (SEGmentation Violation)

Gestion des processus : envoyer un signal

Envoyer un signal à un processus :

`kill [-num_signal] PIDs` ou `kill [-name_signal] PIDs`

- `PIDS` : PIDs des processus concernés
- `num_signal` : numéro du signal à envoyer, par défaut numéro de SIGTERM
- `name_signal` : nom du signal à envoyer, par défaut SIGTERM
  sous la forme SIGXXXX ou XXXX selon les O.S.
- Signaux les plus utilisés : SIGTERM (15), SIGKILL (9), SIGSTOP et SIGCONT

  Privilégier SIGTERM (arrêt propre en général) par rapport à SIGKILL (arrêt brutal)
- Nécessite d'être propriétaire du processus

- Commande similaire utilisant un nom de processus au lieu d'un PID : `killall`

Gestion des processus : envoyer un signal (exemple)

Exemple sur une machine sous Mac OS X

```
europa:~/OOFE/TESTS/ mouronval$ ../../exe/oofe G2Hr.oofe  > sortie.txt &
[1] 729
europa:~/OOFE/TESTS/ mouronval$ ps
  PID  TT  STAT      TIME COMMAND
  593  p1  S      0:00.20 -bash
  729  p1  R      0:01.58 ../../exe/oofe G2Hr.oofe

europa:~/OOFE/TESTS/ mouronval$ kill -STOP 729
[1]+  Stopped                 ../../exe/oofe G2Hr.oofe >sortie.txt

europa:~/OOFE/TESTS/ mouronval$ kill -CONT 729

europa:~/OOFE/TESTS/ mouronval$ ps
  PID  TT  STAT      TIME COMMAND
  593  p1  S      0:00.20 -bash
  729  p1  R      0:11.04 ../../exe/oofe G2Hr.oofe

europa:~/OOFE/TESTS/ mouronval$ kill -9 729
[1]+  Killed                  ../../exe/oofe G2Hr.oofe >sortie.txt
```

Gestion des processus : récapitulatif arrêt

Marche à suivre pour terminer un processus (récapitulatif)

- Condition nécessaire : être propriétaire du processus
- Si le processus est running en avant plan dans un terminal : `CTRL c`
- Si échec ou si le processus n'est pas attaché au terminal

  1/ Emuler un autre terminal sous le même login, si besoin

  2/ Utiliser la commande `ps` ou `top` pour visualiser le PID du processus concerné

  3/ Terminer le processus `kill PID` ou `kill -9 PID` (si échec)

Gestion des processus : commandes complémentaires

- Attention : en général, la mort d'un processus père entraîne la mort de ses processus fils (`SIGHUP`)
  Lancer un processus fils en tâche de fond (`&`) n'empêche pas cela en général

  ⇒ Solution : commande `nohup`

  Exemple d'utilisation : cas d'une connexion à une machine distante par ssh, lancement d'un calcul et déconnexion

```
europa:~ mouronval$ ssh -X formation@rose
(…)
formation@rose%  nohup mon_prog  < entree.txt > sortie.txt &
formation@rose%  exit
```

- Gérer les priorités des processus : `nice` et `renice`

- Lancer des commandes en différé : `at`

- Planifier des tâches régulières : `crontab`

Gestion des processus : cas des calculateurs

Cas des calculateurs (IDRIS, CINES, Mesocentre ECP…)

- Frontales (login, compilation…) et nœuds de calcul
- Travaux exécutés « en batch » via un **gestionnaire de travaux**
  - Soumission du job depuis la frontale (réservation de ressources…)
  - Calculs lancés sur nœuds de calcul inaccessibles à l'utilisateur
    ⇒ contrôle des processus via le gestionnaire (et non par `top`, `ps`, `kill`…)
- Gestionnaires de travaux
  - PBS Pro (sur Igloo)
  - LoadLeveler
  - SGE (*Sun Grid Engine*)
  - …

Rappel sur les shells : 2 grandes familles

- Shells du type Bourne Shell
  - sh ou bsh (*Bourne SHell*, Steve Bourne)
    le plus ancien (AT&T), simple mais sans certaines fonctionnalités (alias, historique…)
  - **ksh** (*Korn Shell*, David Korn)
    extension du Bourne Shell
  - **bash** (*Bourne Again SHell*, Brian Fox & Chat Ramey)
    le plus utilisé sur GNU/Linux (GNU bash)
  - …
- Shells du type C-Shell
  - csh (*C-SHell*, Bill Joy)
    shell BSD, syntaxe proche de celle du langage C
  - **tcsh** (*TENEX C-SHell*)
    version « moderne » du csh, shell par défaut pour les machines sous FreeBSD

