

---

# Administration système UNIX

THIERRY BESANÇON – PHILIPPE WEILL

**2008 – 2009 / version 11.0**

Partie 1

Université Pierre et Marie Curie  
Formation Permanente  
4, place Jussieu  
75252 Paris Cedex 05  
web : <http://www.fp.upmc.fr>

---

Les animateurs de ce cours peuvent être joints aux adresses suivantes :

Thierry.Besancon@formation.jussieu.fr

Philippe.Weill@formation.jussieu.fr

Ce cours est disponible au format PDF sur le web à l'URL :

<http://www.formation.jussieu.fr/ars/2008-2009/UNIX/cours/>

Si vous améliorez ce cours, merci de nous envoyer vos modifications ! : -)

Des énoncés de TP et leurs corrigés sont disponibles sur le web à l'URL :

<http://www.formation.jussieu.fr/ars/2008-2009/UNIX/tp/>

Copyright (c) 1997-2009 by Thierry.Besancon@formation.jussieu.fr

This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, v1.0 or later (the latest version is available at <http://www.opencontent.org/openpub/>).

"...the number of UNIX installations has grown to 10, with more expected..."

- Dennis Ritchie and Ken Thompson, June 1972

## Table des matières

N<sup>o</sup> de transparent

<b>Chapitre 1</b>	<b>UNIX : généralités, historique</b>	<b>2</b>
§ 1	UNIX, un système d'exploitation	3
§ 2	Terminologie	4
§ 3	Panorama de quelques UNIX du marché	6
§ 4	Les différentes familles d'UNIX	9
§ 5	Distributions LINUX	12
§ 6	UNIX à la formation permanente	13
<b>Chapitre 2</b>	<b>Définition du rôle de l'administrateur</b>	<b>15</b>
§ 1	Les principales missions de l'administrateur	16
§ 2	Quelques règles de bon sens	19
§ 3	Connaissances de base d'un administrateur	21
<b>Chapitre 3</b>	<b>Premiers contacts avec UNIX</b>	<b>22</b>
§ 1	Utilisation du clavier	24
§ 2	Votre compte UNIX : <code>login</code> , mot de passe	33
§ 3	Principales règles sur les mots de passe	34
§ 4	Changer son mot de passe UNIX : <code>passwd</code> , <code>yppasswd</code>	35
§ 5	Connexion sur un terminal texte UNIX	36
§ 6	Connexion sur un terminal graphique UNIX	38
§ 7	Les langages de commandes UNIX : les shells	43
§ 8	Formes générales des commandes UNIX	49
<b>Chapitre 4</b>	<b>Sources de documentation</b>	<b>51</b>
§ 1	Introduction	52
§ 2	Documentation UNIX en ligne : <code>man</code>	53
§ 3	RFC, Internet drafts	58
§ 4	FAQ	59
§ 5	HOWTO Linux	60
§ 6	Newsgroups	61
§ 7	Moteur de recherche Google	62
§ 8	Documentations constructeur online	64
§ 9	Documentations généralistes online	65
§ 10	Librairies parisiennes	66
§ 11	Magazines	67
§ 12	Formats des documentations	68
<b>Chapitre 5</b>	<b>Éditeurs de texte UNIX</b>	<b>75</b>
§ 1	Panorama d'éditeurs de fichier texte	76
§ 2	Éditeur de fichier texte : <code>vi</code>	77
§ 3	Éditeur de fichier texte : <code>vim</code>	91
§ 4	Éditeur de fichier texte : <code>view</code>	92
§ 5	Fins de ligne	93
<b>Chapitre 6</b>	<b>Commandes de manipulation de base d'objets UNIX</b>	<b>94</b>
§ 1	Notions d'objets sous UNIX	95
§ 2	Inode	99
§ 3	Notions de fichier sous UNIX	100
§ 4	Notions de répertoire sous UNIX	101
§ 5	Notions de chemins absolus et relatifs	106
§ 6	Positionnement dans l'arborescence : <code>cd</code>	113
§ 7	Position dans l'arborescence : <code>pwd</code>	114
§ 8	Liste des objets : <code>ls</code>	115
§ 9	(Windows : Liste des objets : <code>dir.exe</code> )	131
§ 10	Création de répertoires : <code>mkdir</code>	132
§ 11	(Windows : création de répertoires : <code>md.exe</code> , <code>mkdir.exe</code> )	134
§ 12	Déplacer et renommer des objets : <code>mv</code>	135

§ 13	(Windows :: déplacer des objets : <code>move.exe</code> ) . . . . .	141
§ 14	(Windows :: renommer des objets : <code>ren.exe, rename.exe</code> ) . . . . .	142
§ 15	Duplication d'un objet : <code>cp</code> . . . . .	143
§ 16	(Windows :: Duplication d'un objet : <code>copy.exe</code> ) . . . . .	148
§ 17	(Windows :: Duplication d'un objet : <code>xcopy.exe</code> ) . . . . .	149
§ 18	Destruction d'un objet : <code>rm</code> . . . . .	150
§ 19	(Windows :: Destruction d'un objet : <code>del.exe</code> ) . . . . .	156
§ 20	Suppression de répertoires : <code>rmdir</code> . . . . .	157
§ 21	(Windows :: suppression d'un répertoire : <code>rd.exe</code> ) . . . . .	158
§ 22	Liens sur objets . . . . .	159
§ 23	Lien hard sur objets : <code>ln</code> . . . . .	164
§ 24	Lien symbolique sur objets : <code>ln -s</code> . . . . .	170
§ 25	(Windows :: raccourci, shortcut) . . . . .	177
§ 26	(Windows :: lien symbolique : junction) . . . . .	181
<b>Chapitre 7</b>	<b>Gestion de versions de fichiers</b> . . . . .	<b>185</b>
§ 1	Introduction . . . . .	186
§ 2	SCCS . . . . .	187
§ 3	RCS . . . . .	188
§ 4	CVS . . . . .	189
§ 5	SUBVERSION . . . . .	195
<b>Chapitre 8</b>	<b>Commandes de manipulation de base d'objets UNIX (suite)</b> . . . . .	<b>196</b>
§ 1	Affichage du contenu d'un fichier texte : <code>cat</code> . . . . .	197
§ 2	(Windows :: Affichage du contenu d'un fichier texte : <code>type.exe</code> ) . . . . .	198
§ 3	Affichage du contenu d'un fichier texte : <code>more</code> . . . . .	199
§ 4	(Windows :: Affichage du contenu d'un fichier texte : <code>more.exe</code> ) . . . . .	200
§ 5	Affichage du contenu d'un fichier texte : <code>less</code> . . . . .	201
§ 6	Comptage de lignes dans un fichier : <code>wc</code> . . . . .	202
§ 7	Comparaison de deux fichiers : <code>diff</code> . . . . .	203
§ 8	Comparaison de deux fichiers binaires : <code>cmp</code> . . . . .	208
§ 9	Extraction des premières lignes de fichiers : <code>head</code> . . . . .	209
§ 10	Extraction des dernières lignes de fichiers : <code>tail</code> . . . . .	210
§ 11	Extraction de colonnes de fichiers : <code>cut</code> . . . . .	213
§ 12	Tri d'un fichier : <code>sort</code> . . . . .	215
§ 13	(Windows :: Tri d'un fichier : <code>sort.exe</code> ) . . . . .	223
§ 14	Élimination des lignes redondantes d'un fichier : <code>uniq</code> . . . . .	224
§ 15	Création d'un fichier vide : <code>touch</code> . . . . .	226
§ 16	Modification des dates d'un objet : <code>touch</code> . . . . .	227
§ 17	Création d'objets temporaires : <code>/tmp</code> . . . . .	229
§ 18	(Windows :: variable <code>temp</code> , répertoire <code>temp</code> ) . . . . .	231
§ 19	Manipulation des noms d'objets : <code>basename</code> . . . . .	232
§ 20	Nature d'un objet : <code>file</code> . . . . .	234
§ 21	Affichage du contenu d'un objet binaire : <code>od</code> . . . . .	237
§ 22	Commande de traduction de caractères : <code>tr</code> . . . . .	239
§ 23	Information sur le remplissage des disques : <code>df</code> . . . . .	243
§ 24	(Windows :: Information sur le remplissage des disques : <code>df.exe</code> ) . . . . .	248
§ 25	Calcul de la place disque occupée : <code>du</code> . . . . .	250
§ 26	(Windows :: Calcul de la place disque occupée : <code>diruse.exe</code> ) . . . . .	256
§ 27	(Windows :: Calcul de la place disque occupée : <code>diskuse.exe</code> ) . . . . .	257
§ 28	Compression de fichiers : <code>compress, uncompress, zcat</code> . . . . .	258
§ 29	Compression de fichiers : <code>gzip, gunzip, gzcat</code> . . . . .	260
§ 30	Compression de fichiers : <code>bzip2, bunzip2, bzcat</code> . . . . .	262
§ 31	Archivage de fichiers/répertoires : <code>tar</code> . . . . .	265
§ 32	Commandes issues du monde Windows : <code>zip, unzip</code> . . . . .	270
§ 33	(Windows :: PowerArchiver Command Line) . . . . .	273

§ 34	Impression : <code>lpr, lpq, lprm</code> . . . . .	274
§ 35	Impression de fichiers texte : <code>a2ps</code> . . . . .	276
§ 36	Impression de fichiers texte : <code>enscript</code> . . . . .	278
§ 37	Utilitaires pour disquettes PC : <code>mtools, mcopy</code> . . . . .	280
§ 38	Utilisation de clefs USB . . . . .	281
<b>Chapitre 9</b>	<b>Attributs des objets UNIX</b> . . . . .	<b>292</b>
§ 1	Définition des droits d'accès d'un objet . . . . .	293
§ 2	Changements des droits d'accès d'un objet : <code>chmod</code> . . . . .	298
§ 3	Droits d'accès par défaut lors de création d'objets : <code>umask</code> . . . . .	301
§ 4	Régler le <code>umask</code> de façon permanente . . . . .	308
§ 5	Attribut spécial de fichier : <code>bit setuid</code> . . . . .	309
§ 6	Attribut spécial de fichier : <code>bit setgid</code> . . . . .	313
§ 7	Attribut spécial de répertoire : <code>sticky bit</code> . . . . .	317
§ 8	Attributs de date d'un objet : <code>mtime, atime, ctime</code> . . . . .	322
§ 9	Consultation de l'horloge : <code>date</code> . . . . .	325
§ 10	Modification des dates d'un objet : <code>touch</code> . . . . .	326
<b>Chapitre 10</b>	<b>Expressions régulières et commandes UNIX associées</b> . . . . .	<b>328</b>
§ 1	Regular expressions ( <code>regexps</code> ) . . . . .	329
§ 2	Recherche de <code>regexp</code> dans un fichier : <code>grep</code> . . . . .	345
§ 3	Modification à la volée de contenu de fichiers : <code>sed</code> . . . . .	354
<b>Chapitre 11</b>	<b>Commande de recherche d'objets : <code>find</code></b> . . . . .	<b>364</b>
§ 1	Recherche d'objets : <code>find</code> . . . . .	365
§ 2	Confusion courante . . . . .	377
§ 3	Quelques Difficultés . . . . .	378
<b>Chapitre 12</b>	<b>Commandes UNIX réseau de base</b> . . . . .	<b>380</b>
§ 1	Nom de machine : <code>hostname</code> . . . . .	381
§ 2	Nom de système, de machine : <code>uname</code> . . . . .	382
§ 3	Test de connectivité : <code>ping</code> . . . . .	384
§ 4	(Windows :: Test de connectivité : <code>ping.exe</code> ) . . . . .	386
§ 5	Tests de connectivité : <code>tracert</code> . . . . .	387
§ 6	(Windows :: Test de connectivité : <code>tracert.exe</code> ) . . . . .	390
§ 7	Transfert de fichiers : <code>ftp</code> . . . . .	391
§ 8	(Windows :: Transfert de fichiers : <code>ftp.exe</code> ) . . . . .	394
§ 9	Lancement de commande à distance : protocole SSH, <code>ssh</code> . . . . .	397
§ 10	(Windows :: Connexion à distance interactive SSH : <code>putty.exe</code> ) . . . . .	399
§ 11	Recopie de fichiers à distance : protocole SSH, <code>scp</code> . . . . .	400
§ 12	(Windows :: Recopie de fichiers à distance : <code>winscp.exe</code> ) . . . . .	401
§ 13	Point d'entrée réseau de la Formation Permanente . . . . .	403
§ 14	Liste des utilisateurs connectés : <code>users</code> . . . . .	404
§ 15	Liste des utilisateurs connectés : <code>who, who am i, whoami</code> . . . . .	405
§ 16	Liste des utilisateurs connectés : <code>w</code> . . . . .	406
§ 17	Liste des utilisateurs connectés : <code>finger</code> . . . . .	407
§ 18	Navigation Web : URL . . . . .	408
§ 19	Navigateur Web : <code>lynx</code> . . . . .	410
§ 20	Navigateur Web : <code>netscape</code> . . . . .	411
§ 21	Navigateur Web : <code>mozilla</code> . . . . .	412
§ 22	Navigateur Web : <code>firefox</code> . . . . .	413
§ 23	Navigateur Web : <code>opera</code> . . . . .	416
§ 24	Pseudo navigateur Web : <code>wget</code> . . . . .	417
§ 25	Pseudo navigateur Web : <code>curl</code> . . . . .	419
§ 26	Courrier électronique à la Formation Permanente . . . . .	421
§ 27	Courrier électronique : redirection . . . . .	423
§ 28	Courrier électronique : <code>mail, mailx, Mail</code> . . . . .	424
§ 29	Courrier électronique : <code>netscape</code> . . . . .	425

