Introduction à UNIX
commandes / shell / outils

Un peu d 'histoire

- AT&T 's Bell Labs 1969-1971 : Première version d 'UNIX
- Interactive Systems Corporation 1977 : Première distribution commerciale
- Berkeley Software Distribution 1977 : Système BSD (C-shell, TCP/IP)
- 1978 : Version 7 (Bourne-shell)
- Sun Microsystems 1983 : Station de travail UNIX, System V (NFS)

- « Standard »:
  - System V
  - BSD

- Versions majeurs d 'Unix : BSD, HP-UX, SunOS et Solaris, IRIX, Digital Unix et Ultrix, AIX, NeXTSTEP et OpenStep, SCO Unix, Linux
- Émulateur : mingwin, cygwin

Spécificité

- OS écrit en langage C :
  - API
  - applications portables
  - système ouvert (très peu de code en assembleur), X/OPEN Portability Guide (XPG) « standard »
- Multi-tache
- Multi-utilisateur
- Interface utilisateur, X-Windows
- InterProcess Communication

Architecture

- Système :
  - Noyau ou Kernel
  - Shell, interpéteur de commande (Korn Shell, Bourne Shell, C Shell)
  - Serveur graphique Xwindows (X11R6)
- Utilisateur :
  - Librairies
  - compilateurs (C, C++…)
  - Outils de développement
  - programmes utilisateur
- Stockage :
  - Espace physique, disques
  - Espace logique, Système de Fichiers
- Communications, TCP/IP, Ethernet, TokenRing

Premiers pas

- Lancer cygwin :
  - Dans une invite de commande DOS
  - dans un terminal X11 : rxvt, xterm…
  - Répertoire de travail au lancement de cygwin :

    /home/<utilisateur>
  - shell, interpréteur de commande par défaut bash Bourne-again-shell
  - fichiers .kekechose (dot files) ; .profile, .bashrc, .Xdefaults…

Environnement sous bash

- Environnement de travail ⟷ Variables d'environnement ⟷ Fichiers d'initialisation
- Initialisation :
  - /etc/bash.bashrc, /etc/profile
  - /home/…/.bash_profile, /home/…/.bashrc
- Afficher la valeur d'une variable :

  $ echo $NOM_DE_VARIABLE
- Quel shell ?

  $ ps

```
    PID    PPID    PGID     WINPID   TTY  UID    STIME COMMAND
   2524    2392    2524       4068  con  400 12:45:28 /usr/bin/ps
```

  $ echo $SHELL

```
  /bin/bash
```

Principales Variables d'Environnements

- LOGNAME — nom de l 'utilisateur ou login
- PATH — chemin de recherche des commandes
- HOME — répertoire de travail par défaut
- SHELL — shell utilisé, par défaut bash
- TERM — type de terminal (rxvt, xterm, kterm…)
- MANPATH — chemin de recherche du manual en ligne
- DISPLAY — IP du poste d 'affichage (ex : 193.1.1.167:0.0)
- PRINTER — nom de l 'imprimante par défaut
- PAGER — programme d 'affichage du contenu d'un fichier (less, more)

Personnaliser l 'Environnement

- Afficher la valeur des variables définies :

  $ set

- Changer temporairement la valeur d 'une variable

  $ export VAR1=valeur

  - exemple :

    $ export EDITOR=vi

- Changement permanent de la valeur d 'une variable

  - éditer le fichier .bash_profile et insérer par exp. la ligne :

    $ export PRINTER=mon_imprimante

  - sauvegarder le .profile
  - effet immédiat des changement :

    $ . .bash_profile

    $ source .bash_profile

Les commandes Unix - localiser

- Comment le shell trouve les commandes ?
  - Dans les répertoires énumérés dans la variable d'environnement $PATH, dans l'ordre
  - exemple:

    /bin:/usr/bin:/usr/local/bin:$HOME/bin:.

    le point à la fin de la ligne désigne le répertoire courant
- Quelle commande j 'utilise ?