Identifier les shells disponibles : `cat /etc/shell`

- Remarque : sh (shell POSIX) est souvent bash et
  csh est souvent tcsh (ou des liens vers ces derniers)

⇒ Utiliser `nom_de_shell --version` pour vérifier et `ls -l` pour voir les liens

Exemple sur une machine sous Mac OS X
sh est ici bash

```
europa:~ mouronval$ cat /etc/shells
/bin/bash
/bin/csh
/bin/ksh
/bin/sh
/bin/tcsh
/bin/zsh
europa:~ mouronval$ sh --version
GNU bash, version 2.05b.0(1)-release (powerpc-apple-darwin8.0)
```

L'environnement de travail : identifier les shells disponibles (suite)

Exemple sur une machine sous GNU/Linux Fedora

Ici pas de ksh installé par défaut, sh et csh sont des liens

```
formation@orchis% cat /etc/shells
/bin/sh
/bin/bash
/bin/zsh
/bin/tcsh
/bin/csh
formation@orchis% ls -l /bin/*sh
-rwxr-xr-x 1 root root 755688 2008-10-23 16:30 /bin/bash
lrwxrwxrwx 1 root root      4 2009-03-24 17:16 /bin/csh -> tcsh
lrwxrwxrwx 1 root root      4 2009-03-24 17:10 /bin/sh -> bash
(...)
```

L'environnement de travail : changer de shell (exemples)

Shell de connexion (*login shell*)

- En mode texte, lancé après identification (login, mot de passe)
  En mode graphique, lancé entre la bannière de connexion et l'affichage du bureau

- Choix du shell de connexion de l'utilisateur fait par l'administrateur à l'ouverture du compte (sous GNU/Linux dans le fichier `/etc/passwd`)

- Affichage : `echo $SHELL`

- Modification : `chsh -s path_to_shell` (ou `ypchsh`)
  Modification conservée et valable pour les prochaines connexions

```
europa:~ mouronval$ echo $SHELL                    # exemple sur Mac OS X
/bin/bash
europa:/etc mouronval$ chsh -s /bin/csh
chsh: netinfo domain "." updated
```

L'environnement de travail : changer de shell

Shell interactif ordinaire

- Lancé à l'ouverture d'un terminal ou en tapant le nom du shell dans un terminal
- Connaître le shell courant : `ps` (ou `echo $0`)
  - Rappel : en général (mais modifiable) pour les utilisateurs
    - prompt "$" pour les Bourne shells
    - prompt "%" pour les C-shells
- Changer de shell en cours de session (non permanent) :
  - Taper simplement le nom du shell parmi ceux disponibles (`bash`, `tcsh`, `ksh`…)
  - Chaque nouveau shell s'empile au dessus du précédent
  - Sortir du shell courant : `exit`

L'environnement de travail : changer de shell (exemples)

Exemple de changement de shell interactif ordinaire

```
Last login: Mon Jun 21 12:46:45 on ttyp1          # ouverture d'un terminal sur Mac OS X
Welcome to Darwin!
europa:~ mouronval$ ps
  PID  TT  STAT      TIME COMMAND
  603  p1  S      0:00.02 -bash                    # bash est le shell courant
europa:~ mouronval$ echo $0
-bash                                             # même info
europa:~ mouronval$ echo $SHELL                   # shell de connexion
/bin/bash
europa:~ mouronval$ ksh                            # passage en ksh
$ ps
  PID  TT  STAT      TIME COMMAND
  603  p1  S      0:00.02 -bash                    # notez l'empilement des shells
  627  p1  S      0:00.01 ksh
$ echo $0                                         # ksh est  maintenant le shell courant
ksh
$ echo $SHELL                                     # shell de connexion (inchangé)
/bin/bash
$ exit                                            # retour au shell précédent (bash)
europa:~ mouronval$ ps
  PID  TT  STAT      TIME COMMAND
  603  p1  S      0:00.02 -bash
europa:~ mouronval$ echo $0
-bash
```

Quel shell choisir ?

- sh (Bourne) : syntaxe de script la plus interprétable par les autres (POSIX, bash, ksh)
- bash : le plus utilisé sur GNU/Linux et Mac OS X
- ksh : utilisé sur les Unix commerciaux
- csh et tcsh moins utilisés

⇒ **bash et ksh généralement recommandés** (plus de fonctionnalités que sh, bonnes compatibilité et disponibilité) pour les scripts

L'environnement de travail : classification des variables

- Classification des variables
  - utilisateur (définies par l'utilisateur)
  - système (variables réservées du shell) et spéciales

    Informations nécessaires au fonctionnement de l'interpréteur et/ou de certains programmes lancés à partir de celui-ci

    Variantes suivant les shells

- De type chaînes de caractères

- Nom des variables sensible à la casse (MAJ/min)

