Unix : cours de survie

Gregory Sainte-Luce - Olivier DELHOMME

Centre des matériaux, école nationale supérieure des mines de paris

Rentrée 2015

Définitions

Conventions

- `«toto», «sly»` font référence à des utilisateurs,
- `«mygroup», «admin»` font référence à des groupes d'utilisateurs,
- `«monfichier», «monautrefichier»` font référence à des noms de fichiers,
- `«mondossier», «monautredossier»` font référence à des noms de dossiers.

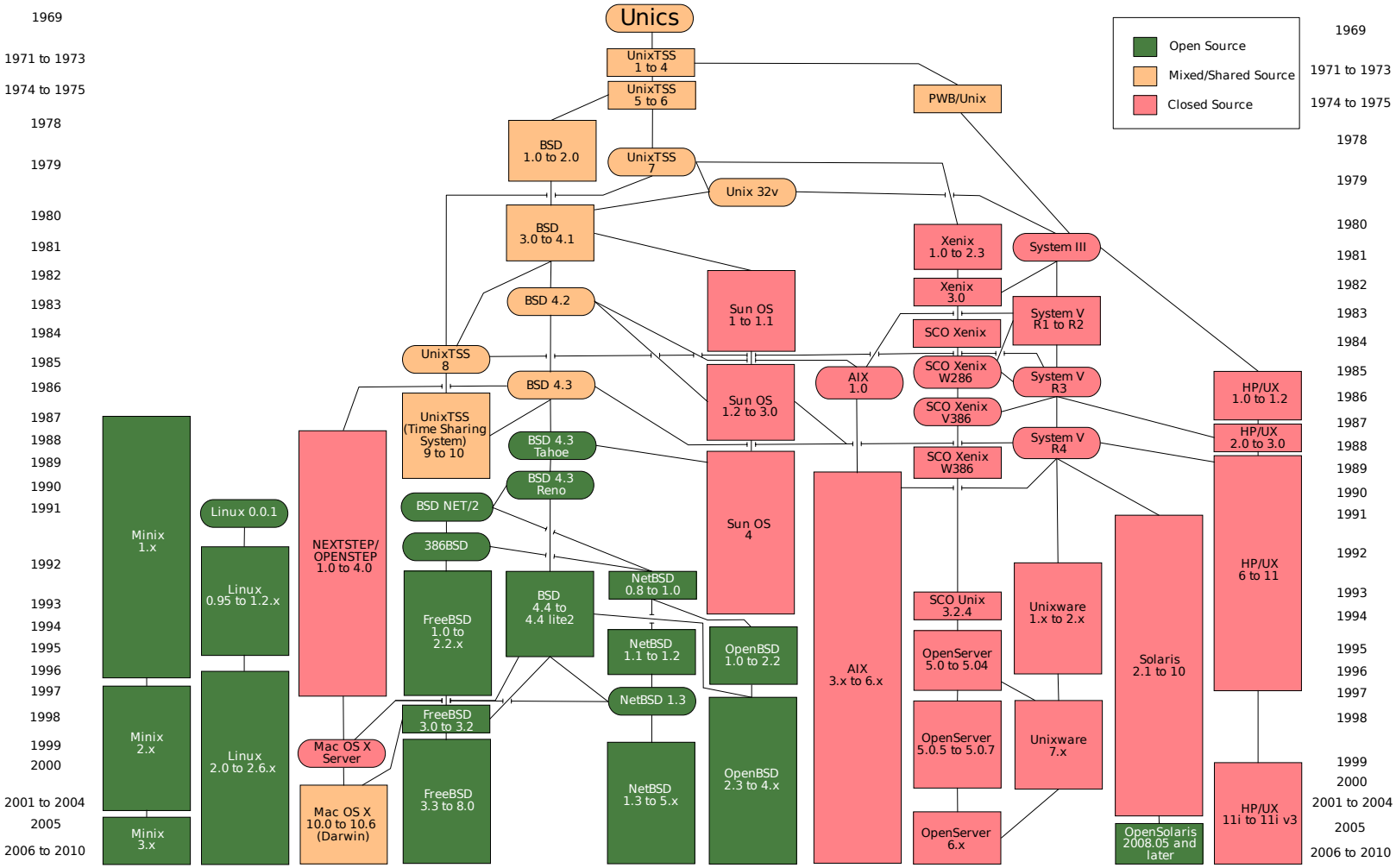
Historique

Quelques dates

- Années 60 : création de MULTICS,
- 1969 Création de UNICS (AT&T),
- 1970 Création d'UNIX (en assembleur pour PDP11),
- 1972 Ré-écriture en C (domaine public grâce à un décret anti-trust sur AT&T),
- 1980 –> diffusion dans toutes les universités,
- 1983 Publication du system V et levée du décret : UNIX est vendu par AT&T,
- 1991 Sortie de GNU/Linux 0.0.1,
- 1993 Tous les unix sont basés sur System V,
- 2002 Mac OS X (dont le coeur est un BSD),
- 2012 Unix a 42 ans (101010)
- 2012 GNU/Linux a 21 ans