§ 30	Courrier électronique : <code>mozilla</code> . . . . .	426
§ 31	Courrier électronique : <code>thunderbird</code> . . . . .	427
§ 32	Courrier électronique : <code>webmail</code> de la Formation Permanente . . . . .	444
<b>Chapitre 13</b>	<b>Pratique du Bourne shell</b> . . . . .	<b>445</b>
§ 1	Affichage d'une chaîne de caractères : <code>echo</code> . . . . .	446
§ 2	Principe d'exécution par le shell d'une commande UNIX . . . . .	448
§ 3	Caractères spéciaux du shell : métacaractères . . . . .	449
§ 4	Métacaractères tabulation, espace . . . . .	450
§ 5	Métacaractère retour charriot . . . . .	451
§ 6	Métacaractère point-virgule . . . . .	452
§ 7	Métacaractères parenthèses ( ) . . . . .	454
§ 8	Contrôle des commandes lancées : <code>&amp;</code> , <code>fg</code> , <code>bg</code> , <code>kill</code> , <code>^C</code> , <code>^Z</code> . . . . .	457
§ 9	Contrôle des processus : <code>ps</code> . . . . .	465
§ 10	Contrôle des processus : <code>kill</code> . . . . .	468
§ 11	Contrôle des processus : <code>top</code> . . . . .	472
§ 12	(Windows :: Contrôle des processus : <code>taskmgr.exe</code> ) . . . . .	474
§ 13	Métacaractères : <code>'</code> , <code>"</code> , <code>\</code> . . . . .	475
§ 14	Lancement d'une commande par le shell . . . . .	480
§ 15	Interprétation de la ligne de commande . . . . .	481
§ 16	File descriptors : <code>stdin</code> , <code>stdout</code> , <code>stderr</code> . . . . .	486
§ 17	Métacaractères de redirection : <code>&lt;</code> , <code>&gt;</code> , <code>&gt;&gt;</code> , <code>&lt;&lt;</code> , <code>'</code> , <code> </code> , <code>2&gt;</code> , <code>&gt;&amp;</code> . . . . .	489
§ 18	Trou noir pour redirection : <code>/dev/null</code> . . . . .	504
§ 19	Métacaractères : <code>*</code> , <code>?</code> , <code>[]</code> , <code>[^]</code> . . . . .	506
§ 20	Métacaractère <code>\$</code> et variables shell . . . . .	508
§ 21	Variables d'environnement shell . . . . .	513
§ 22	Variable d'environnement <code>PATH</code> . . . . .	522
§ 23	Régler son <code>PATH</code> de façon permanente . . . . .	525
§ 24	Variable d'environnement <code>TERM</code> . . . . .	527
§ 25	Commande <code>stty</code> . . . . .	533
§ 26	(Windows :: Variables d'environnement) . . . . .	537
§ 27	Ordre d'évaluation de la ligne de commande . . . . .	539
§ 28	Se déconnecter du shell : <code>exit</code> , <code>Ctrl-D</code> . . . . .	542
§ 29	Shell de login . . . . .	543
§ 30	Shell interactif – Shell non interactif . . . . .	544
§ 31	Fichiers d'initialisation pour <code>bash</code> . . . . .	545
§ 32	Fichiers d'initialisation pour <code>sh</code> . . . . .	552
§ 33	Fichiers d'initialisation pour <code>tcsh</code> . . . . .	553
§ 34	Fichiers d'initialisation pour <code>csh</code> . . . . .	555
§ 35	Complétion interactive . . . . .	556
§ 36	(Windows :: Complétion interactive) . . . . .	557
<b>Chapitre 14</b>	<b>Programmation en Bourne shell</b> . . . . .	<b>558</b>
§ 1	Introduction . . . . .	559
§ 2	Caractéristiques d'un shell script . . . . .	560
§ 3	Structure d'un shell script . . . . .	561
§ 4	Code de retour d'un shell script : <code>exit</code> . . . . .	565
§ 5	Passage de paramètres à un shell script : <code>\$1</code> à <code>\$9</code> . . . . .	566
§ 6	Liste des paramètres d'un shell script : <code>\$*</code> , <code>\$@</code> . . . . .	572
§ 7	Variable prédéfinie <code>\$?</code> . . . . .	573
§ 8	Variable prédéfinie <code>\$\$</code> . . . . .	575
§ 9	Commandes internes du shell : <code>builtins</code> . . . . .	578
§ 10	Commandes internes du shell : <code>type</code> . . . . .	580
§ 11	Commande d'affichage : <code>builtin echo</code> , <code>/bin/echo</code> . . . . .	581
§ 12	Entrée interactive : <code>read</code> . . . . .	587
§ 13	Structure <code>if-then-else</code> . . . . .	590

§ 14	Structure <code>case</code> . . . . .	594
§ 15	Commande <code>test</code> . . . . .	599
§ 16	Structure de boucles : <code>while</code> , <code>for</code> , <code>until</code> . . . . .	605
§ 17	Contrôle du flux d'exécution : <code>break</code> , <code>continue</code> . . . . .	613
§ 18	Debugging d'un shell script : <code>set -x</code> . . . . .	616
§ 19	Faux ami : commande <code>script</code> . . . . .	617
<b>Chapitre 15</b>	<b>Programmation en langage AWK</b> . . . . .	<b>620</b>
§ 1	Introduction . . . . .	621
§ 2	Syntaxe de la commande <code>awk</code> . . . . .	622
§ 3	Structure d'un programme AWK . . . . .	623
§ 4	Variables prédéfinies de AWK . . . . .	624
§ 5	Masques sous AWK . . . . .	625
§ 6	Opérateurs de AWK . . . . .	626
§ 7	Instructions de AWK . . . . .	627
§ 8	Principales fonctions prédéfinies . . . . .	629
§ 9	Exemples . . . . .	630
<b>Chapitre 16</b>	<b>Langage perl</b> . . . . .	<b>632</b>
§ 1	Introduction . . . . .	633
§ 2	Les nombres Perl . . . . .	634
§ 3	Les chaînes de caractères Perl . . . . .	635
§ 4	Les variables Perl . . . . .	636
§ 5	Les listes simples . . . . .	638
§ 6	Les listes associatives . . . . .	641
§ 7	Structure de contrôle <code>if/else</code> . . . . .	644
§ 8	Structure de contrôle <code>while, until, for</code> . . . . .	646
§ 9	Structure de contrôle <code>foreach</code> . . . . .	647
§ 10	Opérateurs de comparaison / Opérateurs logiques . . . . .	648
§ 11	Entrées/sorties : flux . . . . .	649
§ 12	Entrées/sorties : entrée standard, <code>STDIN</code> . . . . .	650
§ 13	Entrées/sorties : sortie standard, <code>STDOUT</code> . . . . .	651
§ 14	Entrées/sorties : sortie erreur, <code>STDERR</code> . . . . .	652
§ 15	Créer, ouvrir et fermer un fichier, <code>open()</code> , <code>close()</code> , <code>eof()</code> , <code>die()</code> . . . . .	653
§ 16	Lire et écrire dans un fichier . . . . .	654
§ 17	Supprimer un fichier, <code>unlink()</code> . . . . .	655
§ 18	Renommer un fichier, <code>rename()</code> . . . . .	656
§ 19	Créer un lien symbolique, <code>symlink()</code> , <code>readlink()</code> . . . . .	657
§ 20	Modifier les propriétés d'un fichier, <code>chmod()</code> , <code>chown()</code> , <code>chgrp()</code> . . . . .	658
§ 21	Créer et supprimer un répertoire, <code>mkdir()</code> , <code>rmdir()</code> . . . . .	659
§ 22	Exécuter un programme, <code>exec()</code> , <code>open()</code> . . . . .	660
§ 23	Expressions régulières, <code>\$_</code> . . . . .	662
§ 24	Expressions régulières : substitution dans une chaîne, <code>s///, ~ =</code> . . . . .	663
§ 25	Expressions régulières : <code>split()</code> , <code>join()</code> . . . . .	664
§ 26	Les fonctions . . . . .	665

# Administration de systèmes UNIX

Thierry Besançon

Formation Permanente de l'Université de Paris 6

Formation ARS 2008 – 2009

Tôme 1

## Chapitre 1

# *UNIX : généralités, historique*



Les missions d'un système d'exploitation sont :

- mise à disposition de ressources matérielles : espace disque, temps d'exécution sur le microprocesseur central, espace mémoire, etc.
- partage équitable de ces ressources entre les utilisateurs pour atteindre le but de système multi-utilisateurs

<b>Mono utilisateur</b>	Une seule personne utilise l'ordinateur
<b>Multi utilisateur</b>	Plusieurs personnes peuvent utiliser le système en même temps. Le système s'assure qu'un utilisateur n'interfère pas sur un autre.
<b>Mono tâche</b>	Un seul processus tourne à un instant.
<b>Multi tâche</b>	Plusieurs processus donnent l'impression de tourner en même temps.
<b>Multi tâche préemptif</b>	L'OS détermine quand un processus a eu assez de temps CPU.
<b>Multi tâche non préemptif</b>	Le processus détermine lui même quand il a eu assez de temps CPU.

## Exemples

<b>MS DOS</b>	mono utilisateur, mono tâche
<b>Windows 95, 98, ME</b>	mono utilisateur, multi tâche non préemptif
<b>Windows NT, 2000, XP, 2003</b>	mono utilisateur, multi tâche préemptif
<b>IBM OS/2</b>	mono utilisateur, multi tâche préemptif
<b>UNIX</b>	multi utilisateur, multi tâche préemptif

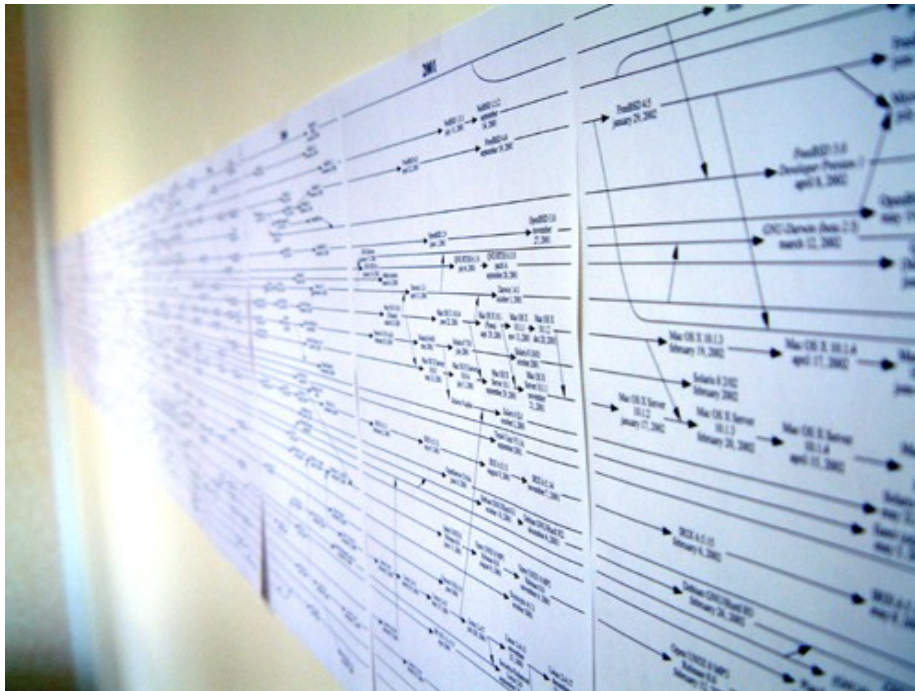
## Chapitre 1 • UNIX : généralités, historique

### §1.3 • Panorama de quelques UNIX du marché

<i>Marque</i>	<i>Site web</i>	<i>Version d'UNIX</i>	<i>Constructeur de hardware</i>
<b>APPLE</b>	<a href="http://www.apple.com">http://www.apple.com</a>	MacOS X 10.x	oui
<b>CRAY</b>	<a href="http://www.cray.com">http://www.cray.com</a>	Unicos?.?	oui
<b>HP</b>	<a href="http://www.hp.com">http://www.hp.com</a>	HP-UX 11.x	oui
<b>COMPAQ</b>	<a href="http://www.digital.com">http://www.digital.com</a>	Tru64 Unix 5.x	oui
<b>IBM</b>	<a href="http://www.ibm.com">http://www.ibm.com</a>	AIX 5.x	oui
<b>SGI</b>	<a href="http://www.sgi.com">http://www.sgi.com</a>	IRIX 6.x.y	oui
<b>SUN</b>	<a href="http://www.sun.com">http://www.sun.com</a>	Solaris 10	oui
<b>SANTA CRUZ</b>	<a href="http://www.sco.com">http://www.sco.com</a>	Unixware 7.x	non
<b>LINUX</b>	<a href="http://www.kernel.org">http://www.kernel.org</a>	noyau 2.6.x	non
<b>FREEBSD</b>	<a href="http://www.freebsd.org">http://www.freebsd.org</a>	FreeBSD 6.x	non
<b>NETBSD</b>	<a href="http://www.netbsd.org">http://www.netbsd.org</a>	NetBSD 1.x	non
<b>OPENBSD</b>	<a href="http://www.openbsd.org">http://www.openbsd.org</a>	OpenBSD 3.x	non

L'arbre généalogique d'UNIX est très complexe. Cf

<http://www.levenez.com/unix/>



Se reporter à l'annexe pour un schéma détaillé.