  $ which ls

  `/usr/bin/ls`

  $ type ma_cmd

  `ma_cmd est /usr/rbb/bin/ma_cmd`

Les commandes Unix - options, arguments

commande options argument(s) :

- exemple :

  $ cp file1 file2 — copie file1 dans file2

- syntaxe : options disponibles, nombre d 'arguments autorisés
- options :

  $ wc -w file1 — compte les lignes contenues dans file1

  $ ls -lrt ou ls -l -t -r — liste le contenu du répertoire courant trié dans l 'ordre chronologique

- arguments :

  $ date — pas d 'argument fourni

  $ cd rep — change de répertoire, nouveau répertoire de travail rep

Aide en ligne

- Commande man : man [section] nom_de_commande
- exemple :

$ man cp	affiche la description de cp dans le manuel
$ man 1 ls	la 1ère section du manuel traite des commandes usuelles
$ man -k 'copy files'	l'option -k affiche les commandes dont les pages du manuel contiennent la chaîne caractères
$ apropos mail	similaire à man -k
$ man -t mv	imprime le manuel de mv, on peut aussi utiliser : man mv | lpr

Les commandes Unix - caractères

- Unix est case-sensitive : Algo.cc != algo.cc !=ALGO.cc
- Metacaratères, caractères ayant une signification spéciale pour le shell :

  ; & ( ) | ^ < > ~ % { } $ # ' " \ @@ * ? [ ] - !
- Libérer les metacaractères, les faire préceder par \

  $ ls \#print.cc\# \ annule la signification du metacaractère #

  #print#
- Le joker * pour une chaîne de caractères quelconque

  $ more doc*.txt affiche le contenu de tous les fichiers commençant par doc avec l 'extension txt, y compris doc.txt
- Le joker ? pour 0 ou 1 apparition de tout caractère

  $ cat chap?.tex | wc -l affiche le nombre total de lignes des fichiers chap1.tex chape.tex chapV.tex...

Les commandes Unix - manipuler

- Plusieurs commandes en une seule ligne :

  $ cd newdir ; ls -l change de répertoire, puis en affiche le contenu

- "piper" des commandes, connecter plusieurs commandes :

  $ ls -l | wc -l compte le nombre de fichiers dans le
  342 répertoire courant

- Commande en serpent

  $ vi mon_fichier_avec un_nom_presque_aussi_long\
  > _que_la_muraille_de_chine.asie

Les commandes Unix - pratique

- alias, Raccourcir les commandes pour plus d 'ergonomie :
  il est préférable que les alias se trouve dans le fichier .bashrc que dans le .profile

  alias xterm='xterm -ls -fn r14'

  unalias xterm annule l 'alias xterm

- Complétion de nom de fichier :

  $ ls .bash <ESC><ESC> affiche tous les fichiers commençant par .bash

  .bash_history .bash_profile .bashrc

- Edition de la ligne de commande

En plus des flèches du clavier

- Ctrl-P Ctrl-N Ctrl-A Ctrl-E
- Ctrl-B Ctrl-F Ctrl-D

Redirection d 'entrée - sortie

- Redirection d 'Input STDIN (clavier par défaut)

  $ grep "COPY PBM1234" < cpy.liste

- Redirection d 'Output STDOUT (écran par défaut)

  $ ls *.cbl > cbl.liste si cbl.liste existe déjà, il sera écrasé sauf si ajouté la protection : set -o noclobber est au .profile

- Redirection d 'Erreur STDERR (écran par défaut)