L'environnement de travail : variables (exemples)

- Lister les variables connues du shell courant : `set`

Exemple sur Mac OS X, en bash et tcsh

```
europa:~ mouronval$ set
BASH=/bin/bash
HOME=/Users/anne-sophiemouronval
LOGNAME=mouronval
MAILCHECK=60
PATH=/sw/bin:/sw/sbin:/Applications/CEI/bin:/bin:/sbin:/usr/bin:/usr/sbin (...)
PS1='\h:\w \u\$ '
PWD=/Users/anne-sophiemouronval
USER=mouronval
(...)

europa:~ mouronval$ tcsh
europa:~ mouronval% set
path    (/Applications/CEI/bin /sw/bin /sw/sbin (...))
prompt  [%m:%c3] %n%#
tcsh    6.12.00
user    mouronval
(...)
```

L'environnement de travail : variables du shell

- Contenu des principales variables réservées (dépend du shell)
  - USER et LOGNAME : nom de login de l'utilisateur connecté
    (USER utilisé par programmes BSD et LOGNAME par ceux issus de System V)
  - HOME : répertoire de login de l'utilisateur (homedir)
  - SHELL : nom du shell de login
  - HOSTNAME : nom de la machine
  - DISPLAY : identifiant du terminal d'affichage à utiliser
    dans le gestionnaire de fenêtres (X11)
  - PWD : répertoire de travail courant
  - **PATH** : liste des répertoires de recherche des commandes externes
  - MANPATH : liste des répertoires de recherche des pages de manuel
  - PS1 (bash, ksh…) et prompt (csh…) : mise en forme du prompt de niveau 1
- Chaque utilisateur peut définir ses variables et modifier les variables réservées

- Initialiser et modifier une variable

bash et ksh	csh et tcsh
`VARNAME=string` Attention : pas d'espace autour du signe =	`set VARNAME = string` (Aucun ou 2 espaces autour du signe =)

  - Utiliser les simples quotes '' si caractères spéciaux

- Afficher le contenu d'une variable : `echo $VARNAME`

Exemple en bash

```
europa:~ mouronval$ echo $USER
mouronval
europa:~ mouronval$ USER= myname                    # espace après signe = … erreur !
-bash: myname: command not found
europa:~ mouronval$ USER=myname
europa:~ mouronval$ echo $USER
myname
europa:~ mouronval$ NEW_VAR='nouvelle variable'  # exemple d'utilisation de quotes
europa:~ mouronval$ echo $NEW_VAR
nouvelle variable
```

Cas de la variable PATH : attention, ne pas écraser son contenu !

```
europa:~ mouronval$ echo $PATH
/sw/bin:/sw/sbin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:(...)

europa:~ mouronval$ which ls       # which cherche ls dans les répertoires de PATH
/bin/ls

europa:~ mouronval$ ls
Desktop                          README.txt
(...)

europa:~ mouronval$ PATH=$HOME/mes_scripts       # écrasement de PATH  (syntaxe de bash)

europa:~ mouronval$ echo $PATH
/Users/anne-sophiemouronval/mes_scripts

europa:~ mouronval$ ls                                  # le shell ne connaît plus ls !
-bash: ls: command not found
```

Si la variable PATH est écrasée, le shell ne connaît plus les commandes externes se trouvant dans /bin, /sbin… !!!

Remarque : `echo` est une commande *interne* au shell (*cf.* annexes)

Cas du PATH : ajouter le répertoire au contenu précédent

- bash, ksh : `PATH=$PATH:new_dir` ou `PATH=new_dir:$PATH`
- csh, tcsh : `set path=($path new_dir)` ou `set path=(new_dir $path)`
- Différence entre les 2 écritures : l'ordre de recherche des commandes

```
europa:~ mouronval$ echo $PATH
/sw/bin:/sw/sbin:/bin:/sbin:/usr/bin:/usr/sbin:(...)

europa:~ mouronval$ PATH=$PATH:$HOME/mes_scripts        # ajout au contenu précédent

europa:~ mouronval$ echo $PATH
/sw/bin:/sw/sbin:/bin:/sbin:/usr/bin:/usr/sbin:(...):/Users/annesophiemouronval/mes_scr
ipts

europa:~ mouronval$ ls                                  # ls encore trouvée
(...)
```

- Remarque valable pour les variables contenant des chemins : MANPATH (pages de `man`), LD_LIBRARY_PATH (bibliothèques dynamiques) ou DYDLD_LIBRARY_PATH (sous Mac OS X)…

L'environnement de travail : exportation des variables

- Par défaut, variables définies au niveau du shell : variables locales, inconnues des commandes lancées à partir de celui-ci (sous-shell, programmes)