Plusieurs guides de transition d'un système à un autre système sont disponibles.

Se reporter à <http://www.unixporting.com/porting-guides.html>

**Du point de vue de l'utilisateur, les divers UNIX se ressemblent beaucoup.**

**Du point de vue de l'administration, les divers UNIX ont chacun des spécificités** (les commandes liées au hardware varient, on trouve des extensions propres à chaque constructeur). En pratique, l'administrateur attend toujours.

Plusieurs tentatives d'unification :

- *System V Interface Definition* de AT&T (SVID, SVID2, SVID3 en 1989)
- IEEE POSIX (POSIX1003.1 en 1990)
- X/OPEN Portability Guide (XPG4 en 1993) du consortium X/OPEN (créé en 1984)

Mais...

Il reste 2 grandes familles d'UNIX issues d'un schisme :

- la famille **System V** avec notamment la dernière version connue sous le nom de **System V release 4** (alias *SVR4*)
- la famille **BSD** issue de l'université de Berkeley (BSD  $\equiv$  *Berkeley Software Distribution*)

Votre rôle : connaître les principes et les mécanismes d'UNIX afin de savoir s'adapter à n'importe quel UNIX.



Ken Thompson et Dennis M. Ritchie, les parents d'UNIX  
On notera les teletypes 33 !

## Chapitre 1 • UNIX : généralités, historique

### §1.5 • Distributions LINUX

Il existe beaucoup de distributions LINUX car LINUX n'est la propriété de personnes mais de toute la communauté de programmeurs informatiques.

Les principales distributions sont :

- Red Hat, <http://www.redhat.com>
- Suse, <http://www.suse.com>
- Mandrake / Mandriva, <http://www.mandrake.com>
- Debian, <http://www.debian.org>
- Knoppix, <http://www.knoppix.org>

La salle de TP de la Formation Permanente est équipée de PC sous Red Hat ou Mandrake (information non disponible au moment de l'écriture de ce support de cours).

Vos interlocuteurs (dans cet ordre décroissant d'importance) :

- Vassiliki Spathis  
email : `Vassiliki.Spathis@formation.jussieu.fr`  
Responsable des formations, elle informera les autres techniciens des interventions à réaliser.
- adresse email « assistance »  
email : `assistance@formation.jussieu.fr`

Vous **devez** signaler :

- tout problème de compte ; **faire au plus vite si vous soupçonnez que votre compte est piraté.**
- problème sur les imprimantes : cartouche d'encre vide, bourrage papier, etc. ;
- problème sur le poste de travail : terminal en mode inhabituel, clavier cassé, souris hors service, etc. ;
- problème anormal avec un logiciel : le logiciel ne fonctionne plus comme d'habitude, un logiciel a disparu, le logiciel ne fonctionne pas du tout comme le précise la documentation, etc. ;

Vous devez signaler les problèmes comme un patient donne ses symptômes à un médecin.

**Un jour prochain, c'est à vous que les utilisateurs signaleront les problèmes. Alors mettez-vous à notre place dès maintenant en adoptant une attitude d'administrateur système : précision, détails, etc.**

## Chapitre 2

# Définition du rôle de l'administrateur

## Chapitre 2 • Définition du rôle de l'administrateur

### §2.1 • Les principales missions de l'administrateur



- gérer les comptes utilisateurs (tâche simple et automatisable)
- assister et éduquer les utilisateurs (réponses à leurs questions, documentation à jour)
- gérer les logiciels (installation, configuration, mise à jour)
- gérer le matériel (panne, remplacement, ajout)
- assurer la sécurité du système et des utilisateurs (sauvegardes fiables et régulières, contrôle d'accès, utilisations abusives de ressources)
- vérification de l'adéquation du matériel avec son utilisation (identifier les goulets d'étranglement)
- maintenance de premier niveau (diagnostiquer une panne, appel de la maintenance constructeur)
- gestion quotidienne (multiples tâches, petites ou grosses)

Autres facettes du métier :

- diplomatie, police
- aspects légaux (chiffrement, etc.)
- enquêtes judiciaires (vol, saccage, piratage informatique, articles pédophiles, etc.)
- relations commerciales
- politique d'utilisation des machines

Bien sûr, la charge de travail dépend de la taille du site.

L'administrateur est en première ligne lorsqu'un problème surgit. C'est lui qu'on incrimine naturellement lorsque quelque chose ne marche pas.



- 1 Votre pire ennemi, c'est vous : attention à ce que vous faites !  
Exemple « `rm /tmp *` » (explications sur l'expansion des metacharacters du shell page 506)
- 2 Si vous êtes fatigué, ne faites rien.
- 3 Pas de modification importante un vendredi après-midi.
- 4 Soyez sûr de pouvoir revenir en arrière : sauvegarder tout fichier qui doit être modifié :  

```
% mv config.ini config.ini.orig  
% cp config.ini.orig config.ini  
% vi config.ini
```
- 5 Documentez ce que vous faites.
- 6 Faites comme si vous ne pouviez pas venir demain.

**Administrateur système == technicité + rigueur + bon sens**

Administrateur d'UNIX : d'abord un **utilisateur expert d'UNIX**

- environnement utilisateur
- aide en ligne
- système de fichiers
- utilisation du shell
- utilisation d'un éditeur de texte
- commandes de base
- programmation shell

## Chapitre 3

# *Premiers contacts avec UNIX*

Avant de commencer : n'ayez pas peur d'expérimenter. Le système ne vous fera pas de mal.

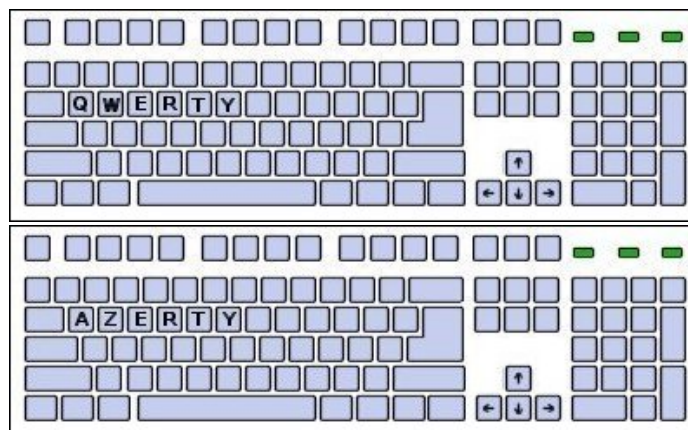


En mode utilisateur, vous ne pouvez rien abîmer en utilisant le système. UNIX, par conception, possède des notions de sécurité, afin d'éviter aux utilisateurs «normaux» de le déconfigurer.

En mode administrateur, bien sûr, faites attention. **On limitera tout travail en mode administrateur au minimum.**

## Il faut savoir se servir d'un clavier !

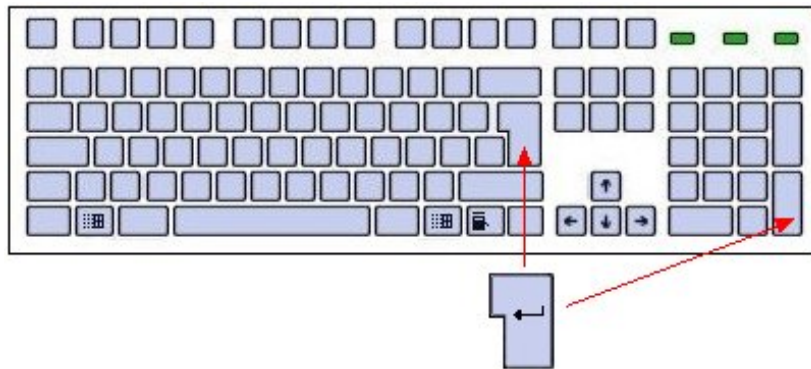
Il existe des claviers : américain, français, etc. déclinés en autant de modèles qu'imaginables.



(images de claviers trouvées sur le site

<http://www.freinet.org/creactif/bruyeres/lab0111.html>)

Rappel : touche enter, entrée, valider



Rappel : touche shift



Appuyer sur la touche donne « à »



Appuyer sur ces 2 touches donne « 0 »

Rappel : touche shift lock, caps lock, majuscule



On n'utilise pas la touche caps lock. Non !

Hack windows : Ctrl2cap d'URL

<http://www.sysinternals.com/files/ctrl2cap.zip>

Rappel : touche tab, tabulation



Rappel : touche escape, esc, Échap, échappement



Equivalent : Ctrl + [

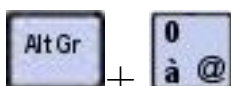
Rappel : touche Alt Gr (absente sur clavier QWERTY)



Appuyer sur la touche donne « à »



Appuyer sur ces 2 touches donne « 0 »



Appuyer sur ces 2 touches donne « @ »



Rappel : pavé numérique



Le pavé numérique

On n'utilise pas le pavé numérique. Non !

Rappel : num lock



On n'utilise pas num lock. Non !

## Chapitre 3 • Premiers contacts avec UNIX

### §3.2 • Votre compte UNIX : login, mot de passe

Un utilisateur UNIX est équivalent à :

- un identificateur (sur 8 lettres en général), son «nom» au sens informatique ; appelé **login** ;
- un mot de passe **confidentiel** ;

Gare aux sanctions en cas d'«amusement» avec le compte d'un autre utilisateur !

Il existe des chartes informatiques  $\equiv$  règlements informatiques.

## Chapitre 3 • Premiers contacts avec UNIX

### §3.3 • Principales règles sur les mots de passe

- un mot de passe ne se prête pas !
- un mot de passe ne s'oublie pas !
- un mot de passe n'est pas facile à trouver ! :
  - évitez qu'il ne se rapporte pas à vous (nom, voiture, chien)
  - évitez les mots dans des dictionnaires
  - évitez les prénoms
  - il doit comporter au moins 6 caractères, en général 8
  - les majuscules et les minuscules sont différenciées
  - utiliser des chiffres et des caractères spéciaux  
par exemple « Kpiten[ », « \&7oubon », etc. <sup>1</sup>

---

<sup>1</sup>Ces mots de passe sont mauvais. Pourquoi ?



## Chapitre 3 • Premiers contacts avec UNIX

### §3.4 • Changer son mot de passe UNIX : passwd, yppasswd

La commande standard pour changer son mot de passe sur une machine UNIX est `passwd`.

Sur les systèmes UNIX qui utilisent un mécanisme de centralisation des mots de passe (appelé NIS), la commande pour changer son mot de passe est `yppasswd`.