  $ rm *.int 2> err.liste

- Redirection d 'Output avec concaténation

  $ cat exemple1.cc > tout.cc

  $ cat exemple2.cc >> tout.cc

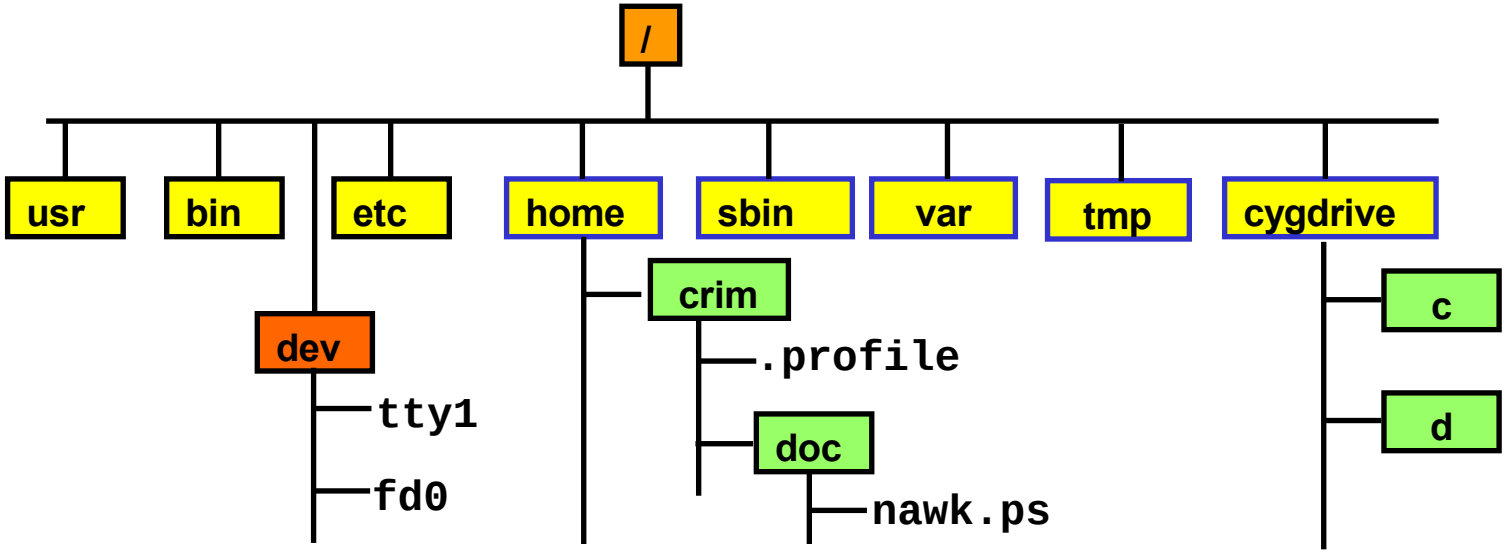
- Redirection d 'Output et d 'Erreur dans le même fichier

  $ find . -name *.cbl >> cbl.liste 2>&1 &1 = cbl.liste

  $ find . -name *.cbl 2>&1 > cbl.liste &1 = STDOUT

Système de Fichiers

- Types d 'objets d 'un système de fichiers :
  - Fichiers ordinaires : Données texte (ASCII) ou code (BINAIRE)
  - Répertoire : ensemble de fichiers, répertoires, liens
  - Fichiers spéciaux : unités matérielles ou logiques (ex : /dev/rmt0)
- Structure hiérarchique :



Répertoires - visite guidée

- Où suis-je ?

  $ pwd ou echo $PWD affiche le répertoire courrant

  ```
  /home/crim/bin
  ```

- Changer de répertoire avec la cmd cd
  - Chemin absolu

    $ cd /usr/local/bin chemin commençant par /
  - Chemin relatif : à partir du répertoire courant

    $ pwd

    ```
    /home/crim
    ```

    $ cd bin cela suppose que bin existe

    ```
    /home/crim/bin
    ```

    $ cd .. remonte d 'un cran dans la hiérarchie des rép.

    ```
    /home/crim
    ```

- Créer un répertoire

  $ mkdir /users/newapp attention aux perm. d'accès

Répertoires - visite guidée

- Le saviez-vous ?

  $ cd ~ — ~ : raccourci pour le répertoire personnel

  ```
  /home/crim
  ```

  $ cd ~crim — ~crim : raccourci pour le rép. Pers. du user crim

  ```
  /home/crim
  ```

- Lister le contenu d 'un répertoire :

  $ ls /srv/www

  ```
  cgi-bin  htdocs
  ```

  $ ls -l /etc/profile

  ```
  -rwxr-x---  1 faqman Aucun 6528 Oct 18 13:04 /etc/profile
  ```

  $ ls -F cur — / identifie les rép. * les exec.

  ```
  Feb2006/ gothere*  dirtmp/ TODO.html
  ```

  $ ls -a ~ — -a affiche fichiers cachés : .kekchose

  ```
  .  .. .rhosts .profile essai.c
  ```