Historique

Arbre généalogique



Environnement de travail

Quel environnement aujourd'hui

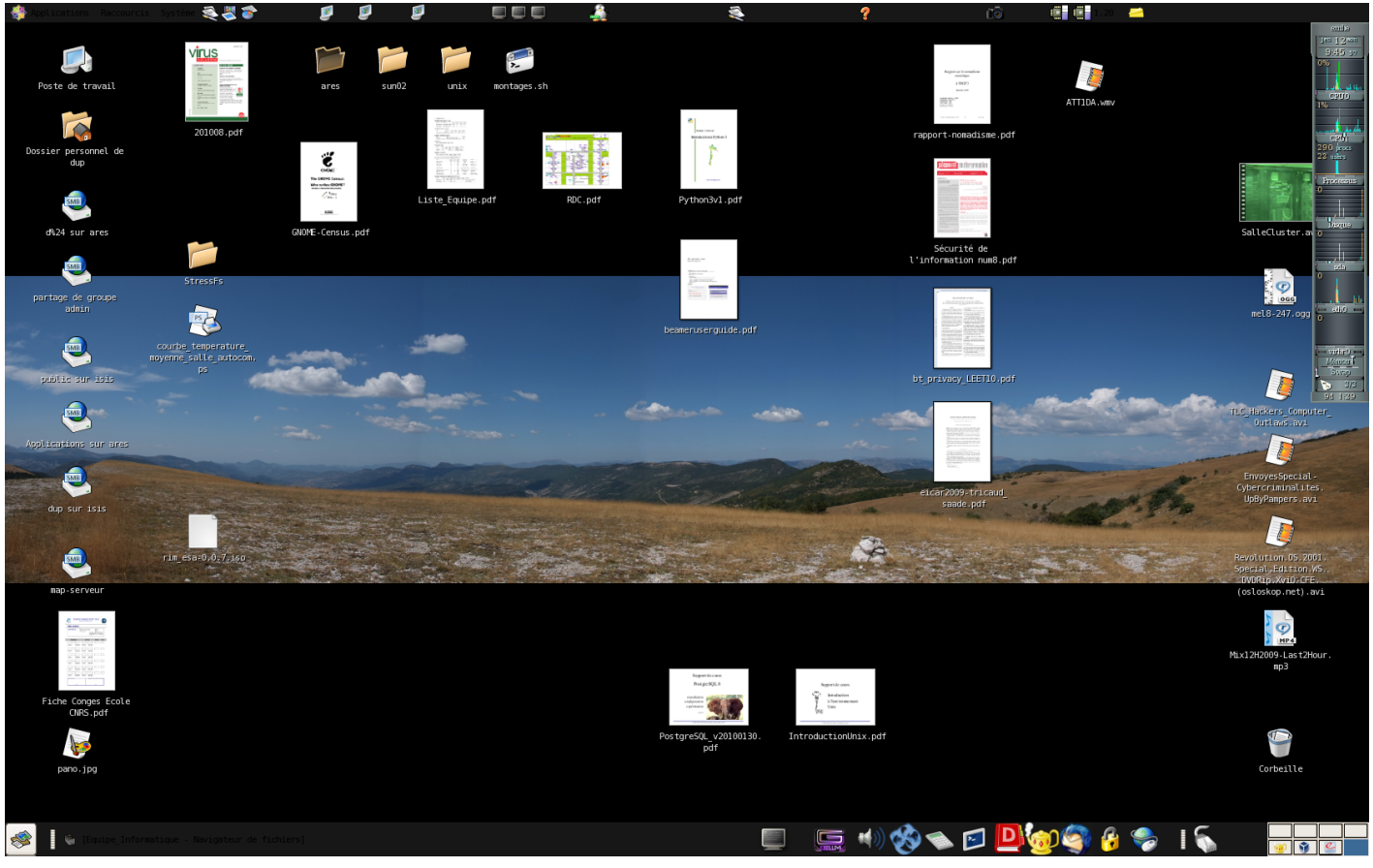


Figure: Environnement graphique

Environnement de travail

Quel environnement aujourd'hui

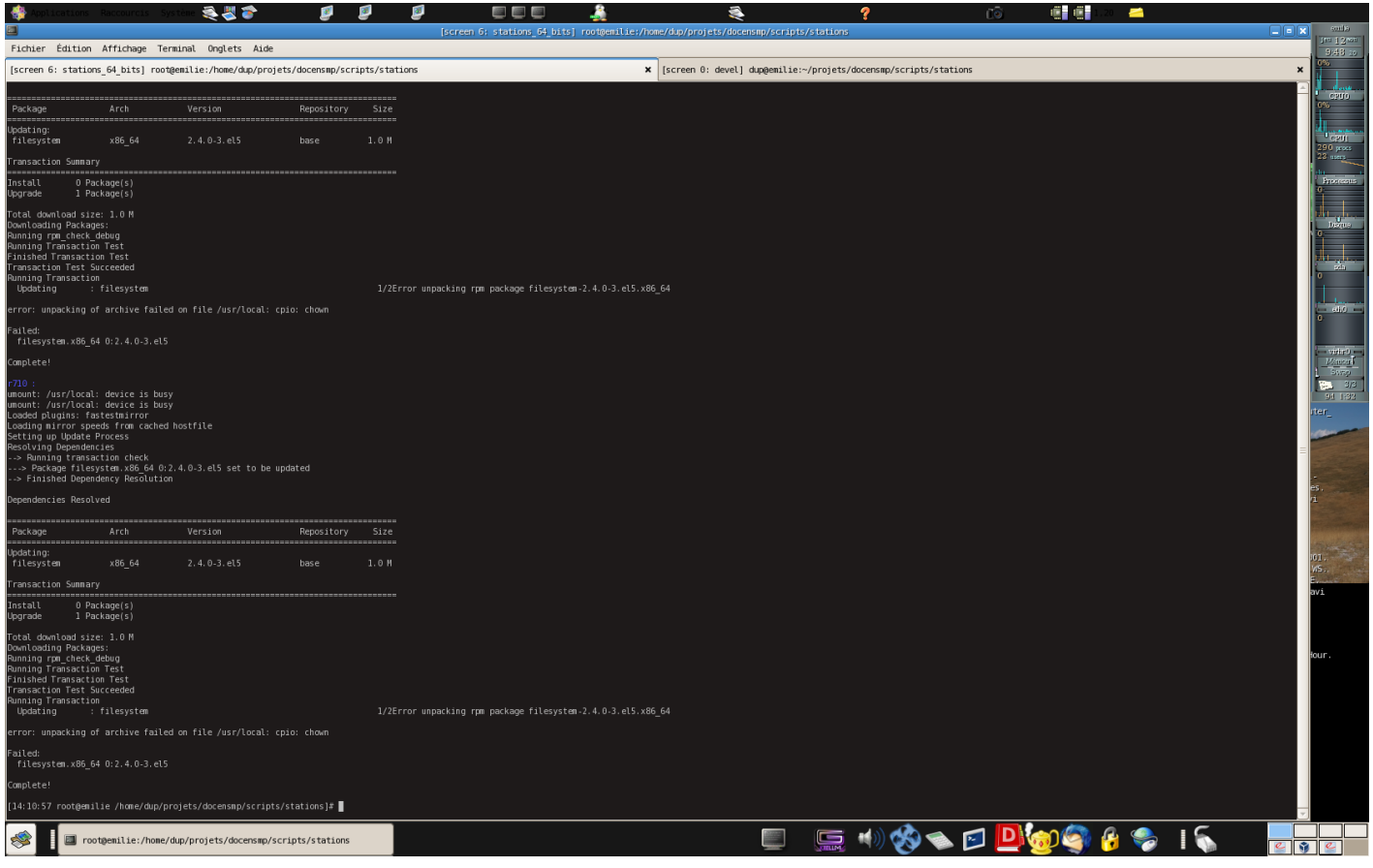


Figure: Ligne de commande

Astuces diverses

- La touche majuscule permet de faire les majuscules accentuées ÉÈÀÛ...,
- La sélection avec la souris puis le clic du bouton du milieu réalise un copier/coller,
- L'appui sur la touche «chapeau» puis sur un chiffre (hors pavé numérique) permet de faire un exposant : ^{123456789}.