- Solution : exporter la variable

Shell	bash et ksh	csh et tcsh
Exportation seule	`export VARNAME`	`setenv VARNAME`
Initialisation et exportation	**`export VARNAME=string`**	**`setenv VARNAME string`**

```
europa:~ mouronval$ NEW_VAR=email                    # en shell bash
europa:~ mouronval$ echo $NEW_VAR
email
europa:~ mouronval$ tcsh                             # sous-shell tcsh
[europa:~] mouronva% echo $NEW_VAR
tcsh: NEW_VAR: Undefined variable.
[europa:~] mouronva% exit
exit
europa:~ mouronval$ export NEW_VAR                   # retour au shell bash et export
europa:~ mouronval$ tcsh                             # sous-shell tcsh
[europa:~] mouronva% echo $NEW_VAR                   # la variable est maintenant connue
email
```

L'environnement de travail : exportation des variables (exemple)

- Cas du PATH : attention à ne pas l'écraser à l'exportation
  Ajouter le nouveau répertoire au contenu précédent :
  - bash, ksh : `export PATH=$PATH:new_dir` ou `export PATH=new_dir:$PATH`
  - csh, tcsh : `setenv PATH $PATH:new_dir` ou `setenv PATH new_dir:$PATH`
    (en pratique, inutile : `path` est automatiquement exportée en csh et tcsh)
- Autres variables souvent exportées
  - OMP_NUM_THREADS (nombre de threads pour les calculs utilisant OpenMP)
  - LD_LIBRARY_PATH ou DYDL_LIBRARY_PATH (sous Mac OS X)

L'environnement de travail : variables exportées (suite)

- Afficher l'ensemble des variables exportées

Nota : en général, pour les C-shells, variables locales au shell courant en minuscules et variables exportées en MAJUSCULES (l'exportation crée une 2ème variable)

Shell	bash et ksh	csh et tcsh
Commande	`env`	`env`

```
europa:~ mouronval$ env                          #  en bash, liste des variables exportées
MANPATH=/sw/share/man:/usr/share/man:/usr/local/share/man:(…)
SHELL=/bin/bash
USER=mouronval
PATH=/sw/bin:/sw/sbin:/Applications/CEI/bin:/bin:/sbin:/usr/bin:(…)
(…)

europa:~ mouronval$ NEW_VAR=email
europa:~ mouronval$ set |  grep NEW_VAR   # rappel : grep recherche une chaîne de caractères
NEW_VAR=email                            # NEW_VAR est visible du shell courant …
europa:~ mouronval$ env |  grep NEW_VAR   # mais ne figure pas dans la liste des variables exportées
europa:~ mouronval$ export NEW_VAR
europa:~ mouronval$ env |  grep NEW_VAR
NEW_VAR=email
```

L'environnement de travail : les alias

Créer (ou redéfinir) ses propres commandes à partir de commandes existantes : "les alias"

- Afficher la liste complète des alias ou l'alias correspondant : `alias [alias_name]`
  Des alias peuvent exister par défaut

- Modifier ou créer un alias :
  `alias alias_name 'command [options] [args]'` (en csh, tcsh)
  `alias alias_name='command [options] [args]'` (en ksh, bash)

- Utilisation : taper le nom de l'alias

- Suppression d'un alias : `unalias alias_name`

- Portée d'un alias : par défaut, celle du shell dans lequel il a été défini

- En cas de redéfinition d'une commande (`ls`…), l'alias prime sur la commande originale

- Conserver la définition des alias : utiliser les fichiers d'environnement

L'environnement de travail : les alias (exemples)

Exemple 1 : définir un alias en tcsh pour Matlab et un autre pour la connexion au calculateur Igloo

```
europa:~ mouronval$ tcsh

europa:~ mouronval% alias m75 'linux32 /hosts/thuya/MSS/APP/matlab_R2007b/bin/matlab -
nojvm -c /hosts/thuya/MSS/APP/matlab7/etc/license.dat -arch=glnx86'

europa:~ mouronval% alias igloo 'ssh -X mouronv@igloo.calcul.ecp.fr'

europa:~ mouronval% igloo                                    # utilisation de l'alias « igloo »
password:

europa:~ mouronval% alias                                    # affiche les alias
m75     linux32 /hosts/thuya/MSS/APP/matlab_R2007b/bin/matlab -nojvm -c
/hosts/thuya/MSS/APP/matlab7/etc/license.dat -arch=glnx86
igloo ssh -X mouronv@igloo.calcul.ecp.fr
```