- Copier un répertoire et ses sou-rép. (option -r) :

  $ cp -r ~/ dir1 ./dir2 — si dir2 existe dir1 sera un sous-rép. de dir2

Répertoires

- Renommer ou déplacer un répertoire dans un filesystem :

  $ mv dir1 /home/crim/dir2 même remarque que pour copie

- Supprimer un répertoire :

  $ rmdir testrep le répertoire à supprimer doit être vide

  $ rm -r testrep supprime tout le contenu du répertoire

  $ rm -ir testrep

```
rm: Voulez-vous sélecter des fichiers dans le répertoire testrep?
      Entrez o pour oui. o
rm: Voulez-vous supprimer testrep/file1? o
rm: Voulez-vous supprimer testrep? o
```

- Comparer deux répertoire :

  $ diff rep1 rep2 le répertoire à supprimer doit être vide

```
diff rep1/file1 rep2/file1
2d1
< ssss
Uniquement  dans  rep2: file2
```

Fichiers

- Répertoire = sommaire de fichiers et sous-répertoires
  - Nom du fichier + inode : pointeur unique par fichier
- Créer un fichier avec :

  $ cat > file1

  a que coucou entrée au clavier

  Ctrl-D

  $ echo utiliser un éditeur pour les fichiers conséquents > conseil

  $ touch fichier rafraîchi la date du fichier s 'il existe

  $ vi mon_fichier éditeur de texte, la méthode standard
- Nom de fichier flottant

  $ ls CHAP0[1-2]B[0-2]*

  ```
  CHAP01B0.dvi CHAP02B0.dvi CHAP02B02.tex CHAP01B0.tex CHAP01B2.tex
  ```

  $ ls [pP]art[0-9]

  ```
  part1 part2 Part0
  ```
- Renomer un fichier

  $ mv liste_nom.txt liste_nom.old option -i

Fichiers - Manipuler

- Copier un fichier

  $ cp test save

  $ cp ~tal/TAL8201.txt .

  $ cp editions_?.sh /srv/scripts/editions

- Supprimer un fichier

  $ rm liste_nom.old option -i

- Afficher le contenu d 'un fichier

  $ cat test

  ```
  fichier se nomme test et ne contient que cette ligne :-)
  ```

  - more, less <barre d 'espace> forward, <b> backward
  - head, tail option -n nombre entier de ligne à afficher du début du fichier head, fin du fichier tail

- Type d 'un fichier

  $ file /usr/sbin/makewhatis

  ```
  /usr/sbin/makewhatis: Bourne-Again shell script text executable
  ```

Fichiers - Contenu

- Comparaison de fichier :

$ diff fichier1 fichier2 sdiff pour fichier de texte uniquement

```
2a3,4
> Jean JRS@pollux.ucs.co
> Jim  jim@frolix8
```

- Tri de fichier

$ sort fichier le séparateur de champs est par défaut espaces

$ sort -t: fichier : séparateur

$ sort -o res fichier output fichier res

$ sort -o res fichier output fichier res

$ sort +2 fichier clé de tri à partir du 3ème champ

$ sort -5 fichier clé de tri s 'arrête à la fin du 5ème champ

$ sort +1 -3 fichier clé de tri du 2ème au 3ème champ

$ sort -m file1 file2 >res -m fusionne le résultat du tri dans res

$ sort -u file élimine les doublons, voir aussi uniq

$ sort -d file tri selon le dictionnaire

Fichier - Localiser - explorer

- ## Chercher des fichiers

  $ find chemin -name fichier -print

  $ find . -name "cme000[1-9]*.*" -print

```
./cme0001.sh
./cme0004.exe.rf
```

- ## Chercher le contenue de fichiers

  $ grep chaîne_de_caractère fichier

  $ grep PATH .profile

```
export PATH=$PATH:/usr/local/bin:${HOME}/bin
```

  $ ls -l | grep '^d.......x'

```
drwxrwsr-x   2 crim      DVPT          512 14 avr 11:06 bin
drwxrwsr-x   2 crim      DVPT          512 19 avr 18:24 command
```