Pourquoi la ligne de commande ?

- Pas d'équivalent graphique aussi puissant,
- Mise en oeuvre immédiate,
- Le travail à distance est facilité (plus rapide qu'en graphique) et c'est parfois la seule façon d'accéder à un serveur (par exemple un serveur de calcul),

Présentation de la ligne de commande

- Elle est formée par une invite de commande : `tp1.toto`[31]%
  - Cette invite de commande est modifiable,
  - Ici il indique le nom de la machine et le nom de login et le numéro de la commande entre crochets,
  - Elle est fournie par un program que l'on nomme le shell,
- On peut saisir du texte et des commandes sur cette ligne comme par exemple `man`, `pwd`, `clear`.

Le Terminal - La ligne de commande

Généralités sur les commandes

- Une commande est formée ainsi : **commande** *arguments*
  - L'espace est un séparateur (entre la commande et les arguments et pour les arguments),
  - Les arguments peuvent être optionnels,
  - Les arguments peuvent êtres des options ou des noms de fichiers,
  - Chaque commande a ses propres options,
  - Les options courtes s'indiquent avec un seul - et les longues avec deux,
  - Les options courtes peuvent s'additionner,
  - **Souvent les options *-h*, *–help* fournissent l'aide de la commande**,
  - **La commande `man` suivie du nom d'une commande donne l'aide de cette dernière.**,
  - Un double - optionnel permet d'indiquer la fin des options.

- Il n'y a pas de lecteurs sous Linux / Unix,
- Tout est sous forme d'arbre dont les noeuds sont les dossiers et les feuilles les fichiers,
- Un nouveau «lecteur» est «raccroché» sur un dossier.
- Les dossiers `.` et `..` sont spéciaux et présents dans tous les dossiers.

Système de fichiers

L'arborescence du système de fichier au centre

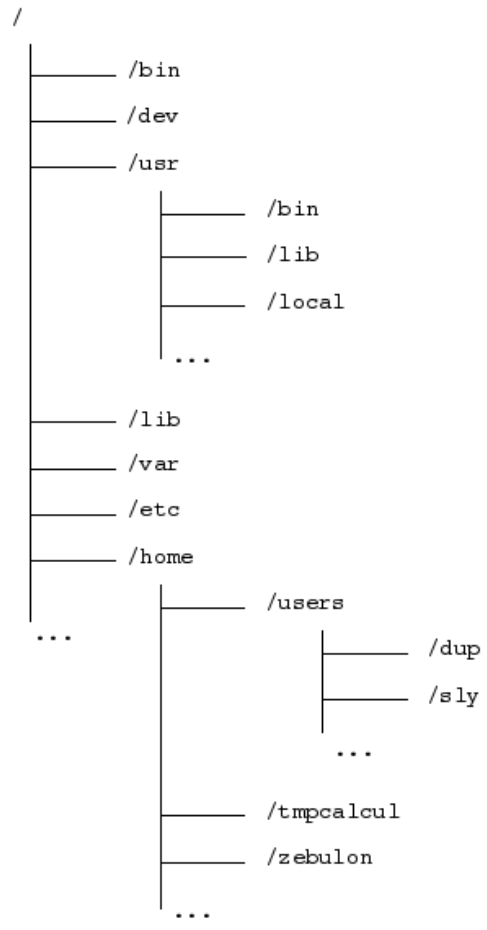


Figure: arborescence

Quelques commandes utiles pour se déplacer

- `pwd` Indique de manière absolue où l'on se trouve dans l'arborescence,
- `cd` pour 'Change Directory' Permet de naviguer dans l'arborescence. `cd -` permet de revenir au dossier précédent.

Chemins relatifs et absolus :

- Il est possible de se déplacer de deux façons :
  - De manière absolue (par rapport à la racine),
  - De manière relative (par rapport à l'endroit où on se trouve dans l'arborescence),

Une commande utile pour lister les fichiers et dossiers

- `ls` Liste les fichiers et dossiers (tout est fichier !)
  - option -d pour un dossier,
  - option -a pour voir tous les fichiers y compris les cachés,
  - option -t pour classer en fonction de la date (-r pour renverser l'ordre),
  - option -s pour obtenir la taille en blocs,
  - option -l pour voir plus de détails.
- Exemples :
  - `ls -l -a`
  - `ls -lart`

Digression sur l'expansion des noms de fichiers

- On peut utiliser des méta-caractères pour indiquer un raccourci ou un ensemble de fichiers
- Les méta-caractères :
  - ~
  - *
  - ?
  - [ ]

Exemples :

- `lrwxrwxrwx`
- `drwxr-xr-x`
- `-rw-r-r-`

1 lettre et 3 groupes de 3 lettres : `d rwx r-x r-x`

- le type de fichier (-, d, l, c, b, s),
- d pour un repertoire, l pour un lien, s pour un socket , b block device, c caractere device,
- les droits sur le fichier en fonction du groupe (utilisateur, groupe, tous les autres).
- 'r' indique un droit de lecture,
- 'w' indique un droit d'écriture ou de modifier
- 'x' indique un droit d'exécution pour un fichier.
- 'x' indique le droit de traverser pour un repertoire

- la commande est `chmod` pour 'change mode',
- les lettres 'u', 'g' et 'o' indiquent respectivement l'utilisateur, le groupe et les autres,
- '+' pour ajouter des droits et '-' pour enlever des droits,
- '=' pour ecraser les droits.
- on peut indiquer l'ensemble des droits avec la notation modale.

Exemples :

- `chmod o+r monfichier`,
- `chmod g+w monfichier`,
- `chmod go-rx monfichier`.

Changer les droits en mode numerique

- la commande `chmod` peut etre egalement utiliser en mode numerique sur base octale ,
- à chaque groupe de droits on affecte une valeure numerique,
- 'R = 4 , W = 2 , X = 1' ,
- '0' pour donner aucun droit.