**C'est le cas de la formation permanente ⇒ `yppasswd`**

## Chapitre 3 • Premiers contacts avec UNIX

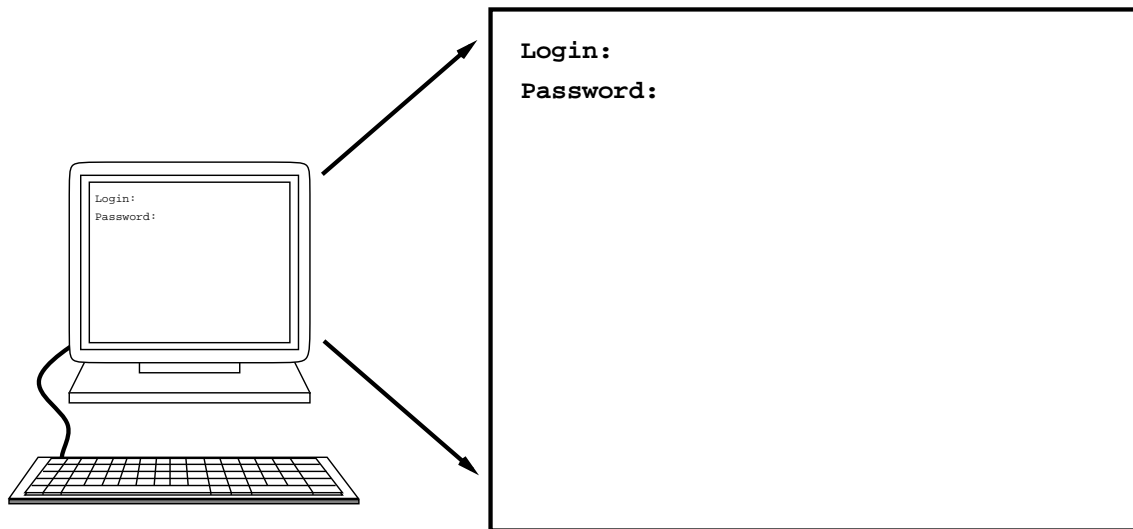
### §3.5 • Connexion sur un terminal texte UNIX



Terminal texte (modèle VT100)

Se reporter à <http://www.vt100.net/>

La demande du login et du mot de passe ressemble globalement à :



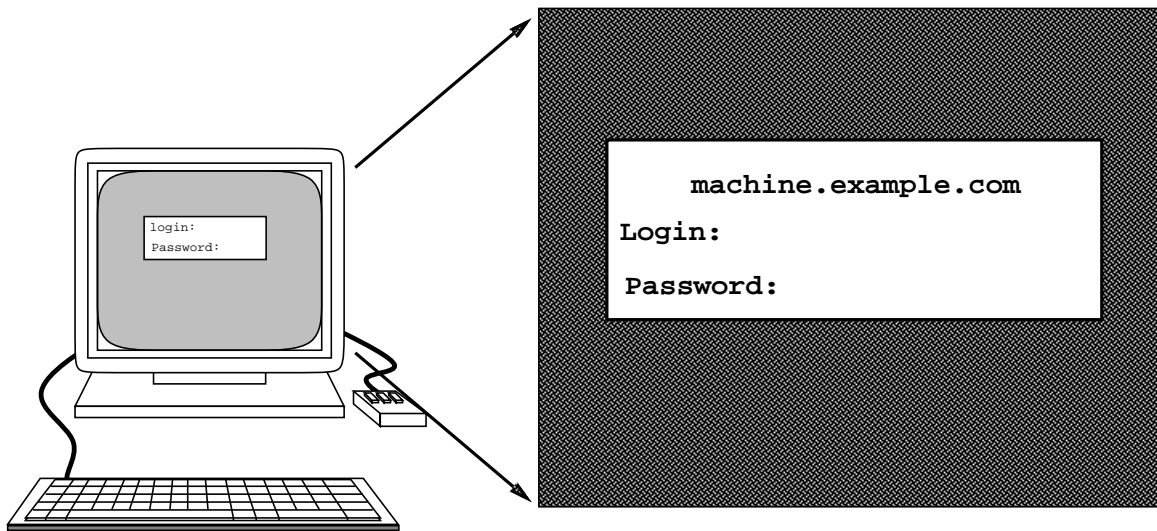
## Chapitre 3 • Premiers contacts avec UNIX

### §3.6 • Connexion sur un terminal graphique UNIX



Station de travail UNIX (SUN Blade 100)

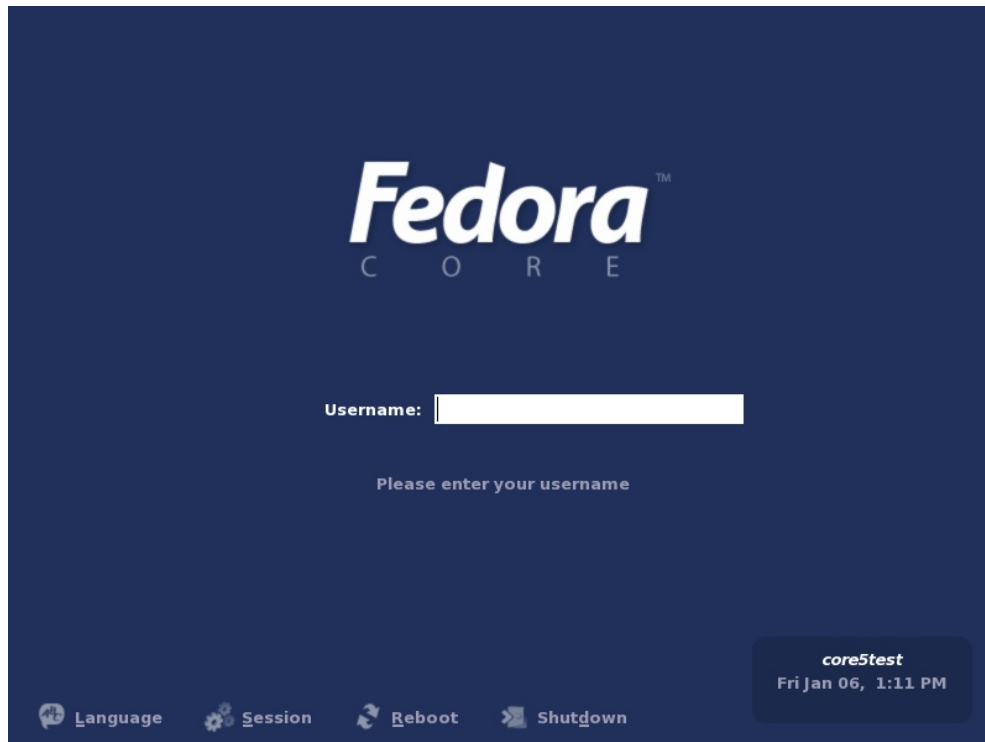
La demande du login et du mot de passe ressemble globalement à :



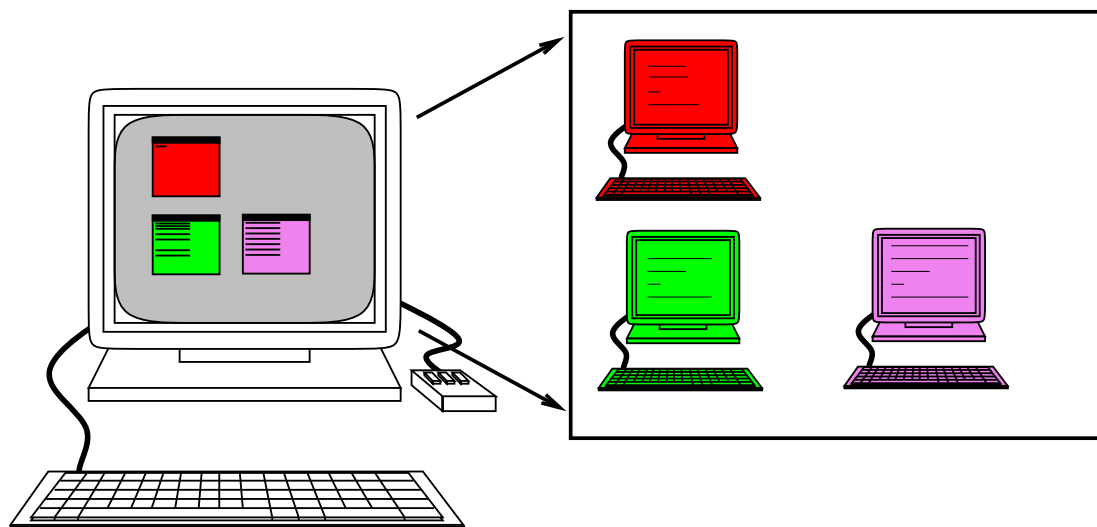
Par exemple :



Par exemple :



Une fois connecté via l'interface graphique, on utilisera principalement un programme d'émulation de terminal de type texte qui fournit dans une fenêtre une connexion comme sur un terminal texte :



L'émulateur de terminal s'appelle « `xterm` ».

## Chapitre 3 • Premiers contacts avec UNIX

### §3.7 • Les langages de commandes UNIX : les shells

A l'origine, des teletypes puis des consoles texte.

⇒ l'interaction de base se fait au moyen de phrases à taper sur un clavier (par opposition aux interfaces graphiques à la Windows ou de Macintosh).



A gauche, console DIGITAL VT100.  
A droite, teletype DIGITAL.

Le shell est un programme qui permet la saisie et l'interprétation de ce qui est tapé. Le shell est juste une interface avec le système.

MS-DOS comporte un shell aux possibilités restreintes par rapport aux shells UNIX.

Le shell est aussi un vrai langage de programmation, interprété (non compilé) offrant les structures de base de programmation de tout autre langage.

Sous UNIX, le shell est un programme au même titre qu'un autre. Le shell de travail est **interchangeable** par un autre shell (à la syntaxe près comme de bien entendu).

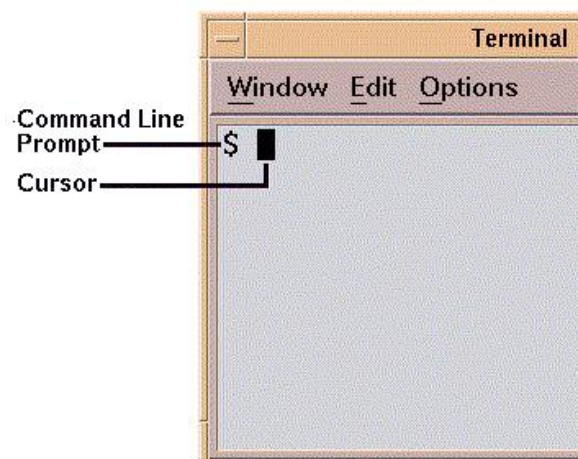
Les shells les plus répandus :

Shell	Nom du programme	Description
Bourne shell	sh	disponible sur toute plateforme UNIX
C shell	csh	shell développé pour BSD
Korn shell	ksh	Bourne shell amélioré par AT&T
Bourne again shell	bash	Shell distribué avec linux ; version améliorée de sh et csh

Dans ce cours, on distinguera le **shell de programmation** (car on peut programmer grâce à un interpréteur de commandes s'il est bien pensé) du **shell de travail** lors d'une session interactive. Les 2 shells n'ont pas de raison d'être identiques (cf plus loin sur ce que cela implique).

Tous les shells se présentent sous la même forme à l'écran lorsqu'ils fonctionnent :

- une chaîne de caractères affiche que le shell attend que l'utilisateur tape quelque chose au clavier ; c'est le **prompt**.
- un *curseur* qui va se déplacer au fur et à mesure de la saisie des commandes



Pour ce cours, on utilisera le caractère % pour désigner le prompt d'un utilisateur normal :

```
% commande-utilisateur
```

Pour ce cours, on utilisera le caractère # pour désigner le prompt de l'administrateur :

```
# commande-administrateur
```

**Il ne faudra jamais taper la chaîne de prompt lorsque vous testerez par vous mêmes les commandes indiquées.**

Pour terminer une session shell, on tape la commande commune à tous les shells :

```
% exit
```

## Chapitre 3 • Premiers contacts avec UNIX

## §3.8 • Formes générales des commandes UNIX

Une commande UNIX  $\equiv$  un ensemble de mots séparés par des caractères blancs (caractère espace, tabulation)

Le premier mot : le nom de la commande

Le reste des mots : les paramètres de la commande

Particularités de certains mots : des options qui changent le comportement de la commande

En pratique on trouvera donc écrit :

```
commande [options] parametres
```

Les 2 crochets « [ » et « ] » indiquent que les options ne sont pas obligatoires. Il ne faut pas taper ces crochets sur la ligne de commande.

### ◇ Comment spécifie-t-on une option ?

Une option est quelque chose de prévu par le programme  $\Rightarrow$  c'est le programmeur qui aura toujours le dernier mot.

Il reste une tendance générale : Une option est introduite par le signe « - » et est souvent constituée d'une seule lettre comme par exemple « -a ». (mais attention aux exceptions nombreuses)

Souvent on pourra cumuler et condenser des options :

```
ls -a -l  $\equiv$  ls -al
```

Souvent (mais pas tout le temps), l'ordre des options n'a pas d'importance. (cf `getopt(1)` ou `getopt(3)`)

```
ls -a -l  $\equiv$  ls -al  $\equiv$  ls -la  $\equiv$  ls -l -a
```



## Chapitre 4

***Sources de documentation***

## Chapitre 4 • Sources de documentation

## §4.1 • Introduction

Beaucoup de documentation disponible. **Il faut lire la documentation.**  
Souvent en anglais.