  $ ps -fela | grep crim |grep -v grep

```
crim     792    3376 con  10:12:19 /usr/bin/ps
crim        3328    3376 con  10:12:19 /usr/bin/bash
```

Fichiers - Plus

- Extraire des données

  $ cut -f 1,5 -d: /etc/passwd

  -d permet spécifier un délimiteur par défaut espace ou tabulation

```
su:User with special privileges

daemon:
     bin:
    sys:
    adm:System Administrator
    pierre:Pierre Harper
```

  -f champ

  $ cut -c18-30 exple.txt

  -c colonne

- Imprimer
  - lpr –pHP304 search.res

  si les varibles d 'environnement PRINTER ou LPDEST sont positionnée option -p inutile

Fichiers - Transfert

- La commande ftp

$ ftp seveur_distant ouvrir la connexion open seveur_distant

ftp> lcd network — changer de répertoire local

```
Local directory now /users/rbb/network
```

ftp> ! — décrocher vers une session shell

$ date

```
jeu 20 avr 15:45:05 DFT 2000
exit
```

ftp> !vi .profile — lancer une commande shell interactive

ftp> ls — lister le contenu du répertoire distant

```
200 Exécution de la commande PORT terminée.
150 Etablissement de la connexion de données pour ..
150 Opening ASCII mode data connection for file list.
public
reports
226 Transfert terminé.
```

ftp> cd /public — changer de répertoire distant

```
250 Exécution de la commande CWD terminée.
```

ftp> ? — aide help dir

Fichiers - Transfert

ftp> bin — mode de transfert binaire

```
200 Type défini à I.
```

ftp> ascii — mode de transfert texte

```
200 Type défini à A ; format défini à N.
```

ftp> put unfichier — transfert unfichier sur le serveur distant

```
200 Exécution de la commande PORT terminée.
150 Etablissement de la connexion de données pour unfichier.
226 Transfert terminé.
120070 bytes sent in 0,1637 seconds (716,2 Kbytes/s)
local: gunzip éloigné: gunzip
```

ftp> get unfichier — transfert unfichier sur le poste local

voir aussi prompt off, mget et mput

Fichiers - Archivage

- Création d 'archive tar

  $ tar cvf mesdocs.tar file_1 … file_n

  $ gtar cvf dev.tar ./dev

  $ tar cvf /dev/rmt0 /srv sauvegarde sur bande magnétique

- Rajout

  $ tar rvf dev.tar ./rapport.tex option r interdite avec une bande

- Visualisation du contenu

  $ tar tvf mesdocs.tar

- Extraction

  $ tar xvf dev.tar

  $ tar xvf /dev/rmt0

- Compression de fichier

  $ gzip mesdocs.tar

Fichiers - Archivage

- Décompresser de fichier

  $ gunzip mesdocs.tar.gz

- Manipulation d 'une archive compressée

  $ gtar ztvf dev.tar.Z visualisation

  $ gtar zxvf mesdocs.tar.gz extraction

Liens symboliques

- La commande ln

  $ ln source cible la source et la cible sont des répertoire ou des fichiers

- Lier des fichiers entre système de fichiers différents

  $ ln -s /usr/local/bin/tar ~crim/gtar lien symbolique

- Lier des répertoires, utilisateurs soft-link uniquement

  $ ln -s /home/crim/test1 test0 ! chemin absolu pour la source !

  $ ln -s /home/crim/test1 ~/nsmail/sent ~/rep/test3

  crée les liens test1 et sent dans le sous-répertoire test3

- Supprimer des répertopire créés avec ln

  $ rm ~/rep/test3/test1 supprime uniquement le lien

Permissions d 'accès

- Afficher les permissions

  $ ls -l doit.sh

```
-rwxr-xr-x  1       bbb        DVRT            3649       Feb 22 15:51 doit.sh
```

- Kezako ?
  - r lecture
  - w écriture
  - x exécution, accéder au répetoire

-rwxr-xr-x

u g o

  - u utilisateur
  - g groupe
  - o autres utilisateurs

Permissions d 'accès

- Permission par défaut : 666 fichiers, 777 répertoires
- Permissions en chiffres :