Exemples :

- `chmod 755 monfichier : donne au propriétaire tous les droits, aux membres du groupe et aux autres les droits de lecture et d'accès.`,
- `chmod 700 monfichier donne uniquement tous les droits au propriétaire` ,

- la commande est `chown` pour 'change owner',
- on indique le nouveau propriétaire (éventuellement le groupe) et les fichiers auxquels cela s'applique,
- l'option '-R' permet de faire les changements récursivement (pour toute une arborescence),
- la commmande `chgrp` permet de ne changer que le groupe.

Exemples :

- `chown toto monfichier`
- `chown toto:mygroup monfichier`
- `chown -R toto:admin mondossier`
- `chgrp -R mygroup monautredossier`

Recherche un fichier en fonction de critères externes

- '`find nomdudossier options`' permet de rechercher (récursivement) des fichiers en fonction de critères externes.
  - '-type' pour rechercher des fichiers d'un certain type (f pour fichier, d pour dossier, l pour lien),
  - '-size n' fichier dont la taille est au moins n,
  - '-perm -g=rx' en fonction des droits sur les fichiers (ici lecture et exécution au groupe),
  - '-user u' les fichiers doivent appartenir à l'utilisateur u,
  - '-group g' les fichiers qui appartiennent au groupe g,
  - '-exec action' exécute l'action sur chacun des fichiers trouvés.

Exemples :

- `find /etc -type d -perm -o=rx`
- `find /etc -type f -size 4k -exec cat {} \;`

Commandes pour travailler avec des fichiers (1/11)

- `cp src dst` pour 'copy' :
  - copie un fichier ou des fichiers,
  - option '-r' (récursif) pour la copie d'un dossier et de ses sous dossiers,
  - option '-p' (préserve) pour préserver les droits,
  - option '-a' (archive) pour copier tout tel quel sans changements.
- `mv src dst` pour 'move' :
  - déplace ou renomme un fichier ou un dossier.
- `mkdir nomdudossier` pour 'make directory' :
  - Une option possible est l'option '-p' (parents) qui demande la création des dossiers parents s'ils n'existent pas.

'`dst`' peut être un dossier ou un fichier. Il peut y avoir plusieurs sources mais qu'une seule destination !

Commandes pour travailler avec des fichiers (2/11)

- `rm nomfichier` pour 'remove' :
  - détruit un fichier (ou un dossier),
  - option '-r' pour supprimer des dossiers (attention la commande est récursive).
- `rmdir nomdossier` pour 'remove directory' soit détruire un dossier.

Attention

La suppression via ces commandes est définitive (il n'y a pas de dossier «Poubelle»).

Commandes pour travailler avec des fichiers (3/11)

- `file nomfichier` pour connaitre le type du fichier en fonction de son contenu,
- `cat nomfichier1 nomfichier2` pour afficher ou concatener des fichiers,
- `grep PATRON nomfichier` permet de rechercher PATRON dans un fichier et d'en afficher les correspondances.
  - '-E' pour utiliser les expressions rationnelles étendues,
  - '-r' permet d'effectuer une recherche récursive (dans les sous-dossiers),
  - '-i' ne pas tenir compte de la casse (majuscules/minuscules),
  - '-l' afficher le nom du fichier plutôt que la correspondance.

Digressions sur les expressions rationnelles (1/2))

Une expression rationnelle c'est un patron, c'est à dire un modèle qui décrit une suite de caractère formant une chaine de caractère.

- `|` indique un choix : essai|test,
- `.` indique n'importe quel caractère,
- `?` défini un groupe qui existe 0 ou une fois,
- `*` défini un groupe qui existe 0 ou n fois,
- `+` défini un groupe qui existe 1 ou n fois,
- `^` recherche une correspondance en début de ligne,
- `$` recherche une correspondance en fin de ligne,
- `[ ]` défini une classe de caractère,
- `[^ ]` défini le complément de la classe de caractère,
- `{m,n}` défini au moins m correspondances et au plus n.

Digressions sur les expressions rationnelles (2/2))

Exemples :

- `[cC]hat|[cC]hien` : chat ou Chat ou chien ou Chien (grep -i -E 'new|conf' /etc/*),
- `chu+t` chut ou chuut ou chuuuuuut, etc ...,
- `peu[xt]?` Les mots qui contiennent peu ou peux ou peut,
- `^trax$` correspond aux lignes qui ne comportent que le mot trax,
- `c[o]?nf$` cnf ou conf en fin de ligne (grep -i -E 'new$|conf$' /etc/*),
- `se[a-Z]*es` les mots qui commencent par es et finnissent par se : sees, services, sevices, ...,
- `^rc[^a-Z]{0,1}\.d` toutes les lignes qui commencent par "rc" qui ont, ou pas, un caractère qui n'est pas dans l'ensemble "a-Z" puis qui comporte un "." et un "d".

Commandes pour travailler avec des fichiers (4/11)

Tout est fichier ! *(et flux)*

- Une commande est toujours connectée à 3 «flux» :
  - le flux d'entrée (stdin) (descripteur de fichier 0),
  - le flux de sortie (stdout) (descripteur de fichier 1),
  - le flux d'erreurs (stderr) (descripteur de fichier 2),
- On peut rediriger les flux avec les signes > et < :
  - > redirige le flux de sortie vers un fichier (écrase le fichier),
  - » redirige le flux de sortie vers un fichier (concatène à la suite),
  - < redirige le flux d'un fichier vers l'entrée de la commande,

Commandes pour travailler avec des fichiers (5/11)

- le pipe '`|`' (AltGr-6) ou tube permet de passer un flux d'une commande à l'autre (pour chainer les commandes). C'est à dire passer le flux de sortie de la première commande au flux d'entrée de la seconde commande :
  - `cat nomfichier | grep essai`
- `wc` permet de compter le nombre de lignes, de mots et de caractères dans un fichier :
  - `cat nomfichier | grep essai | wc -l`

Commandes pour travailler avec des fichiers (6/11)

Voir le début ou la fin d'un fichier :

- `head`
- `tail`
- option -n pour indiquer un nombre de ligne (head et tail),
- option -f pour indiquer à tail de ne pas quitter et de suivre les ajouts dans la fin du fichier.

Voir le contenu de fichiers trop longs (commandes interactives) :

- `more`
- `less`
- `most`

Exemple : `ls -ls /dev | less`

Commandes pour travailler avec des fichiers (7/11)

- `cut` pour n'afficher que certains champs d'un fichier,
- `sed` pour traiter, à la volée, un flux (voir le man !).