Extrait d'un rapport d'un ancien élève de ARS :

*Logiciel XXXXX : Documentation claire et exhaustive (j'avais le choix entre la version Anglaise ou Japonaise, la version Anglaise est très bien!!!!!!!)*

Il existe une documentation électronique accessible pendant le fonctionnement du système : c'est l'aide en ligne.

La commande donnant l'aide est **man**. Elle donne accès aux pages de manuel des commandes UNIX qui sont réparties selon des sections comme suit :

- section 1 ≡ commandes normales
- section 2 ≡ appels systèmes
- section 3 ≡ fonctions de programmation C
- section 4 ≡ périphériques et pilotes de périphériques
- section 5 ≡ format de fichiers système
- section 6 ≡ jeux
- section 7 ≡ divers
- section 8 ≡ commandes de gestion du système

Lorsque l'on verra `getopt (3)`, il faudra se reporter à la commande `getopt` de la section **3** du manuel.

Syntaxe de la commande `man` :

```
man [options] commande
```

avec en particulier comme option :

```
man [numero de section] commande
```

```
man [-s numero de section] commande
```

⇒ Inconvénient : il faut connaître le nom de la commande (nom anglais très souvent)

L'aide est plus là pour se rappeler les nombreuses options des commandes et leurs syntaxes particulières.

Exemple : « man fortune » renvoie :

```

XTerm vt100

FORTUNE(6)          GAMES AND DEMOS          FORTUNE(6)

NAME
  fortune - print a random, hopefully interesting, adage

SYNOPSIS
  /usr/games/fortune [ - ] [ -alsw ] [ filename ]

DESCRIPTION
  fortune with no arguments prints out a random adage. The
  flags mean:

--LESS-- /tmp/man4219 37%

```

On remarque :

- affichage page d'écran par page d'écran pour mieux lire la doc
- plusieurs rubriques (NAME, SYNOPSIS, DESCRIPTION, ...)

On navigue entre les pages d'écran de la documentation par :

- la touche SPC pour avancer (ou f ≡ *forward*)
- la touche b pour reculer (b ≡ *backward*)

```

XTerm vt100

-a Choose from either list of adages.
-l Long messages only.
-s Short messages only.
-w Waits before termination for an amount of time
  calculated from the number of characters in the
  message. This is useful if it is executed as part
  of the logout procedure to guarantee that the mes-
  sage can be read before the screen is cleared.

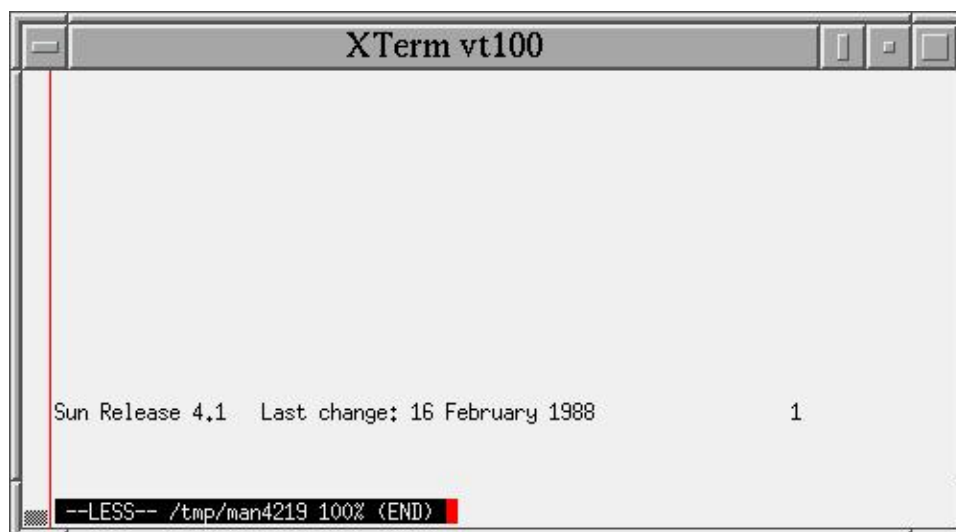
FILES
  /usr/games/lib/fortunes.dat

--LESS-- /tmp/man4219 89%

```

On quitte :

- quand on arrive à la fin de la documentation
- prématurément par la touche q ( $q \equiv quit$ )



## Chapitre 4 • Sources de documentation

### §4.3 • RFC, Internet drafts

RFC = *Request For Comments*

Documents de référence en anglais récupérables aux adresses :

- <ftp://ftp.lip6.fr/pub/rfc/rfc/>
- <ftp://ftp.lip6.fr/pub/rfc/internet-drafts/>
- <http://abcdrfc.free.fr/>

De nombreux autres sites existent.

(en anglais *Frequently Asked Questions*, en français *Foire Aux Questions*)

Documents en anglais récupérables aux adresses :

- `ftp://ftp.lip6.fr/pub/doc/faqs/`

De nombreux autres sites existent.

Documents en anglais récupérables aux adresses :

- `ftp://ftp.lip6.fr/pub/linux/french/docs/`

De nombreux autres sites existent.

Les newsgroups sont des forums de discussion sur internet.

Les thèmes en sont variés. Certains forums sont dans une langue autre que l'anglais.

Sur jussieu, le serveur de news est « `news.jussieu.fr` »

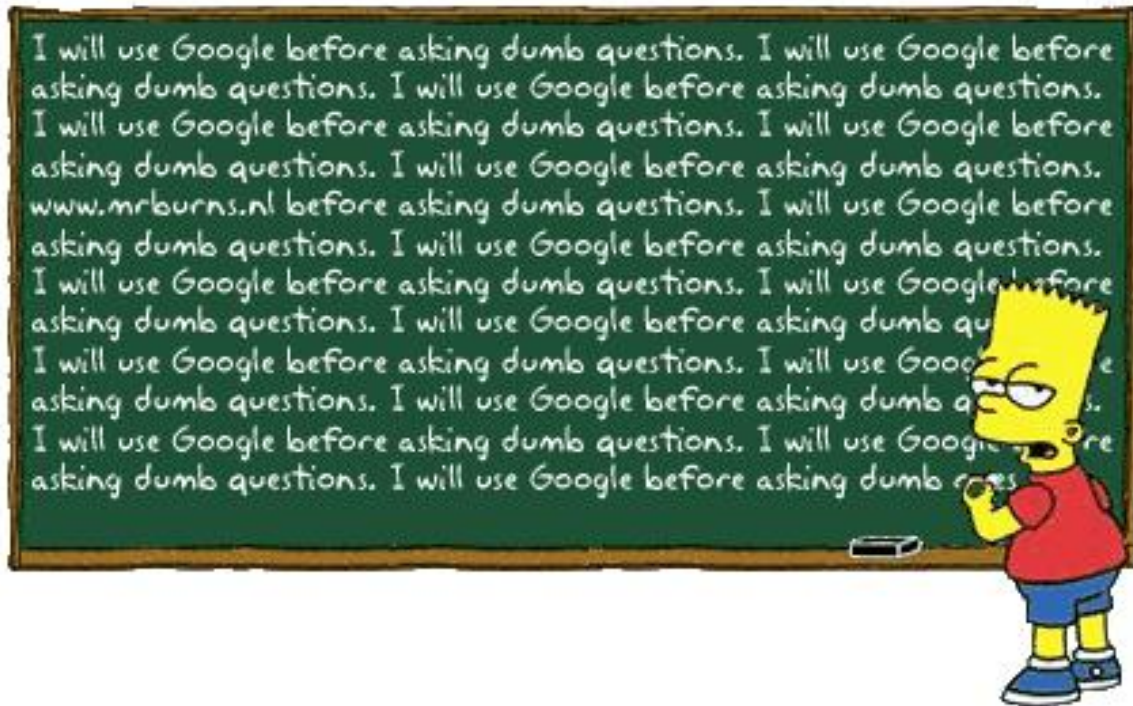
Le protocole réseau des news s'appelle NNTP

Le site `http://www.google.com` offre un moteur de recherche très efficace.



De nombreux autres sites de moteur de recherche existent.

Humour :



## Chapitre 4 • Sources de documentation

### §4.8 • Documentations constructeur online

Certains constructeurs UNIX mettent des documentations et documents online.

Se reporter par exemple à :

- <http://docs.sun.com/>

De nombreux autres sites existent.

Tendance aux encyclopédies collaboratives à base de WIKI :

- <http://www.wikipedia.org> (décliné en plusieurs langues : <http://fr.wikipedia.org>, <http://en.wikipedia.org>, etc.)
- <http://www.commentcamarche.net>
- <http://www.dicodunet.com>
- <http://www.labo-cisco.com>
- <http://www.labo-microsoft.com>

De nombreux autres sites existent.

Certaines librairies ont un rayon informatique bien fourni :

- Le Monde en Tique  
6 rue Maître Albert, 75005 Paris  
<http://www.lmet.fr/>
- Eyrolles  
61 boulevard Saint Germain, 75005 Paris  
<http://www.eyrolles.fr/>
- Infothèque  
81 rue d'Amsterdam, 75008 Paris  
<http://www.infotheque.com/>



De nombreux magazines parlent de LINUX.

- vulgarisation de domaines anciennement réservés à un cercle d'initiés
- CDROM vendus avec ces magazines.
- prix abordables

Quelques magazines que j'apprécie :

- LINUX magazine france
- LINUX journal
- MISC

A vous de vous faire votre opinion. . .

Liste de quelques formats les plus répandus (en vrac)

◇ format PDF : extension « .pdf »

à lire avec :

- Acrobat Reader ; <http://www.adobe.com/products/acrobat/>  
plateforme : UNIX, WINDOWS, MacOS
- Ghostscript ; <http://www.ghostscript.com/>  
plateforme : UNIX, WINDOWS, MacOS
- xpdf ; <http://www.foolabs.com/xpdf/>  
plateforme : UNIX

De nombreux outils dérivés de Ghostscript existent.

◇ format Postscript : extension « .ps »

à lire avec :

- Ghostscript ; <http://www.ghostscript.com/>  
plateforme : UNIX, WINDOWS, MacOS
- ghostview ; <ftp://ftp.lip6.fr/pub/gnu/ghostview/>  
plateforme : UNIX, WINDOWS
- gv ; <http://wwwthep.physik.uni-mainz.de/~plass/gv/>  
plateforme : UNIX, WINDOWS

De nombreux outils dérivés de Ghostscript existent.

◇ format Microsoft Word : extension « .doc »

à lire avec :

- Microsoft Word ; <http://www.microsoft.com/office/word/>  
plateforme : WINDOWS, MacOS
- Microsoft Word viewer ; <http://www.microsoft.com/???/>  
plateforme : WINDOWS
- Star Office ; <http://www.sun.com/software/star/staroffice/>  
plateforme : SOLARIS, LINUX, WINDOWS
- Open Office ; <http://www.openoffice.org/>  
plateforme : UNIX, WINDOWS, MacOS
- antiword ; <http://www.antiword.org/>  
plateforme : UNIX

Peu d'outils sous UNIX en dehors de ceux-ci.

### ◇ format Microsoft Excel : extension « .xls »

à lire avec :

- Microsoft Excel; <http://www.microsoft.com/office/word/>  
plateforme : WINDOWS, MacOS
- Microsoft Excel viewer; <http://www.microsoft.com/???/>  
plateforme : WINDOWS
- Star Office; <http://www.sun.com/software/star/staroffice/>  
plateforme : SOLARIS, LINUX, WINDOWS
- Open Office; <http://www.openoffice.org/>  
plateforme : UNIX, WINDOWS, MacOS
- GNUmeric; <http://www.gnome.org/projects/gnumeric/>  
plateforme : UNIX

Peu d'outils sous UNIX en dehors de ceux-ci.

### ◇ format texte : pas d'extension particulière

à lire avec n'importe quel éditeur de texte :

- vi ; standard  
plateforme : UNIX, WINDOWS, MacOS ?
- emacs; <http://www.gnu.org/software/emacs/>  
plateforme : UNIX, WINDOWS, MacOS ?

Nombreux autres outils sous UNIX en dehors de ceux-ci.

Mais « vi » et « emacs » restent les meilleurs. Le reste est une plaisanterie ou une réinvention de « vi » ou « emacs ».