---	--x	-w-	-wx	r--	r-x	rw-	rwx	
0	1	2	3	4	5	6	7	chmod
7	6	5	4	3	2	1	0	umask

  - exemples

	u	g	o
› chmod 640 nom.dat	rw-	r--	---
› chmod 745 nom.dat	rwx	r-x	r--
› chmod g+x dothis.sh	rwx	rwx	r--
› chmod o-w dothat.sh	rwx	rw-	r-x
› chmod 640 nom.dat	rw-	r--	---
› umask 002	rw-	rw-	r--

- chgrp
  - › chgrp staff ESSAI.sh l 'utilisateur doit être membre de staff

Gérer les travaux et les Process

- La commande ps affiche les travaux et process en cours

$ ps -fela

```
UID     PID    PPID TTY      STIME COMMAND
 400    3908       1 con   11:20:56 /usr/X11R6/bin/XWin
 400    2196       1   0   11:36:53 /usr/bin/sh
crim     412       1 con   14:03:18 /usr/bin/bash
crim    3664       1   1   15:08:31 /usr/bin/emacs
crim    3444     412 con   15:48:01 /usr/bin/xterm
crim    3852    3444   2   15:48:01 /usr/bin/bash
crim    4000    3852   2   16:36:31 /usr/bin/emacs
crim    2264    3852   2   17:03:39 /usr/bin/ps
```

$ ps -u crim -f affiche les process appartenant à crim

```
UID     PID    PPID TTY      STIME COMMAND
crim     412       1 con   14:03:18 /usr/bin/bash
crim    3664       1   1   15:08:31 /usr/bin/emacs
crim    3444     412 con   15:48:01 /usr/bin/xterm
crim    3852    3444   2   15:48:01 /usr/bin/bash
```

- Relation Parent - Enfant
  - process **3444** est l 'enfant du process **412** shell de connexion

Gérer les travaux et les Process

- pour arrêter un process TP : Ctr-C

  $ programme_de_la_mort

```
vive UNIX à bas Windog
^C
```

- pour suspendre un process TP : Ctr-Z

  $ le_meme_programme_de_la_mort

```
vive UNIX à bas Windog
^Z
[1]+   Suspendu (SIGSTP)  le_meme_programme_de_la_mort
```

- Mettre un process en arrière plan

  $ bg

```
[1]              le_meme_programme_de_la_mort &
```

- Quels process sont en background ?

  $ Jobs l 'option -l affiche le numéro du process PID

```
[3]  + En cours d'exécution        le_meme_programme_de_la_mort
[2]  - En cours d'exécution        vi file1
[1]    En cours d'exécution        ftp ftp.bidule.com
```

Gérer les travaux et les Process

- Comment le tuer ? La commande qui tue : kill

  $ kill %1

  ```
  [1]  Terminé                    ftp ftp.bidule.com
  ```

- Comment remettre un job en avant-plant foreground

  $ fg %3

  ```
  le_meme_programme_de_la_mort
  ```

- Comment tuer un process ? La commande qui tue : kill 2ème

  kill [-signal] PID (Process Identifier) utiliser ps pour trouver le PID

  - exemple :

  $ kill 3960

  $ kill -15 3860 consulter le manuel pour signal

Editeur vi - les bases

- Buffer /tmp
  - vi .profile
  - ls -l /var/tmp |grep rbb

```
-rwxr-----   1 rbb      DVPT      163840 28 avr 17:03 Ex83098
```

- Restaurer
  - vi -r fichier
- Modus Operandi :
  - Visualisation — par défaut à l 'ouverture du fichier
  - Insertion ----> commandes :

a A i I o O

<Echap> — pour revenir au mode commande

- Décrocher vers le shell :