Exemples :

- `cat /etc/passwd | cut -d':' -f1,3,4`,
- `ls -ls /etc/sysconfig/ | sed -n 1~2p`,
- `ls -ls /etc/sysconfig/ | sed -e s/d/Y/g`.

Référence :

- Pour sed : http://www.grymoire.com/Unix/Sed.html.

Commandes pour travailler avec des fichiers (8/11)

- `sort` pour trier le contenu des fichiers.
  - option '-k' pour indiquer le champ qui servira de clef de tri,
  - option '-n' pour indiquer un tri numérique,
  - option '-t' pour indiquer le séparateur de champ (l'espace par défaut),
  - option '-r' pour renverser l'ordre du tri.

Commandes pour travailler avec des fichiers (9/11)

Les fichiers compressé et les fichiers archives :

- `gzip` ou `bzip2` ou `xz` compressent un fichier,
- `gunzip` ou `bunzip2` ou `unxz` décompressent un fichier,
- `tar` créé une archive (option 'c') ou extrait les fichiers d'une archive (option 'x').
- `tar zcvf desktop.tar.gz  /Desktop`
- `gunzip desktop.tar.gz`
- `bzip2 desktop.tar`

Commandes pour travailler avec des fichiers (10/11)

- `uniq` pour supprimer (ou garder) les lignes en doublons.
  - sans option affiche une seule fois les lignes en doublons,
  - option '-u' pour n'afficher que les lignes uniques (n'affiche plus celles en doublons),
  - option '-d' pour n'afficher que les lignes dupliquées.
- `diff` affiche les différences entre deux fichiers
- `join` pour effectuer une jointure entre deux fichiers (il faut qu'il y ait un champ commun),
- `comm` pour afficher les lignes communes entre deux fichiers triés.

Commandes pour travailler avec des fichiers (11/11)

Pour aller plus loin vous pouvez aussi regarder le manuel ('man') des commandes suivantes :

- `tee` dupliquer la sortie,
- `fold` justifier le texte,
- `fmt` justifier sur une taille maximale (pas de découpe des mots),
- `tr` transliterer,
- `split` découper des fichiers en morceaux,
- `awk` travailler les fichiers en mode colonne (et non ligne),
- `rev` renverser le texte,
- `ln` créer un lien.

Commandes pour connaitre la place occupée / libre

- `du` pour 'disk usage' donne la taille des fichiers (la place occupée par un ensemble de fichier ou de dossier). Appliquée à un dossier la commande est récursive et donne la taille de tous les fichiers et sous-dossiers (les options intéressantes sont -s et -h),
- `df` pour 'disk free' indique la place occupée et libre des disques connectés à la machine,
- `quota` indique vos quotas : place occupée, place maximale éventuellement temps restant avant blocage (l'option -s permet une meilleure lecture).

Rappel : ne pas débrancher directement les disques et clefs USB, les «démonter» correctement avec l'aide du navigateur.

Utilisateur

Identification

Chaque utilisateur a un identifiant unique dans le système c'est l'UID (User IDentification). De même chaque groupe a son identifiant ou GID (Group IDentification).

- `id` donne les informations de l'utilisateur, son nom, les groupes auxquels il appartient (avec les identifiants numériques),
- `whoami` ne donne que mon identifiant,
- `who` ou `w` donne la liste des utilisateurs connectés à la machine (w donne plus d'informations),
- `groups` indique les groupes de l'utilisateur,
- `finger` Permet d'obtenir le nom des personnes (et pas seulement le login),
- Associée à un nom elle donne plus d'informations. Fichiers `.plan` et `.project`.

- `uname` avec l'option `-a` donne l'ensemble des informations système,
- Les fichiers `/proc/cpuinfo` et `/proc/meminfo` donnent des informations sur les CPU et la mémoire.

Processus

Un processus ? (1/2)

- Il s'agit d'un programme en cours d'exécution (ou pas),
- Chaque commande qui est lancée est un processus (ou un ensemble de processus),
- La ligne de commande elle même est un processus (tcsh chez nous),
- Le système lui même est géré par un ensemble de processus.

Un processus ? (2/2)

- Chaque processus a un numéro (PID - Process IDentification),
- Chaque processus a un processus père (PPID - Parent Process IDentification). Leur père à tous est numéro 1,
- Pour chaque processus le système attribue une priorité (modifiable à la baisse).

Il existe plusieurs commandes :

- `ps` Il existe des tas d'options et de variantes pour `ps`. Quelques options utiles sous Linux sont :
  - '-e' pour avoir tous les processus,
  - '-f' pour avoir l'ensemble des paramètres (full) avec notamment la ligne de commande entière du processus,
  - '-l' pour avoir une sortie complète (longue) et non tronquée,
  - 'f' (sans le -) donne l'arborescence des processus,
- `top` et `htop` sont des programmes interactifs où l'on voit l'évolution des processus en temps (presque) réel.