◇ **format HTML** : extension « .html » ou « .htm »

A lire avec n'importe quel navigateur web :

- Mozilla ; <http://www.mozilla.org/> ; Obsolète : voir Firefox  
plateforme : UNIX, WINDOWS, MacOS
- Firefox ; <http://www.mozilla.com/>  
plateforme : UNIX, WINDOWS, MacOS
- Opera ; <http://www.opera.com/>  
plateforme : UNIX, WINDOWS, MacOS
- Galeon ; intégré au bureau GNOME ;  
<http://galeon.sourceforge.net/>  
plateforme : UNIX
- Konqueror ; intégré au bureau KDE ; <http://www.konqueror.org/>  
plateforme : UNIX

D'autres navigateurs web existent.

A noter Nvu ; Editeur de pages HTML dérivé de Mozilla ;

<http://www.nvu.com/>

plateforme : UNIX, WINDOWS, MacOS

◇ **format graphique** : extension « .jpg » ou « .gif » ou « .png » ou autre

à lire avec :

- GIMP ; <http://www.gimp.org/>  
dessins bitmap à la Adobe Photoshop  
plateforme : UNIX, WINDOWS, MacOS
- XnView ;  
<http://perso.orange.fr/pierre.g/xnview/frhome.html>  
plateforme : UNIX, WINDOWS, MacOS
- Inkscape ; <http://www.inkscape.org/>  
dessins vectoriels à la Adobe Illustrator  
plateforme : UNIX, WINDOWS, MacOS
- Dia ; <http://www.gnome.org/projects/dia/>  
dessins à la Microsoft Visio  
plateforme : UNIX, WINDOWS

De nombreux outils graphiques existent.

## Chapitre 5

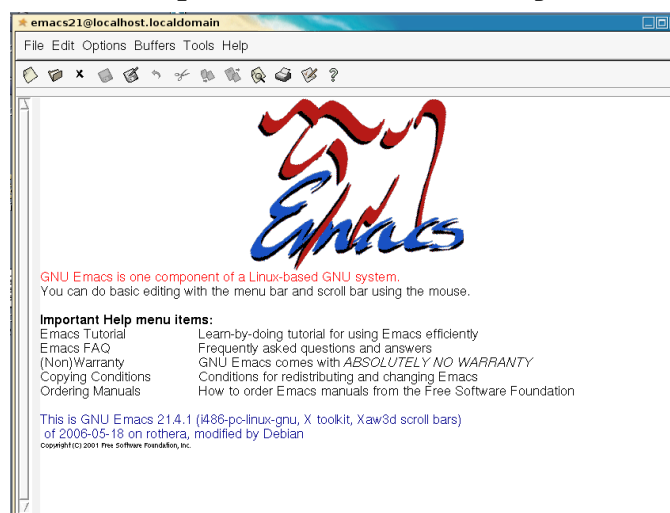
*Editeurs de texte UNIX*

## Chapitre 5 • Editeurs de texte UNIX

## §5.1 • Panorama d'éditeurs de fichier texte

Il existe beaucoup d'éditeurs de texte sous UNIX mais seuls quelques uns sont suffisamment robustes pour être utilisés efficacement et avec confiance :

- « vi » : seul éditeur de texte standard sous UNIX
- « emacs » : très puissant, complexe à maîtriser, simple une fois qu'on sait s'en servir. Cf <http://www.emacs.org>.



(en anglais *visual interface*)

C'est l'éditeur de texte standard sur UNIX. Il fonctionne sur tout type de terminal texte, sur tout UNIX.

Inconvénient :

- il demande de la pratique

Il possède deux modes de fonctionnement :

- un mode de saisie de commandes à appliquer au texte
- un mode de saisie du texte

Cf <http://www.math.fu-berlin.de/~guckes/vi/> pour de la doc.

### ◇ Commande passant en mode saisie de texte

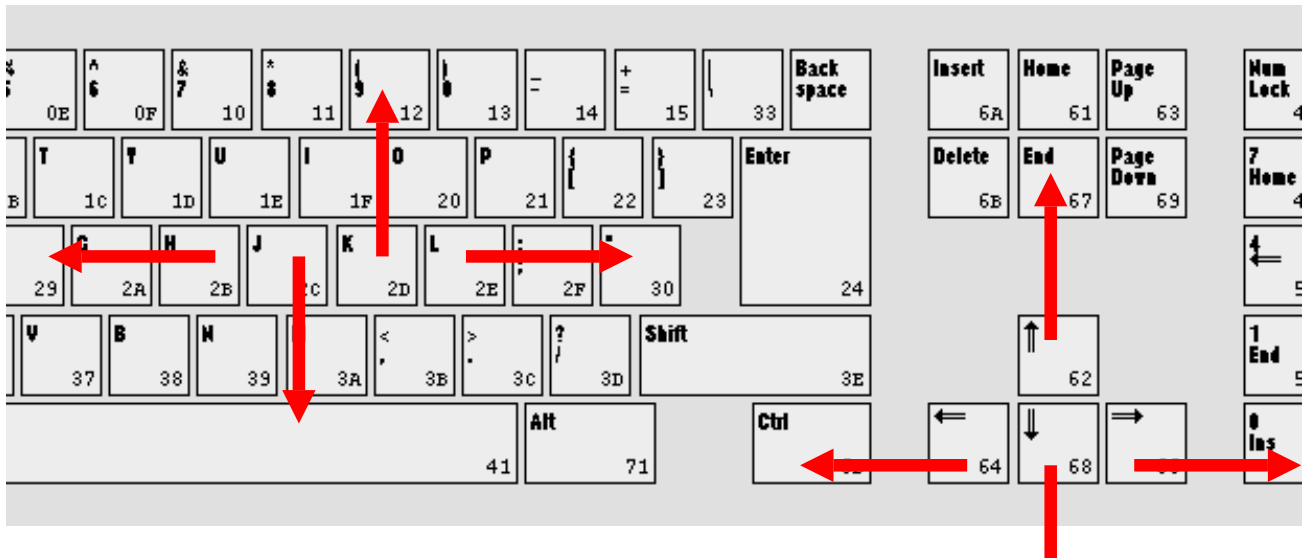
Séquence	Action
i	Insérer à la position courante du curseur (en anglais <i>insert</i> )
a	Insérer à la position suivante du curseur (en anglais <i>append</i> )
I	Insérer en début de ligne
A	Insérer en fin de ligne
o	Ouvrir une nouvelle ligne en dessous du curseur (en anglais <i>open</i> )
O	Ouvrir une nouvelle ligne au dessus du curseur
cw	Changer un mot (en anglais <i>change word</i> )
c\$	Changer jusqu'à la fin de ligne

### ◇ Sortie du mode saisie du texte et passage en mode commandes

On passe du mode saisie de texte au mode commandes par la touche ESC.

## ◇ Commandes de déplacement

Séquence	Action
h ou ←	Déplacer le curseur d'un caractère à gauche
l ou ⇒	Déplacer le curseur d'un caractère à droite
j ou ↓	Déplacer le curseur d'une ligne vers le bas
k ou ↑	Déplacer le curseur d'une ligne vers le haut



## ◇ Commandes de déplacement

Séquence	Action
nombre G	Aller à la ligne «nombre» (en anglais <i>goto</i> )
G	Aller à la dernière ligne
Ctrl-F	Avance d'une page d'écran
Ctrl-B	Reculé d'une page d'écran
Ctrl-G	Affiche le numéro de la ligne courante

◇ Commandes principales

Séquence	Action
x	Détruire le caractère sous le curseur
8x	Détruire 8 caractères
r suivi d'un caractère x	Remplacer le caractère sous le curseur par le caractère X (en anglais <i>replace</i> )
dd	Effacer la ligne courante
d8d	Effacer 8 lignes en comptant la ligne courante
:3,7d	Effacer de la ligne 3 à la ligne 7
:1,\$d	Effacer de la ligne 1 à la dernière ligne
:. ,21d	Effacer de la ligne courante à la ligne 21
dw	Effacer le mot sous le curseur (en anglais <i>delete word</i> )
d8w	Effacer 8 mots
J	Joindre la ligne suivante avec la ligne courante (en anglais <i>join</i> )

Remplacer 8 dans les exemples ci-dessus par le nombre que vous voulez

◇ Commandes principales (2)

Séquence	Action
u	Annuler la dernière commande (en anglais <i>undo</i> )
Ctrl-L	Rafraichir l'écran
.	Répéter la dernière commande
/cerise	Rechercher « cerise » dans le texte vers le bas
?cerise	Rechercher « cerise » dans le texte vers le haut
/regexp	Rechercher la regexp indiquée dans le texte vers le bas (voir page 330)
?regexp	Rechercher la regexp indiquée dans le texte vers le haut (voir page 330)
n	Répéter la dernière recherche (en anglais <i>next</i> )

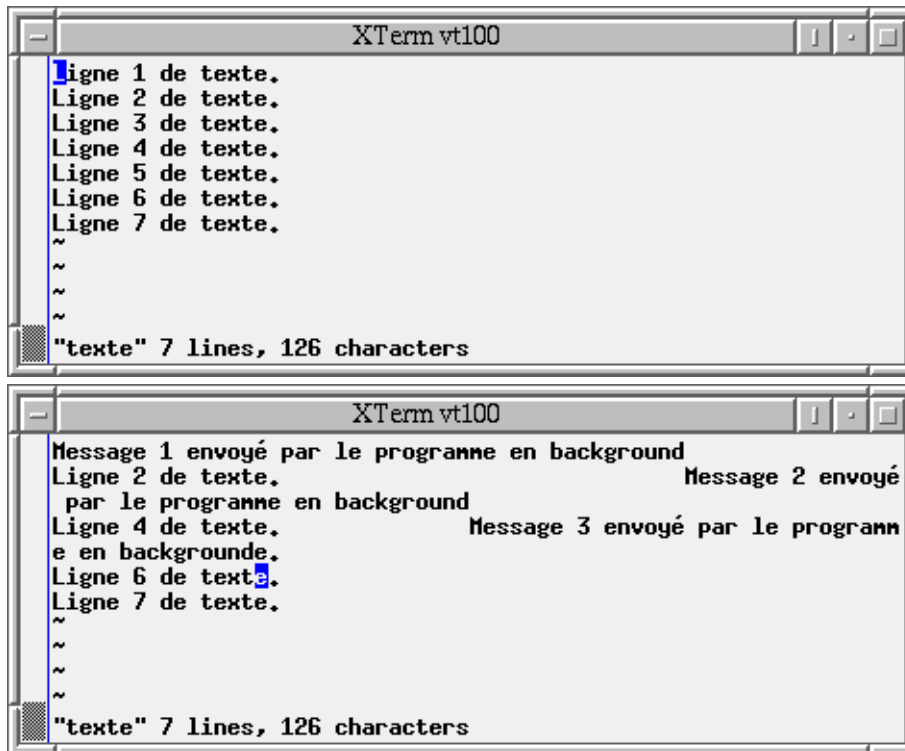
A NOTER : Les commandes commençant par le caractère « : » « apparaissent » en bas de l'écran pour pouvoir lire ce que l'on tape (par exemple un nom de fichier pour sauvegarder).

A NOTER : La commande « . » recommence la dernière commande qui ne commençait pas par « : ».



Exemple d'utilisation du `Ctrl-L` :

Supprimer les affichages parasites par exemple de programmes en tâche de fond (voir page 458).



### ◇ Sauvegarde / Sortie de vi

Séquence	Action
:w	Sauver le fichier édité (en anglais <i>write</i> )
:w ananas	Sauver dans le fichier « ananas »
:q	Quitter vi (en anglais <i>quit</i> )
:q!	Quitter vi sans sauvegarder la moindre chose
:wq	Sauver puis quitter vi (en anglais <i>write + quit</i> )
:e ananas	Editer maintenant le fichier « ananas » (en anglais <i>edit</i> )
:r ananas	Importer le contenu du fichier « ananas » (en anglais <i>read</i> )

### ◇ Commandes de copier/coller

Séquence	Action
yy	Copier la ligne courante dans la mémoire copier/coller (en anglais <i>yank</i> )
p	Coller dans le texte le contenu de la mémoire précédente (en anglais <i>paste</i> )
nombre yy	Copier «nombre» lignes dans la mémoire copier/coller

### ◇ Commandes de substitution

Séquence	Action
:s/ananas/cerise/	Sur la ligne du curseur, remplacer le premier mot « ananas » par « cerise » (en anglais <i>substitute</i> )
:s/ananas/cerise/g	Sur la ligne du curseur, remplacer tous les mots « ananas » par « cerise »
:1,\$s/ananas/cerise/	De la ligne 1 à la dernière ligne (\$), remplacer le premier mot « ananas » par « cerise »
:1,\$s/ananas/cerise/g	De la ligne 1 à la dernière ligne (\$), remplacer tous les mots « ananas » par « cerise »

## Autres exemples de séquences de substitution :

- La séquence « :1,\$s/ananas/cerise/g » remplace de la première ligne à la dernière ligne chaque mot « ananas » par « cerise ».
- La séquence « :%s/ananas/cerise/g » remplace de la première ligne à la dernière ligne chaque mot « ananas » par « cerise ».  
⇒ **On peut employer % à la place de 1, \$.**
- La séquence « :1,\$s//g » remplace de la première ligne à la dernière ligne chaque mot « ananas » par rien du tout, c'est-à-dire que l'on supprime de la première ligne à la dernière ligne chaque mot « ananas »
- La séquence « :1,\$s/\/ananas/cerise/g » remplace de la première ligne à la dernière ligne chaque mot « /ananas » par « cerise ».
- La séquence « :1,\$s;/ananas;cerise;g » remplace de la première ligne à la dernière ligne chaque mot « /ananas » par « cerise ».  
⇒ **On peut employer d'autres caractères de séparation que le caractère « / ».**

- séquence « :1,.s/ananas/cerise/g » remplace de la première ligne à la ligne courante (désignée par « . ») chaque mot « ananas » par « cerise ».
- La séquence « :.,\$s/ananas/cerise/g » remplace de la ligne courante (désignée par « . ») jusqu'à la dernière ligne (désignée par « \$ ») chaque mot « ananas » par « cerise ».
- La séquence « :.,.+3s/ananas/cerise/g » remplace de la ligne courante (désignée par « . ») à 3 lignes plus bas (désignée par .+3) chaque mot « ananas » par « cerise ».
- La séquence « :.-3,.s/ananas/cerise/g » remplace de 3 lignes plus haut que la ligne courante (« .-3 ») à la ligne courante chaque mot « ananas » par « cerise ».