Exemple 2 : redéfinition de la commande unix `rm` (en bash)

```
europa:~ mouronval$ alias                          # ne retourne rien : aucun alias
europa:~ mouronval$ rm texte                       # rm  ne demande pas confirmation
europa:~ mouronval$ alias rm='rm -i'
europa:~ mouronval$ alias rm                       # affiche l'alias correspondant à « rm »
alias rm='rm -i'
europa:~ mouronval$ touch texte                    # creation d'un fichier vide « texte » par touch
europa:~ mouronval$ rm texte                       # rm (alias) demande maintenant confirmation
remove texte? y
```

L'environnement de travail : les alias (exemples)

Exemple 3 : suite à l'exemple 2, passage de bash en tcsh, l'alias n'est pas connu pour tcsh

```
europa:~mouronval$ alias                    # alias définis pour ce shell
alias rm='rm-i'
europa:~mouronval$ tcsh                     # passage de bash a tcsh (empilement)
europa:~mouronval% alias                    # ne retourne rien : aucun alias
europa:~mouronval% exit                     # retour au shell précédent (bash)
europa:~mouronval$ alias                    # on retrouve ses alias
alias rm='rm-i'
europa:~mouronval$ unalias rm               # exemple de destruction d'un alias
europa:~mouronval$ alias                    # ne retourne rien : plus d'alias
```

Exemple 4 : existence d'alias par défaut (cas du ksh sur Mac OS X)

```
europa:~mouronval% ksh
$ alias
(…)
stop='kill -s STOP'
type='whence -v'
```

Fichiers d'environnement : scripts shell lus **systématiquement** et **automatiquement** à la connexion ou l'exécution d'un nouveau shell

- Fichiers système :
  - gérés par l'administrateur
  - lus par tous les utilisateurs à la connexion
  - mettent en place un environnement général pour tous les utilisateurs

- Fichiers utilisateur :
  - à créer par l'utilisateur si besoin
  - propres à chaque utilisateur
  - fichiers dans le homedir de l'utilisateur commençant par un "." visibles via `ls -a`
  - permet de personnaliser son environnement : conserver les alias, variables…

- Fichiers lus dépendent :
  - du shell (bash, ksh, csh…)
  - du type de shell (shell de connexion, shell ordinaire interactif lancé « à la main » dans un terminal, shell non-interactif lancé d'un script…)

- Fichiers d'environnement en fonction du shell (*cf.* http://en.wikipedia.org/wiki/Unix_shell)

Shell	Fichiers système	**Fichiers utilisateur (avec chemin absolu)**
ksh	`/etc/profile`	**`~/.profile`** et **`~/.kshrc`**
bash		`~/.bash_profile` ou `~/.bash_login` ou **`~/.profile`** et **`~/.bashrc`**
csh	`/etc/csh.login` ou `/etc/login` ou `/etc/.login` ou `/etc/.csh.cshrc` ou `/etc/cshrc`	**`~/.login`** et **`~/.cshrc`**
tcsh		**`~/.login`** et **`~/.tcshrc`** ou `~/.cshrc`

- Rappel : «~» fait référence au homedir (comme la variable `HOME`)
- Remarque : il existe des fichiers pour la déconnexion également (`~/.logout…`)

L'environnement de travail : fichiers d'environnement (contenu)

- Que mettre dans `~/.profile` (ou `~/.login`) ?
  - Définition et exportation des variables (PATH, LD_LIBRARY_PATH…)
  - Redéfinition de certains paramètres système :
    `umask` (droits par défaut fichiers à leur création),
    les caractéristiques du terminal…

  Pas d'alias dans ces fichiers

- Que mettre dans `~/.bashrc` (ou `~/.kshrc` ou `~/.cshrc` ou `~/.tcshrc`) ?
  - Alias et fonctions
  - Options du shell (`ignoreoff`…)
    liste de options du shell (bash, ksh) : `set -o`

L'environnement de travail : fichiers d'environnement (relecture)

- Fichiers lus à des moments précis (connexion, ouverture de terminal *etc*)

  ⇒ Modifications effectuées prises en compte

  - à la reconnexion, l'ouverture d'un nouveau terminal… (suivant le fichier)
  - ou en forçant leur relecture (« ressourcer le fichier »)

Shell	`Commande` pour relecture	Exemples
ksh	`.` (point)	`. $HOME/.profile` et `. $HOME/.kshrc`
bash	`.` (point) ou `source`	`. ~/.profile` ou `source ~/.profile` et `. ~/.bashrc` ou `source ~/.bashrc`
csh	`source`	`source ~/.login` et `source ~/.cshrc`
tcsh	`source`	`source ~/.login` et `source ~/.tcshrc`