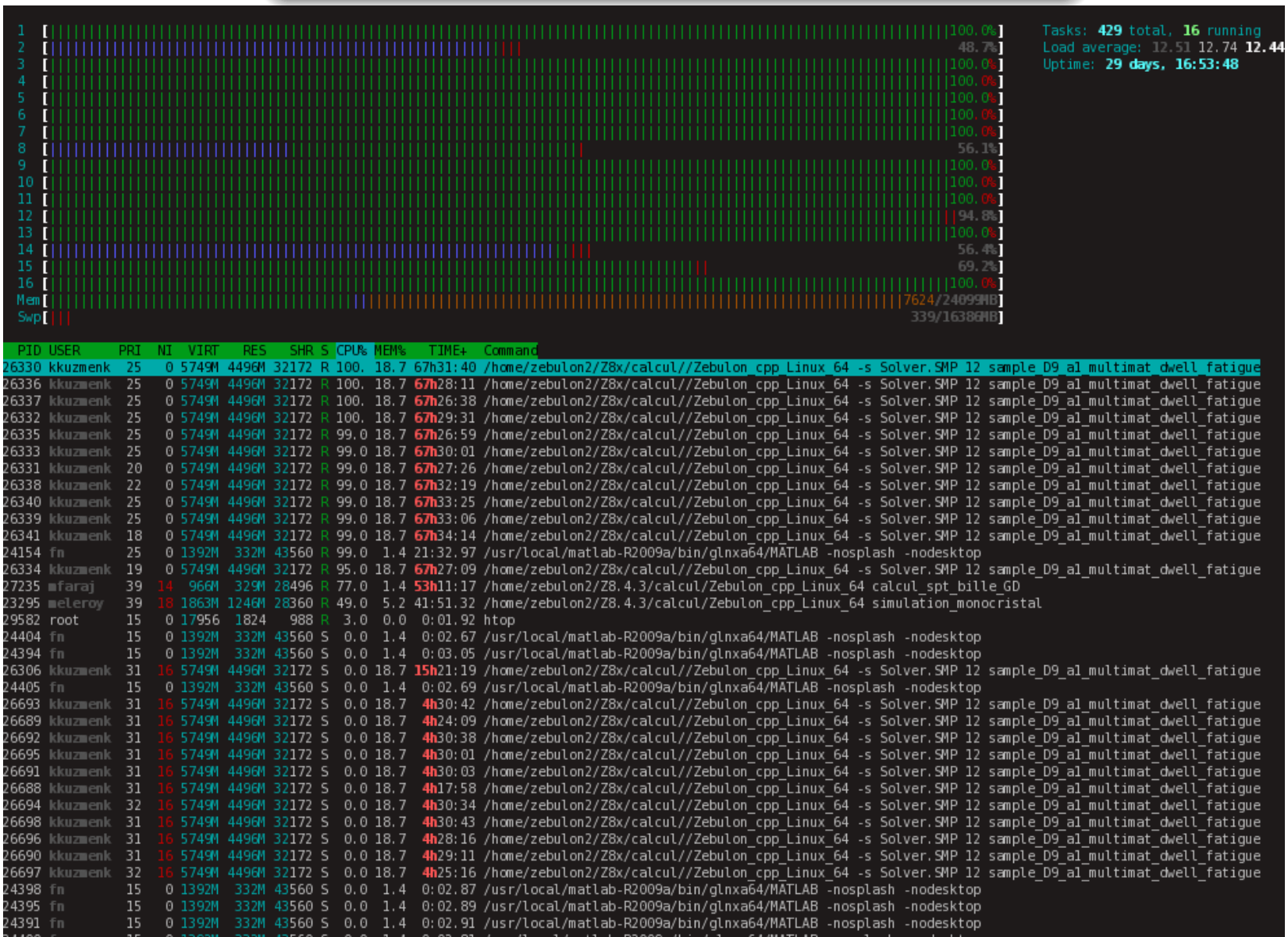
Processus

Voir les processus - un exemple de ps

```
5 S root      6530     1  0  78   0 - 20928 318845 Aug20 ?          0:00 ypbind
1 S root      6642     1  0  82   0 -  6581 -      Aug20 ?          0:00 ./hpiod
1 S root      6666     1  0  78   0 - 39341 -      Aug20 ?          0:00 python ./hpssd.py
5 S root      6684     1  0  75   0 - 15656 -      Aug20 ?          0:00 /usr/sbin/sshd
4 S root      7788  6684  0  77   0 - 23062 -      Aug20 ?          0:00  \_ sshd: lj [priv]
5 S lj        7790  7788  0  75   0 - 23297 -      Aug20 ?          6:03  |   \_ sshd: lj@pts/0
0 S lj        7791  7790  0  75   0 - 19603 -      Aug20 pts/0      0:00  |       \_ -tcsh
0 S lj        3228  7791  0  75   0 - 27909 -      Aug31 pts/0      1:23  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3230  3228  0  75   0 - 17459 rt_sig Aug31 pts/23     0:00  |           |   \_ -csh
0 S lj       13933  3230  0  78   0 - 15994 wait   Sep01 pts/23     0:00  |           |       \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zrun -smp 6 therm.inp
0 S lj       13948 13933 99  75   0 - 268909 184467 Sep01 pts/23  4167:27  |           |           \_ /home/zebulon2/Z8.4.4/calcul//Zebulon_cpp_Linux_64 -s Solver.SMP 6 therm
0 S lj        3650  7791  0  75   0 - 27909 -      Aug31 pts/0      0:00  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3652  3650  0  75   0 - 17492 rt_sig Aug31 pts/27     0:00  |           |   \_ -csh
0 S lj       15022  3652  0  75   0 -  9901 -      Sep01 pts/27     0:00  |           |       \_ gnuplot plot.gnu
0 S lj       15023 15022  0  75   0 - 19969 -      Sep01 pts/27     0:00  |           |           \_ gnuplot_x11
0 S lj        3701  7791  0  75   0 - 28902 -      Aug31 pts/0      1:28  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3703  3701  0  75   0 - 17462 rt_sig Aug31 pts/28     0:00  |           |   \_ -csh
0 S lj        4326  3703  0  78   0 - 15994 wait   Sep01 pts/28     0:00  |           |       \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zrun -smp 4 therm.inp
0 R lj        4341  4326 99  75   0 - 231023 184467 Sep01 pts/28  3329:03  |           |           \_ /home/zebulon2/Z8.4.4/calcul//Zebulon_cpp_Linux_64 -s Solver.SMP 4 therm
0 S lj        3730  7791  0  75   0 - 27909 -      Aug31 pts/0      0:02  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3732  3730  0  75   0 - 17461 -      Aug31 pts/29     0:00  |           |   \_ -csh
0 S lj        3759  7791  0  75   0 - 27909 -      Aug31 pts/0      0:00  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3761  3759  0  77   0 - 17461 -      Aug31 pts/30     0:00  |           |   \_ -csh
0 S lj        3788  7791  0  75   0 - 27909 -      Aug31 pts/0      0:00  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3790  3788  0  75   0 - 17460 -      Aug31 pts/31     0:00  |           |   \_ -csh
0 S lj        3817  7791  0  75   0 - 27909 -      Aug31 pts/0      0:00  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3819  3817  0  75   0 - 17460 rt_sig Aug31 pts/32     0:00  |           |   \_ -csh
0 S lj       18165  3819  0  75   0 -  9870 -      Sep01 pts/32     0:00  |           |       \_ gnuplot plot.gnu
0 S lj       18166 18165  0  75   0 - 20655 -      Sep01 pts/32     0:02  |           |           \_ gnuplot_x11
0 S lj        3863  7791  0  75   0 - 27909 -      Aug31 pts/0      0:00  |           \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        3865  3863  0  77   0 - 17460 -      Aug31 pts/33     0:00  |               \_ -csh
4 S root      7829  6684  0  76   0 - 23062 -      Aug20 ?          0:00  \_ sshd: lj [priv]
5 S lj        7831  7829  0  75   0 - 23158 -      Aug20 ?          3:42      \_ sshd: lj@pts/1
0 S lj        7832  7831  0  75   0 - 19597 rt_sig Aug20 pts/1      0:00          \_ -tcsh
0 S lj       26863  7832  0  76   0 - 15960 wait   Aug30 pts/1      0:00              \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zsopt TC12456.mast
0 S lj       26864 26863  0  78   0 - 15961 wait   Aug30 pts/1      0:00                  \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zmaster -s Zmaster.Interface wx TC12456.mast
0 S lj       26881 26864  0  75   0 - 208779 wait  Aug30 pts/1      0:06                      \_ /home/zebulon2/Z8.4.4/calcul/Zebulon_cpp_Linux_64 -G -s Zmaster.Interface wx TC12456.mast
0 S lj        4760 26881  0  77   0 -  2209 wait   Aug31 pts/1      0:00                          \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zrun therm.inp
0 R lj        4772  4760 99  85   0 - 186563 stext Aug31 pts/1    2792:45  |                           \_ /home/zebulon2/Z8.4.4/calcul//Zebulon_cpp_Linux_64 therm
4 S root      7886  6684  0  75   0 - 23062 -      Aug20 ?          0:00  \_ sshd: lj [priv]
5 S lj        7888  7886  0  75   0 - 23229 -      Aug20 ?          3:00      \_ sshd: lj@pts/2
0 S lj        7889  7888  0  75   0 - 19631 rt_sig Aug20 pts/2      0:00          \_ -tcsh
0 S lj        1112  7889  0  75   0 - 27915 -      Aug27 pts/2      0:00              \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        1114  1112  0  75   0 - 17459 rt_sig Aug27 pts/11     0:00                  \_ -csh
0 S lj       18163  1114  0  75   0 -  9870 -      Sep01 pts/11     0:00                      \_ gnuplot plot.gnu
0 S lj       18164 18163  0  75   0 - 21505 -      Sep01 pts/11     0:00                          \_ gnuplot_x11
0 S lj        1141  7889  0  75   0 - 27918 -      Aug27 pts/2      2:35              \_ xterm -sb -sl 5000 -font fixed -bg grey -cr magenta
0 S lj        1143  1141  0  76   0 - 17460 rt_sig Aug27 pts/21     0:00                  \_ -csh
0 S lj       26931  1143  0  75   0 - 15960 wait   Aug30 pts/21     0:00                      \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zsopt TC12456.mast
0 S lj       26932 26931  0  78   0 - 15961 wait   Aug30 pts/21     0:00                          \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zmaster -s Zmaster.Interface wx TC12456.mast
0 S lj       26949 26932  0  75   0 - 208276 wait  Aug30 pts/21     0:04                              \_ /home/zebulon2/Z8.4.4/calcul/Zebulon_cpp_Linux_64 -G -s Zmaster.Interface wx TC12456.mast
0 S lj        8511 26949  0  78   0 -  2209 wait   Aug31 pts/21     0:00                                  \_ /bin/sh /home/zebulon2/Z8.4.4/bin/Zrun therm.inp
0 R lj        8523  8511 99  85   0 - 185478 stext Aug31 pts/21   2740:02  |                                   \_ /home/zebulon2/Z8.4.4/calcul//Zebulon_cpp_Linux_64 therm
0 S lj       18161  7889  0  75   0 -  9870 -      Sep01 pts/2      0:00              \_ gnuplot plot.gnu
0 S lj       18162 18161  0  75   0 - 27628 -      Sep01 pts/2      0:00                  \_ gnuplot_x11
4 S root      7964  6684  0  76   0 - 23062 -      Aug20 ?          0:00  \_ sshd: lj [priv]
```