: !cmd — exécuter une cmd depuis vi

: sh — ouvrir une session shell

: r!cmd — insérer le resultat d 'une cmd

vi - options

- Options
  - : set option — dans le .exrc ou la variable EXINIT
  - : set all
  - : set
  - : set number
  - : set sw=8
  - : set autoindent
- Sauvegarder un fichier
  - : w — sauvegarde sans fermer le fichier
  - : wq — sauvegarde et ferme la session vi
  - ZZ — idem
  - : q! — ferme la session vi sans sauvegarde
  - : w autre_fichier — sauvegarde d ans un autre fichier

vi se ballader

Fleches : gauche, droite, haut et bas

w : mot suivant

wn : n mots suivants

b : mot précédent

bn : n mots précédents

e : fin du mot actuel

0, ^ : début de ligne

$: fin de ligne

G : fin du fichier

nG : allez à la ligne n

+ : début de ligne suivante

- : fin de ligne précédente

Ctrl-F (Ctrl-D) : une (1/2) page en avant

Ctrl-B (Ctrl-U) : une (1/2) page en arrière

vi plus

H : début de l 'écran — 3H troisième ligne à partir du haut

M : milieu de l 'écran

L : fin de l 'écran — 3L

- **Copier - coller**

  y — se combine avec les commande de déplacement

  yy — copie ligne

  p — coller

  J — joindre la ligne courante et la suivante

- **Tagger - marquer**

  mt — marquer la ligne, t le nom du tag

  mk — marquer la ligne, k le nom du tag

  : 't,'k d — détruit les lignes entre t et k

- **Inserer des fichiers**

  : r zefile — insère zefile après la ligne du curseur

  : 23:r fichier — insère zefile à ligne 24

vi plus+

- Couper

  x

  5dw

  dfleche

  d$

  dd

  ndd

- Répéter dernière frappe

  .

- Annuler une frappe

  u

- Rechercher une chaîne de caractère

  /chaine ou ?chaine

  n ou N

vi

- ## Remplacer du texte

r<car>	remplace le caractère courant par car
R<chaîne><esc>	substitue chaîne à la chaîne courante de même longueur
cw <chaîne><esc>	substitue chaîne au mot courant se combine avec la navigation
cw <chaîne><esc>	remplace le mot courant par chaîne
s<chaîne><esc>	remplace le caractère courant par chaîne
S<chaîne><esc>	substitue chaîne à la ligne

: s/ceci/cela	premier ceci rencontré sur la ligne
: s/truc/muche/g	tous truc de la ligne
: g/mâle/s/mal/gc	avec confirmation, dans tout le fichier
: 1,10 s/truc/muche/g	lignes 1 à 10 seulement

- ## Rechercher

/chaine ?chaine n N

Programmation shell - les bases

- Exemple

```
#!/usr/bin/ksh
# ce script affiche la date,
# l 'heure, le nom d 'utilisateur et
# le répertoire de travail.
echo "Date :"          #ou echo "Date :\\n `date`"
date
echo
echo "Utilisateur : `whoami` \\n"
echo "répertoire : \\c"
pwd
```

Programmation shell - arguments

- Exécution
  - sh mon_script arg1 arg2
  - mon_script arg1 arg2 — attention aux droits d 'exécution
  - . ./autre_script
- Arguments

$1,…,$9	neuf arguments en entrée (shift !)
$0	nom de la commande (script exécuté)
$#	nombre d 'arguments
$?	code retour (0 : succès, échec sinon)
$$	pocess id shell
$!	pocess id dernière commande en background
$-	options passées aux shell
$*	chaîne de caractères de tous les arguments
$@@	idem, exceptés ceux entre quotes

Programmation shell - arguments

- la commande shift, exemple shift_demo :

```
#!/bin/ksh
echo "arg1=$1 arg2=$2 arg3=$3"
shift
echo "arg1=$1 arg2=$2 arg3=$3"
shift
echo "arg1=$1 arg2=$2 arg3=$3"
shift
echo "arg1=$1 arg2=$2 arg3=$3 »
```

› shift_demo un deux trois quatre cinq six sept

```
arg1=un arg2=deux arg3=trois
arg1=deux arg2=trois arg3=quatre
arg1=trois arg2=quatre arg3=cinq
arg1=quatre arg2=cinq arg3=six
arg1=cinq arg2=six arg3=sept
```

Programmation shell - variables

- Evaluation de variable

$var	signifie la valeur de var
${var}	idem, var est protégée
${var-chose}	valeur de var si var définie sinon chose
${var=thing}	valeur de var si var définie sinon chose, var prend comme valeur chose
${var?message}	si définie $var sinon affiche message et sort du shell, si message vide affiche un message standard
${var+thing}	thing si $var définie, sinon rien