Exemple de fichiers d'environnement pour bash

```
europa:~ mouronval$ cat ~/.bashrc
# Mes alias                                          # commentaires
alias rm='rm -i'
alias ls='ls -G'                                     # ls avec couleurs suivant type de fichiers

europa:~ mouronval$ cat ~/.profile
# Si le fichier ~/.bashrc existe, on l'éxecute              # commentaires
if [ -f ~/.bashrc ]; then
       . ~/.bashrc
fi
umask 027                              # droits des fichiers créés : (rwx pour u, r pour g, - pour o)
PATH=$PATH:/usr/local/bin:$HOME/bin
export PATH

europa:~ mouronval$ alias                            # Liste des alias
alias ls='ls -G'
alias rm='rm -i'

europa:~ mouronval$ cat >> ~/.profile                # modification du fichier par cat
PS1='$PWD '                                          # PS1 : prompt
export PS1
^D                                                   # indique la fin du texte entré sur stdin pour cat

europa:~ mouronval$ source .profile                  # force la relecture du fichier

/Users/anne-sophiemouronval                          # modification du prompt prise en compte
```

A retenir

Partie 2

- Système de fichiers (suite)
  - Commandes complémentaires : `which` et `find`
  - Espace disque `du` et `df`
- Gestion des processus
  - Caractéristiques : hiérarchie (PID…)
  - Contrôle (`top` et `kill`)
- Environnement de travail : le shell
  - Les types de shell (bash en particulier)
  - Notion de variables (PATH) et alias
  - Fichiers de démarrage

Formations suivantes

- Makefiles (niveau débutant et niveau avancé)
- Gestion de versions
- Formations IDRIS (gratuites, à Orsay, calendrier sur http://www.idris.fr/)
  - Fortran95-1, Fortran95-2
  - Langage C
  - MPI
  - OpenMP

Babel, IBM Blue Gene/P, 40.960 coeurs, IDRIS

Références

- *Unix & Linux, Utilisation et administration*, J.-M Léry.
  Disponible à MSSMat

- *Linux, Maîtriser l'administration du système*, S. Rohaut.
  Disponible à MSSMat

- *Programmation shell sous Unix/Linux sh (Bourne), ksh, bash*, C. Deffaix Rémi
  Disponible à MSSMat

- *Les processus*.
  http://www.admin-sys.org/spip.php?article71

- *Le Shell*, P. Dax.
  http://www.infres.enst.fr/~dax/polys/shell/

Annexes : liens symboliques

Créer un lien symbolique vers un fichier (ou répertoire) : `ln -s filename linkname`

- Lien par nom, *ie.* redirection (par opposition au lien physique par inode, moins utilisé)
- Si l'original est modifié, le lien l'est aussi (différent d'une copie)
- La destruction du lien symbolique n'affecte pas l'original
- Exemple d'utilisation : mettre des liens des binaires exécutables comme Matlab dans `/usr/bin` (qui fait généralement partie du PATH)

```
europa:~ mouronval$ cat  > README.txt
Un fichier readme
europa:~ mouronval$ ln -s README.txt lien_README
europa:~ mouronval$ ls -l *README*
-rw-r--r--   1 mouronva  anne-sop  18 Oct  5 15:23 README.txt
lrwxr-xr-x   1 mouronva  anne-sop  10 Oct  5 15:24 lien_README -> README.txt
europa:~ mouronval$ cat >> README.txt
ajout de texte dans README.txt
europa:~ mouronval$ cat lien_README
Un fichier readme
ajout de texte dans README.txt
europa:~ mouronval$ rm lien_README
europa:~ mouronval$ cat README.txt
Un fichier README.txt
ajout de texte dans README.txt
```

Notion de « cluster »

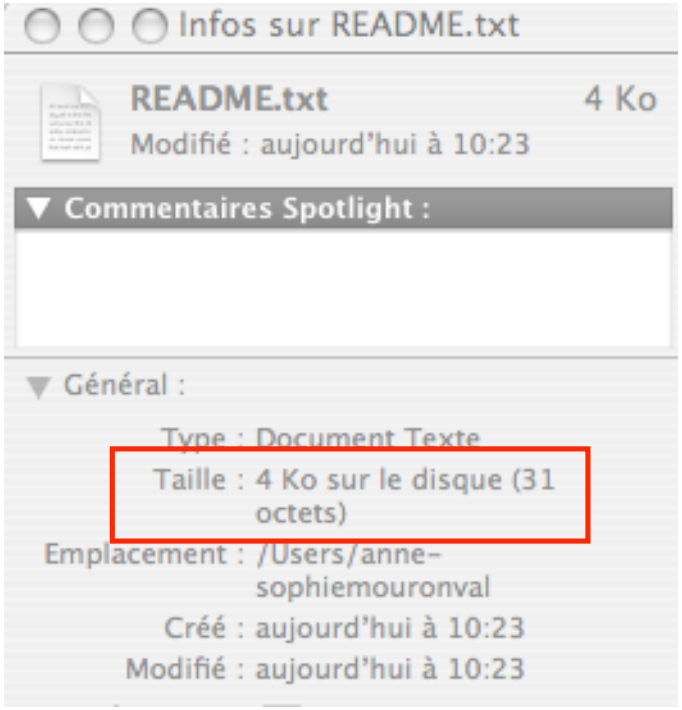
- Cluster : « unité d'allocation » sur laquelle le système de fichiers (FS) est basé
- Un cluster ne peut être utilisé que par un seul fichier
- Taille des clusters dépend du FS, de la taille de la partition, des fichiers attendus…
  - Taille minimale : 512 octets
  - HFS+ (FS de Mac OS X) : en général clusters de 4 Kos
  - ext3 (FS dominant pour GNU/Linux) : en général clusters de 4 Kos