Processus

Voir les processus - un exemple de htop



Repérer la charge de la machine avec top ou htop

Par charge (load) on entend le taux d'occupation ou d'utilisation.

- load average : la moyenne à 1 minute, 5 minutes et 15 minutes de programmes fonctionnant à 100%,
- occupation mémoire : totale, libre, utilisée et en cache (+swap),
- occupation processeur avec top,
  - us = user (les programmes utilisateurs),
  - sys = système (le sytème),
  - wa = iowait (les entrées sorties en attente de traitement),
  - id = idle (quand le processeur ne fait rien).
- occupation processeur avec htop :
  - regarder les barres horizontales qui indiquent le taux d'occupation de chaque processeur (ou coeur).

- `watch` lance un processus (toutes les 2 secondes) et affiche sa sortie.

Suivi visuel de la modification de la liste des fichiers d'un dossier (création ou suppression de fichiers) :

- `watch -n 10 ls -ls mondossier`.

Lancer des processus en arrière plan

Jusqu'à présent on lance le processus et on attend qu'il se termine mais on peut lancer des processus longs en arrière plan :

- `&` - Mis après une commande : `xeyes &`,
- Pour vérifier les processus en cours utilisez la commande `jobs`,
- `Ctrl-Z` arrête le processus en avant plan,
- `bg` (pour background) met un processus arreté en arrière plan,
- `fg` (pour foreground) ramène un processus de l'arrière plan en avant plan,
- `nice` et `renice` permettent la modification de la priorité.

Tuer un ou des processus

Vous ne pouvez tuer que les processus qui vous appartiennent !

- `kill PID` tue le processus numéro PID,
- `pkill` comporte plus d'options permettant de filtrer les processus,
- `killall nomduprocessus` tue tous les processus nommés 'nomduprocessus'.

Connexion à d'autres machines

ssh (secure shell) le remplacant de rlogin (remote login)

Permet de se connecter sur une machine distante :

- `ssh nomdelamachine`,
- `-X` permet de transférer l'affichage graphique (X11),
- `hostname` donne le nom de la machine sur laquelle on est connecté.

scp : copier des fichiers d'une machine à l'autre

Permet de copier des fichiers d'une machine sur une autre.

- `scp src dst`,
- `-r` permet de copier récursivement un dossier,
- `-p` (pour preserve) préserve les permissions des fichiers lors de leur copie.

Exemple :

- `scp monfichier jacala:/home/tmpcalcul/toto`
- `scp -rp tp4:/home/tmpcalcul/toto hathi:/home/tmpcalcul/`

Unix avancé

Les signaux (1/2)

Les programmes peuvent envoyer et/ou recevoir des signaux. On utilise ce mécanisme sans le savoir : Lorsque l'on tape Ctrl-Z ou Ctrl-C le système envoi un signal au processus. Il s'agit de STOP (Ctrl-Z) et INT (Ctrl-C).