## ◇ Principales options

Séquence	Action
:set all	Afficher toutes les options possibles
:set opt	Positionner l'option « opt » à vrai
:set noopt	Positionner l'option « opt » à faux
:set nu	Afficher les numéros de ligne
:set nonu	ne pas afficher les numéros de ligne

Les options peuvent être enregistrées de façon permanente : les copier dans le fichier « \$HOME/.exrc ». Par exemple :

```
% cat $HOME/.exrc
set nu
```

## ◇ Divers

En cas de plantage de vi, utiliser la commande « vi -r exemple.txt » pour essayer de récupérer ce qui est récupérable (en anglais *recover*).

Pour consulter un fichier sans le modifier, faire « vi -R exemple.txt » (en anglais *readonly*). Ne pas confondre avec au dessus.

La version de « vi » dans les salles de TP de la Formation Permanente fait automatiquement une sauvegarde du fichier texte que l'on veut éditer. La sauvegarde automatique du fichier « exemple.txt » a pour nom « exemple.txt~ ».

(en anglais *vi improved*)

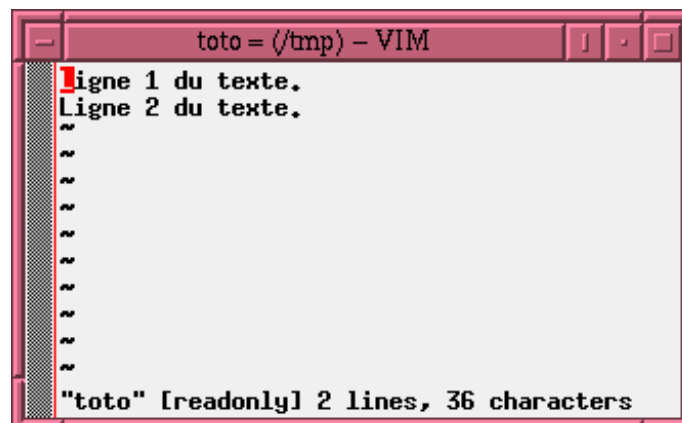
C'est une version améliorée de VI.

Cf <http://www.vim.org>

Intérêt : disponible sur WINDOWS, MACOS

Attention à ne pas vous habituer à des fonctionnalités propres à VIM et non standard dans les autres VI.

La commande « view » lance « vi » en mode readonly.



On a donc : « view exemple.txt » équivalent à « vi -R exemple.txt ».

Commande très pratique.

Comment est codée une fin de ligne dans un fichier texte ?

Pas de standard !

En pratique :

- Sur UNIX : une fin de ligne est codée par le code ASCII 10 (Ctrl-J ; notation du langage C « `\n` »)
- Sur WINDOWS : une fin de ligne est codée par le code ASCII 13 suivi du code ASCII 10 (Ctrl-M Ctrl-J ; notation du langage C « `\r\n` »)
- Sur MACINTOSH : une fin de ligne est codée par le code ASCII 13 (Ctrl-M ; notation du langage C « `\r` »)

Nécessité de savoir convertir.

## Chapitre 6

# *Commandes de manipulation de base d'objets UNIX*

Sur UNIX, plusieurs types d'objets :

- fichiers
- répertoires
- objets associés aux disques durs, CDRoms, bandes magnétiques, etc.
- objets système de communication inter applications
- autres objets systèmes (doors Solaris, etc.)

Ceux que l'on manipule le plus souvent en tant qu'utilisateur :

- fichiers
- répertoires

**Sur UNIX, on fait la différence entre lettres minuscules et lettres majuscules en ce qui concerne les noms d'objets !**

```
% ls -l
total 0
-rw-r--r--  1 besancon ars      0 Oct 16 21:44 EXEMPLE.txt
-rw-r--r--  1 besancon ars      0 Oct 16 21:44 ExEmPlE.txt
-rw-r--r--  1 besancon ars      0 Oct 16 21:44 Exemple.txt
-rw-r--r--  1 besancon ars      0 Oct 16 21:44 eXeMpLe.TXT
-rw-r--r--  1 besancon ars      0 Oct 16 21:44 exemple.TXT
-rw-r--r--  1 besancon ars      0 Oct 16 21:44 exemple.txt
```

**Sur UNIX, on évitera autant que possible les caractères espace, apostrophe, guillemets et les lettres accentuées dans les noms d'objets !**

**Sur UNIX, on préférera nommer les objets avec les lettres minuscules a-z, lettres majuscules A-Z, les chiffres 0-9, le tiret « - », le underscore « \_ », le point « . ».**

**NE PAS UTILISER LE RESTE !**

**Sur UNIX, il y a certaines conventions pour les extensions dans les noms des objets.**

- Extensions pour les langages de programmation : « `programme.c` », « `include.h` »,
- Extensions pour les archives ou les fichiers compressés : « `archive.tar` », « `rapport.gz` »
- pas d'extension pour les fichiers texte  
**Dans ce cours, on clarifiera les choses en utilisant l'extension « `.txt` » du monde Windows quand cela sera plus parlant.**
- pas d'extension pour les fichiers exécutables  
**Dans ce cours, on clarifiera les choses en utilisant l'extension « `.exe` » du monde Windows quand cela sera plus parlant.**
- etc.



Les objets sont manipulables sur le disque dur via l'intermédiaire d'une structure de données appelée « **inode** ».

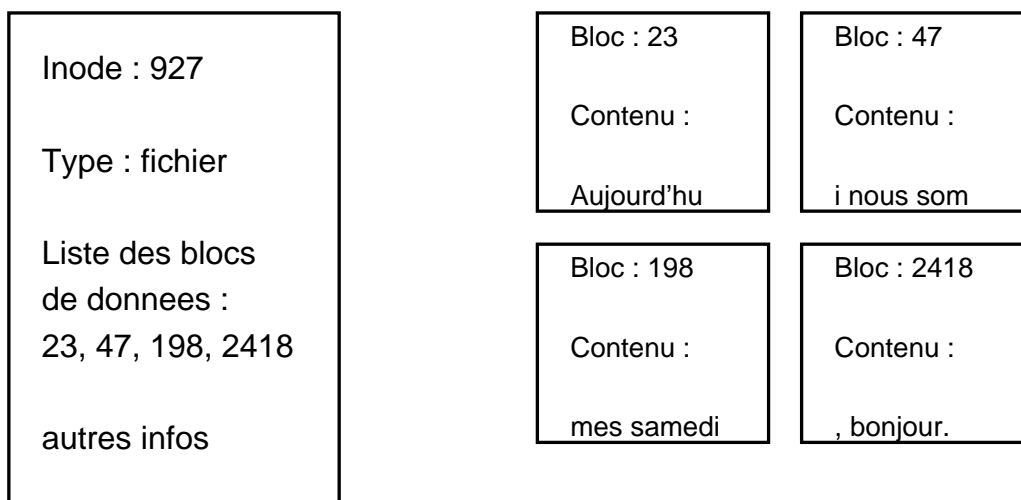
Cela sera revu en détails dans le tome 2.

En gros :

- 1 un inode a un numéro unique
- 2 un inode indique le type de l'objet
- 3 un inode possède la liste des blocs de données de l'objet
- 4 le système UNIX passe son temps à manipuler les inodes

**NOTA BENE : l'inode d'un objet ne stocke pas le nom de l'objet !**

Un fichier correspond à un inode de type fichier :

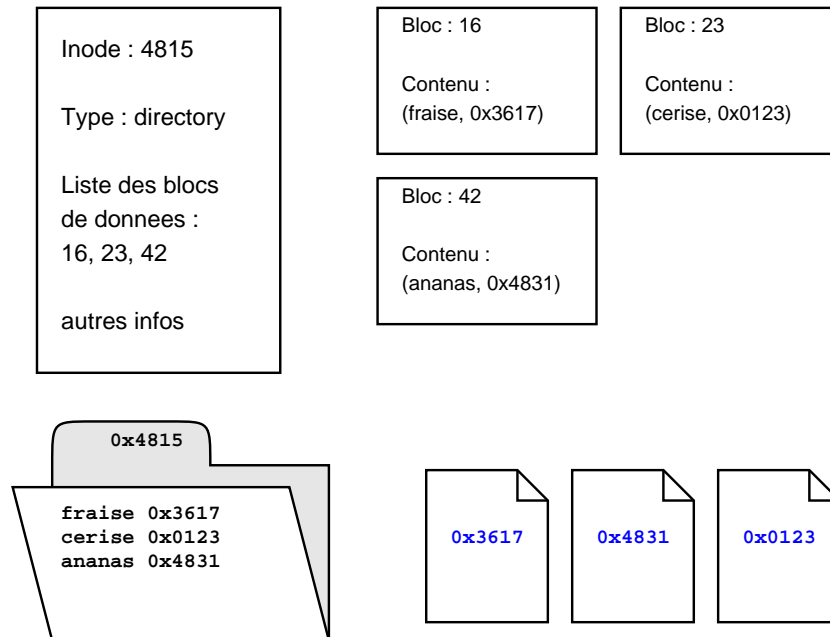


Bloc de meta donnees:  
inode

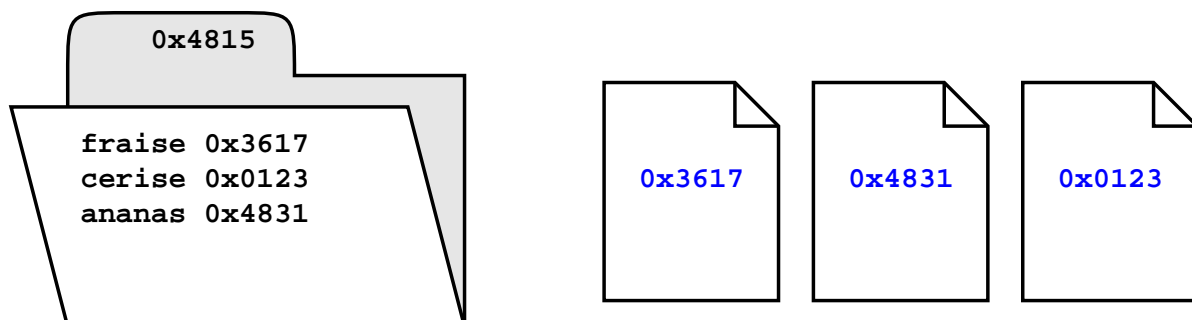
Blocs de donnees

Terminologie : répertoire, dossier, *directory* en anglais

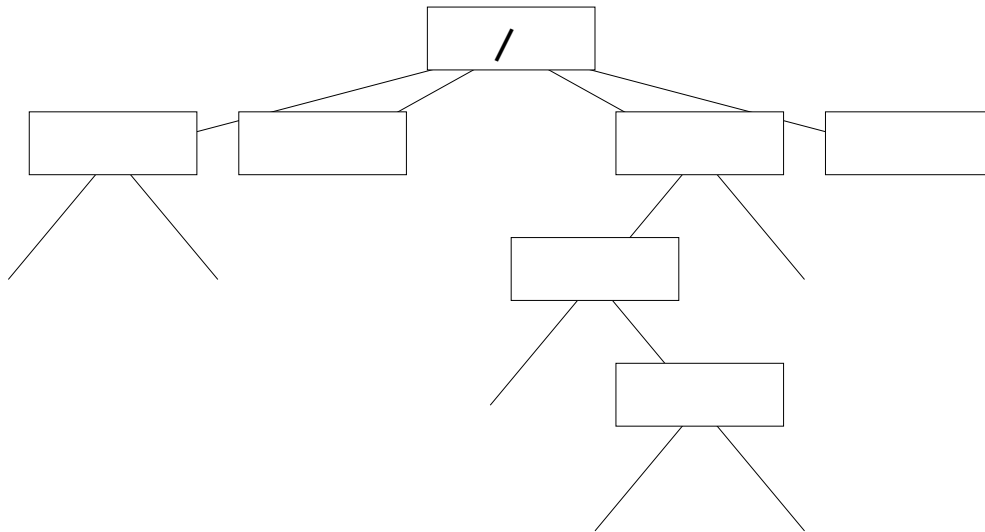
Un répertoire est un « fichier » dont les données sont une liste de noms + numéros d'inode.