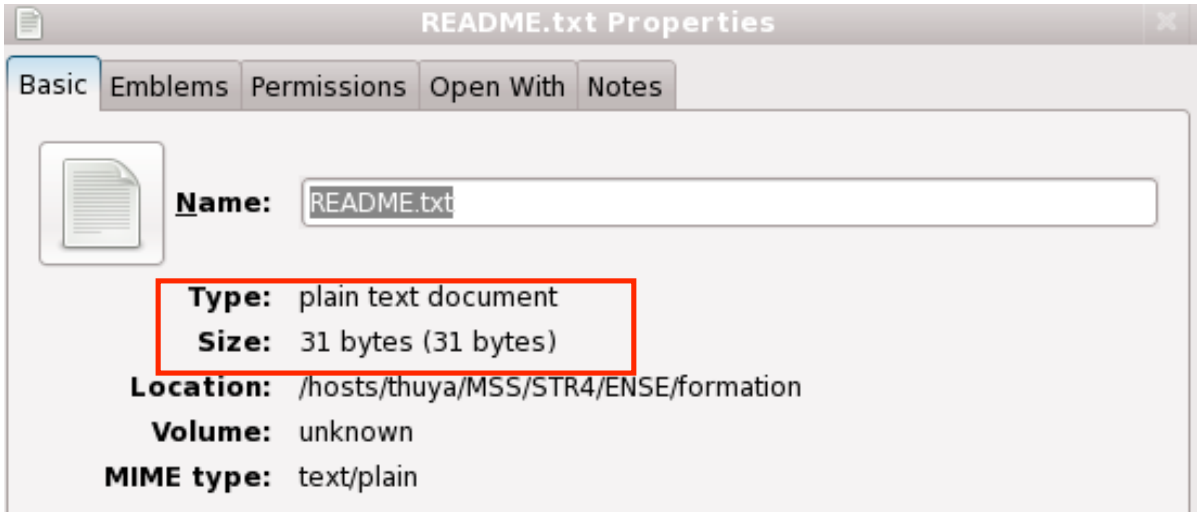
Annexes : taille réelle vs taille occupée sur le disque

Attention à l'interprétation des tailles données par les interfaces ou les commandes

Exemple : fichier README.txt contenant 31 caractères (soit 31 octets=31 bytes)

- Connaître la taille du fichier en utilisant une interface graphique
  - Mac OS X : Se placer sur le fichier et taper ⌘I
    (ou `Fichier > Lire les informations`) : notez « sur le disque »
  - GNU/Linux : Utiliser `Nautilus`





Annexes : taille réelle vs taille occupée sur le disque (suite)

Exemple : fichier README.txt contenant 31 caractères (soit 31 octets=31 bytes)

- Connaître la taille du fichier via les commandes

  Utiliser `man` pour vérifier le sens des options (taille réelle ou occupée ? en blocs ? …)

Exemple sous Mac OS X (10.4)

```
europa:~ mouronval$ man wc
(...)
 -c      The number of bytes in each input file is written to the standard output.

europa:~ mouronval$ man ls
(...)
If the -l option is given, the following information is displayed for each file: (...),
number of bytes in the file (...)
-s  Display the number of file system blocks actually used by each file, in units of
    512 bytes, where partial units are rounded up to the next integer value.

europa:~ mouronval$ man du
The du utility displays the file system block usage for each file argument
(...)
BLOCKSIZE (...) If BLOCKSIZE is not set, and the -k option is not specified, the block
counts will be displayed in 512-byte blocks.
(...)
-h   "Human-readable" output.  Use unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte,
    Terabyte and Petabyte
```

Annexes : taille réelle vs taille occupée sur le disque (suite)

Exemple sur une machine sous Mac OS X (10.4)