- `kill` est la commande qui permet d'envoyer des signaux (avec l'option `-s`). Par défaut elle envoie TERM,
- `kill -l` donne la liste des signaux que l'on peut envoyer,
- Les signaux STOP et KILL ne peuvent ni être ignorés ni interceptés par un processus.

Exemple avec la commande `dd if=/dev/zero of=/dev/null` et le signal USR1.

Les signaux (2/2)

Les signaux les plus usités sont :

- `STOP` qui stoppe l'exécution du processus,
- `CONT` qui reprends l'exécution d'un processus arrêté,
- `TERM` qui demande (poliment) au processus de se terminer,
- `KILL` qui termine le processus (il n'a pas le choix !),
- `USR1` qui est utilisé pour implémenter un signal spécifique à l'application.

Unix avancé

Les alias de commandes

La commande `alias` permet de definir un alias pour une commande (`unalias` pour faire l'inverse)
Quelques exemples :

- `alias` : sans argument donne la liste de tous les alias existants,
- `alias dir ls -ls` : taper dir dans la ligne de commande exécute `ls -ls`,
- `alias ll ls -ls | less` : la même chose mais avec un tube et l'exécution de `less`.

Les variables d'environnement

Il s'agit de variables que l'on peut fixer dans le shell. C'est très utile pour modifier un comportement seulement dans un shell donné (et de manière temporaire).

- par convention notées en majuscules.
- Les variables d'environnement peuvent être utilisées avec les commandes ou les alias de commandes.

Les variables d'environnement

- `setenv` : liste toutes les variables d'environnement (chemins, options de compilation, ...),
- `setenv TPPATH /home/users/toto/TP` : affecte `/home/users/toto/TP` à la variable d'environnement `TPPATH`,
- `$` placé devant le nom d'une variable d'environnement pour y faire référence : `echo $TPPATH`,
- `unsetenv TPPATH` supprime la variable d'environnement `TPPATH`.

Unix avancé

La commande which

Pour connaitre ou le shell localise une commande et laquelle il utilise réellement quand on sous-entend la position de l'exécutable qu'on tente de lancer :

- `which macommande`
- affiche le chemin ou se trouve la commande ou le fichier executable
- tres pratique pour verifier que l'on appelle la bonne commande lorsqu'on a modifié les variables d'environnement PATH d'un shell

La commande module

... ou la manipulation des variables d'environnement sans y penser ...

- `module list` Liste les modules chargés
- `module avail` Liste les modules disponibles
- `module load unmodule` Charge un module
- `module unload unmodule` Décharge un module

Unix avancé

La substition de commande

On peut inclure une commande dans une autre en utilisant les côtes obliques (AltGr-7) : ‘macommande‘.
Exemples :

- `echo -n ‘date -R‘`
- ```
  foreach i ( ‘ls‘ )
    file $i
    echo $i
  end
  ```
- `setenv PWD ‘pwd‘`
  ```

L'éditeur de fichier texte vim

Deux modes :

- mode d'édition : tapez `i` ou `a` pour entrer en mode édition. La touche «Esc»' (ou «Echap») sort du mode d'édition,
- mode normal (ou de déplacement) : dans ce mode taper `:help` pour avoir l'aide.

En mode normal (ou de déplacement) :

- `dd` pour supprimer une ligne,
- `x` pour supprimer un caractère,
- `u` pour undo,
- `p` pour copier la dernière ligne effacée.
- `G` va à la dernière ligne du fichier
- `gg` va à la première ligne du fichier
- `1315` va à la ligne 1315

- `:` pour indiquer qu'une commande suit,
- commande `q` pour quitter (ajouter ! pour forcer),
- commande `w` pour sauvegarder (ajouter ! pour forcer).

On peut chainer les commandes et faire `:wq` pour sauvegarder et quitter.

links :

- `links http://wikipedia.fr`
- Les flèches du haut et du bas pour se balader dans la page,
- Les flèches droites et gauche pour aller et revenir dans l'historique,
- La touche 'd' pour télécharger, la touche 'q' pour quitter.

- La première ligne indique le nom de l'interpréteur `#!/bin/tcsh` pour ce qui nous concerne.

On trouve aussi (et des tas d'autres) :

- `#!/usr/bin/perl`
- `#!/usr/bin/env python`
- `#!/bin/bash` ...

Paramètres des scripts (en tcsh)

- `$0` : le nom de du script
- `$1, $2, $3`... les paramètres 1, 2, 3... du script ou de la fonction
- `$*` : tous les paramètres
- `$?` : le code de retour de la dernière commande
- `$$` : le numéro (PPID) du processus parent
- `$!` : le numéro (PID) du dernier processus mis en tâche de fond
- `$_` : la dernière ligne de commande executée

Unix avancé

Structures de contrôle (en tcsh)

- `if (expression) then...else...endif`
- `switch (expression) case xy:...breaksw...case yx:...breaksw...endsw`
- `while (expression)...end`
- `foreach x (list)...end`

Exemples de structures de contrôle (1/2)

- ```
  foreach x (fichier1 fichier2)
    echo "$x :"
    file $x
  end
  ```
- ```
  switch ($argv[1])
  case -h:
    echo "Usage : exemple options args"
    breaksw
  case -t:
  case -T:
    file monfichier
    breaksw
  endsw
  ```

Exemples de structures de contrôle (2/2)

- ```
  if (-e readme && -r readme) then
  echo "Le fichier readme existe et est lisible"
    cat readme
  else if (-e readme)
    echo "Le fichier existe mais n'est pas lisible"
  else
    echo "Le fichier n'existe pas"
  endif
  ```
- ```
  set n=1
  while ($n < 301)
    echo $n
    @ n = ($n + 1)
    @ n++
  end
  ```

- Opérateurs logiques || &&
- Opérateurs de comparaison de chaines == != =~ !~
- Opérateurs de comparaison de nombres <= >= < >
- Opérateurs arithmétiques | & ^ + - * / % ! ~ « » ( )

Opérateurs de tests sur les fichiers

Quelques opérateurs utiles :

- `-r` Accès en lecture
- `-w` Accès en écriture
- `-x` Exécutable
- `-e` Existence
- `-z` Taille nulle
- `-s` Taille non nulle
- `-f` Fichier texte
- `-d` Dossier
- `-l` Lien symbolique

Merci !

Merci !