Programmation shell - fonctions natives

- Lecture d 'input

```
echo "votre prénom et nom svp:"
read prenom nom
echo "Bonjour $prenom $nom"
```

- Arithmétique

```
b=`expr 22 -33 / 3  \* 6`
echo $b
-44
a=alpha.c
expr $a : '\(.*\).c'
alpha
```

- Evaluation

```
cmd=dir
eval $cmd
```

Programmation shell - fonctions natives

- Exécution de commande sans créer de nouveau process

exec cut -c6-72 PBI7501.cbl

- Arrêter l 'exécution d 'un shell

exit 1 1 désigne le code retour

- Capturer les signaux du système

0	shell exit	1	arrêt
2	interruption (Ctrl-C)	3	quit (pgm core)
9	kill (non-capturable)	15	suppression

  - Exemple

```
trap 'echo `pwd` >>$HOME/errdir' 2 3 15
for i in /bin /usr/bin /usr/any/bin
do
cd $i
…cmd dans le répertoire $i …
done
```

Programmation shell - test

- ## Commande if

```
if condition
then
    cmds    (si condition vraie)
else
    cmds    (si condition fausse)
fi
```

```
if …
then
   …
else if
        …
     fi
fi
```

```
if ...
then
   ...
elif
   ...
fi
```

- ## Commande test

test condition ou [condition] attention aux espaces !!

```
if [ -f $1 -o -d $1 ]
then
    echo "$1 est un fichier régulier ou un répertoire"
else
     echo "$1 n'est ni un fichier régulier ni un répertoire"
fi
```

Programmation shell - opérateur

- Opérateurs binaires
  - &&

```
cmd1 && cmd2
```

```
cmd1
if [ $? -eq 0 ]
then
    cmd2
fi
```

  - ||

```
cmd1 || cmd2
```

```
cmd1
if [ $? -ne 0 ]
then
    cmd2
fi
```

  - ET logique : -a
  - OU logique : -o
  - Negation logique : !

Programmation shell - opérateurs

- Opérateurs de comparaison
  - égalité de chaîne de caractères : =
  - inégalité de chaîne de caractères : !=
  - égalité de nombre : -eq
  - inégalité de nombre : -ne

  ≥ : -ge

  ≤ : -le

  < : -lt

  > : -gt
- Test de fichier et de chaîne de caractères
  - fichier ordinaire : -f nom_fichier
  - fichier non vide : -s nom_fichier
  - répertoire : -d nom
  - chaîne de caractères vide : -z chaine
  - chaîne de caractères non vide : -n chaine

Programmation shell - flux

- case

```
case mot in
    choix1) cmd(s)
            ;;
    choix2) cmd(s)
            ;;
    *) cmd(s)
            ;;
esac
```

Programmation shell - flux

- exemple

```
set `date`
case $1 in
    sam) echo "piscine"
         ;;
    dim) echo "pingpong"
         ;;
    *)   echo "nada"
         ;;
esac
```

Programmation shell - boucle

- for

```
for var in list-de-mots
do
    cmd(s)
done
```

```
#!/bin/ksh
# copie fichier du répertoire courant au sous-répertoire "old"
for i in *
do
    echo $i:
    cp -p $i old/$i
    echo
done
```

Programmation shell - boucle

- while

```
while command-liste1
do
    command-liste2
done
```

  - boucle infinie

```
while [ 1 ]
```

```
#!/bin/ksh
until test -f $FILE
do
    sleep 60
done
echo "$FILE existe"
```

until

```
until command-list1
do
      command-liste2
 done
```

Programmation shell - break et continue

break n — n niveau de la boucle imbriquée arrêtée

continue n — itération suivante sur la boucle de niveau n

```
#!/bin/ksh
while echo "Please enter command"
read response
do
    case "$response" in
        'fin') break                    # arrêt
               ;;
        "")    continue                 # commande nulle
               ;;
        *)     eval $response           # do the command
               ;;
    esac
done
```

Programmation shell – à venir

- arrays : les tableaux
- function : les fonctions définies par l'utilisateur