FS, HFS+, taille des clusters de 4 Kos (4*1024 bytes = 8 blocs de 512 bytes)

```
europa:~ mouronval$ wc -c README.txt              # Nbr de bytes contenus ds le fichier
      31 README.txt

europa:~ mouronval$ ls -l  README.txt             # Nbr de bytes contenus ds le fichier
-rw-r--r--   1 mouronva  anne-sop  31 Jun  9 10:23 README.txt

europa:~ mouronval$ ls -s  README.txt             # Nbr de blocs de 512 Mégabytes occupés
8 README.txt

europa:~ mouronval$ du README.txt                 # Nbr de blocs de 512 Mégabytes occupés
8       README.txt

europa:~ mouronval$ du -h README.txt              # Espace occupé sur le disque en Kilobyte …
4.0K    README.txt
```

Annexes : taille réelle vs taille occupée sur le disque (suite)

Exemple sur une machine GNU/Linux Fedora 9
FS ext3, taille des clusters de 4 Kos

Ici `man du` et `man ls` n'indiquent pas la taille par défaut des blocs (1 Ko)

```
formation@orchis% wc -c README.txt          # Nbr de bytes contenus ds le fichier
31 README.txt

formation@orchis% ls -l README.txt          # Nbr de bytes contenus ds le fichier
-rw-r--r-- 1 formation DEA 31 2009-09-18 15:41 README.txt

formation@orchis% ls -s README.txt          # Nbr de blocs de 1 Kilobyte occupés sur le disque
4 README.txt

formation@orchis% du README.txt             # Nbr de blocs de 1 Kilobyte occupés sur le disque
4 README.txt

formation@orchis% du -h README.txt          # Espace occupé sur le disque en Kilobyte …
4.0K    README.txt
```

Annexes : Kio vs Ko…

Attention à la définition de Ko, Mo (pour achat de baies de disque, disque dur…)

Certains parlent en norme SI (puissance de 10 pour Ko et de 2 pour Kio)
et la majorité des utilisateurs en « usage traditionnel » (puissance de 2 pour Ko)

Multiples d'octets
tels que définis par IEC 60027-2

Préfixe SI			Préfixe binaire		
Nom	Symbole	Valeur	Nom	Symbole	Valeur
kilooctet	Ko	10^3	kibioctet	Kio	2^{10}
mégaoctet	Mo	10^6	mébioctet	Mio	2^{20}
gigaoctet	Go	10^9	gibioctet	Gio	2^{30}
téraoctet	To	10^{12}	tébioctet	Tio	2^{40}
pétaoctet	Po	10^{15}	pébioctet	Pio	2^{50}
exaoctet	Eo	10^{18}	exbioctet	Eio	2^{60}

Multiples d'octets
Usage traditionnel

Nom	Symbole	Valeur
kilooctet	Ko	2^{10}=1024 octets
mégaoctet	Mo	2^{20}=1024 Ko
gigaoctet	Go	2^{30}=1024 Mo
téraoctet	To	2^{40}=1024 Go
pétaoctet	Po	2^{50}=1024 To
exaoctet	Eo	2^{60}=1024 Po

Tiré de http://fr.wikipedia.org/wiki/Octet

Annexes, type des commandes : `type`

Commandes externes et internes : `type`

- externes : fichiers localisés dans l'arborescence
- internes : intégrées au processus shell
- certaines commandes peuvent avoir les 2 implémentations

Exemple sur Mac OS X en bash

```
europa:~ mouronval$ type cd
cd is a shell builtin                    # cd est interne

europa:~ mouronval$ type ls
ls is /bin/ls                            # ls est externe

europa:~ mouronval$ which cd             # cd a aussi une implémentation externe
/usr/bin/cd
```

Annexes, gestion des processus : `time`

Mesurer la durée d'exécution d'une commande ou d'un programme : `time`

- `real` : temps horloge (*wall clock time*), durée totale d'exécution de la commande
- `user` : temps CPU nécessaire pour exécuter le processus
- `sys` : temps relatif aux appels système

```
europa:~ mouronval$ time ./code_seq_c < entree.txt > sortie.txt
  real    65m38.580s
  user    64m35.849s
  sys     0m39.685s
```

Annexes, gestion des processus : `ulimit`

Contrôler les ressources disponibles pour le shell et ses processus fils : `ulimit`

- `-a` : *all*, affiche toutes les limites
- utile par exemple pour
  - limiter la taille des fichiers cores (fichiers générés lors d'un « plantage » d'un calcul)
  - limiter la taille d'un fichier créé par le shell
    (en phase de développement, cela évite la création d'un fichier énorme si le programme boucle en écriture)

Exemple sur une machine sous Mac OS X

```
europa:~ mouronval$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) 6144
file size               (blocks, -f) unlimited
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 256
pipe size            (512 bytes, -p) 1
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 266
virtual memory          (kbytes, -v) unlimited
```

Annexes : fichiers d'environnement (cf. référence N°1)