C'est le répertoire qui donne un nom à un objet :



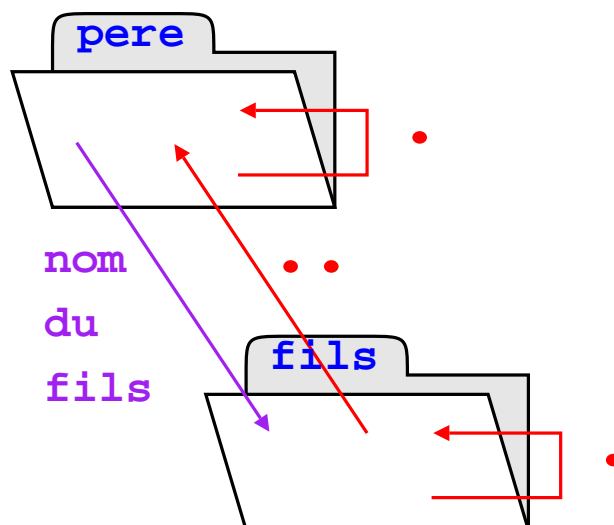
Un répertoire peut renvoyer sur un autre répertoire et ainsi de suite. Cela permet de construire une arborescence représentable par un arbre :



Les objets seront répartis dans l'arborescence.

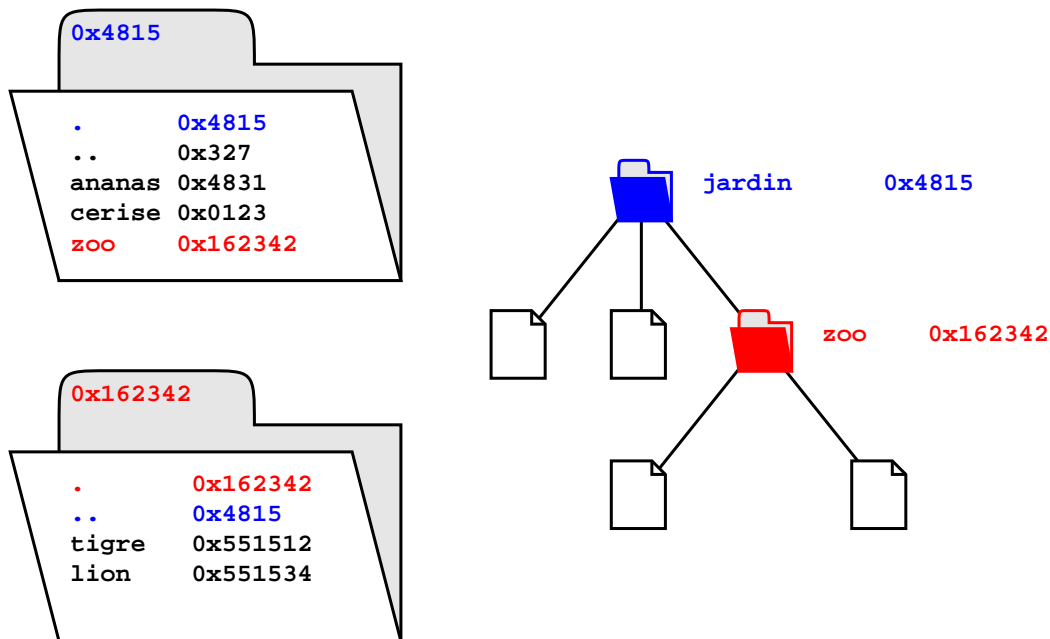
**La racine s'appelle « / », prononcé *slash*.**

Plus exactement les directories sont organisés entre eux de la façon suivante :



UNIX garantit qu'il n'y a pas de boucle dans l'arborescence.

Exemple :



## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

### §6.5 • Notions de chemins absolus et relatifs

Sur UNIX, un objet est manipulable par son chemin dans l'arborescence depuis le point de départ de l'arborescence.

Le chemin est constitué de la liste des noms des répertoires traversés et est terminé par le nom de l'objet en soi.

/ répertoire1 / répertoire 2/ ... / nom

**Point fondamental : utilisation du caractère « / » comme séparateur dans l'énumération des répertoires traversés.**

**Attention : par abus de langage, on confondra l'objet avec le nom de l'objet et avec le chemin d'accès à l'objet.**

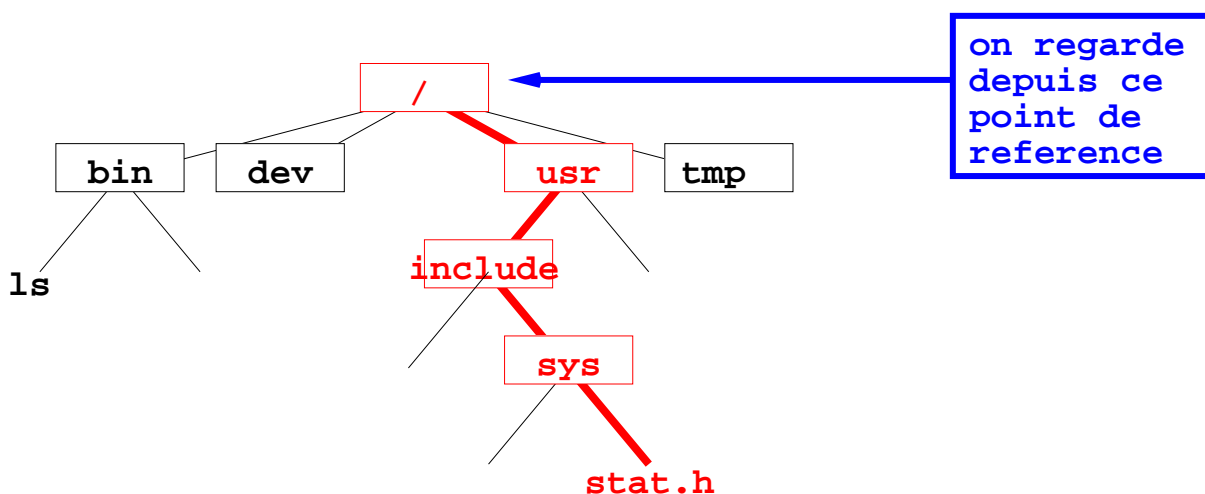
◇ chemin d'accès absolu :

Si le chemin d'accès commence par « / », il s'agit d'un chemin absolu :

chemin d'accès absolu = / répertoire1 / répertoire2 / ... / nom

Un chemin absolu s'exprime par rapport à la racine « / ».

Par exemple : « /usr/include/sys/stat.h »



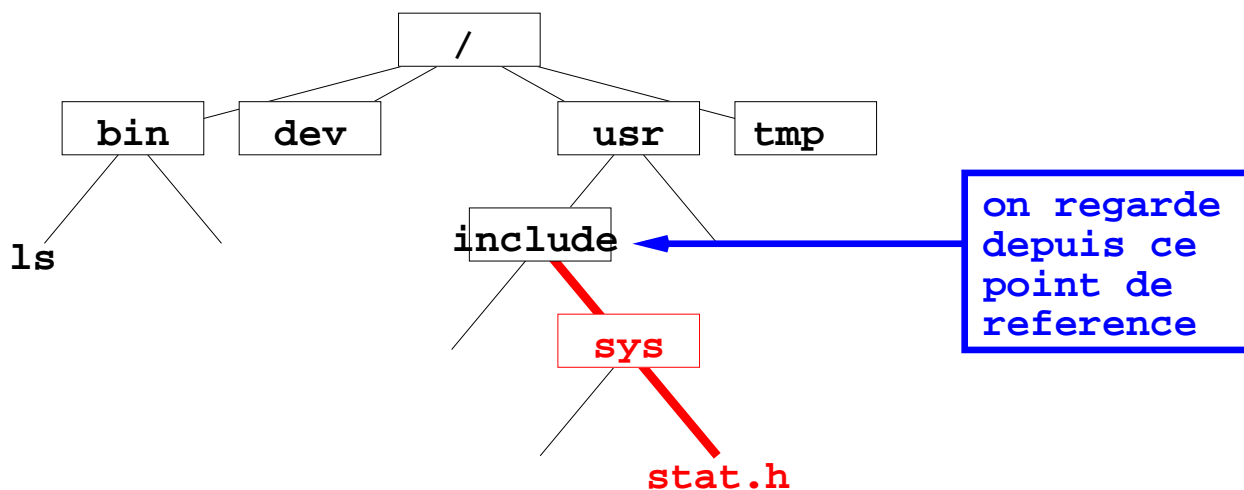
◇ chemin d'accès relatif :

Si le chemin d'accès ne commence pas par « / », il s'agit d'un chemin relatif :

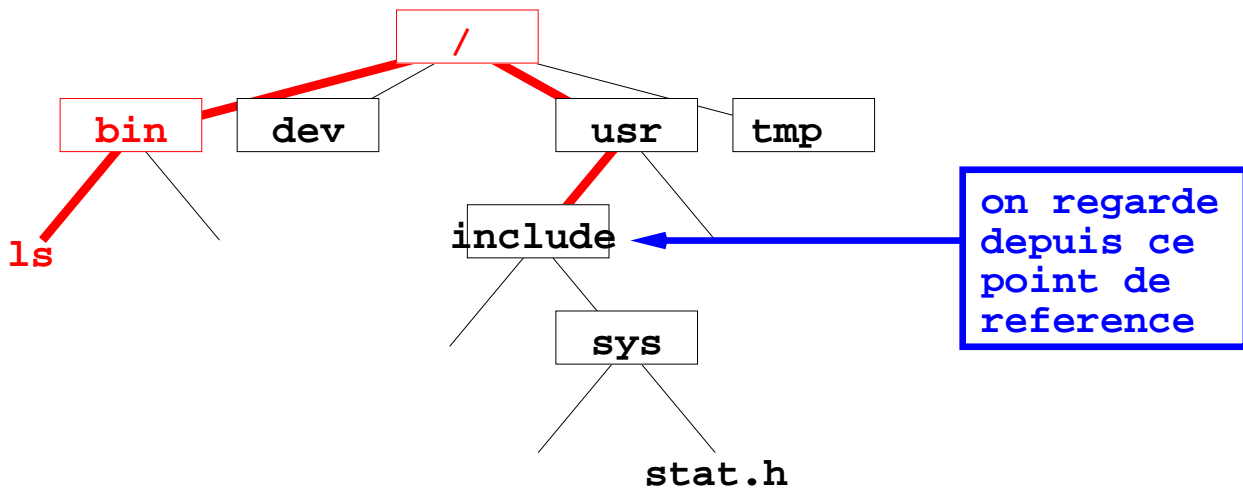
chemin d'accès relatif = répertoire1 / répertoire2 / ... / nom

Un chemin est relatif **par rapport à un point de référence**.

Par exemple, depuis « /usr/include/ », on a le chemin relatif « sys/stat.h » :



Par exemple, depuis « /usr/include/ » on a le chemin relatif  
« ../../bin/ls » :



**Grande importance dans les chemins relatifs des écritures « . » et « .. ».**

Exemples d'utilisation du répertoire courant noté « . » (revus ultérieurement) :

- commande « `find` » pour lancer une recherche à partir de l'endroit courant :

```
find . -name exemple.txt -print
```

- lancer une commande « `commande.exe` » qui se trouve dans le répertoire courant :

```
./commande.exe
```

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.6 • Positionnement dans l'arborescence : cd

(en anglais *change directory*)

Syntaxe : cd répertoire

```
% cd /etc
% cd /usr/include
% cd /inexistant
/inexistant: bad directory
```

Selon le shell, le message d'erreur dans le dernier cas peut changer :

```
% cd /inexistant
bash: /inexistant: No such file or directory
```

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.7 • Position dans l'arborescence : pwd

(en anglais *present working directory*)

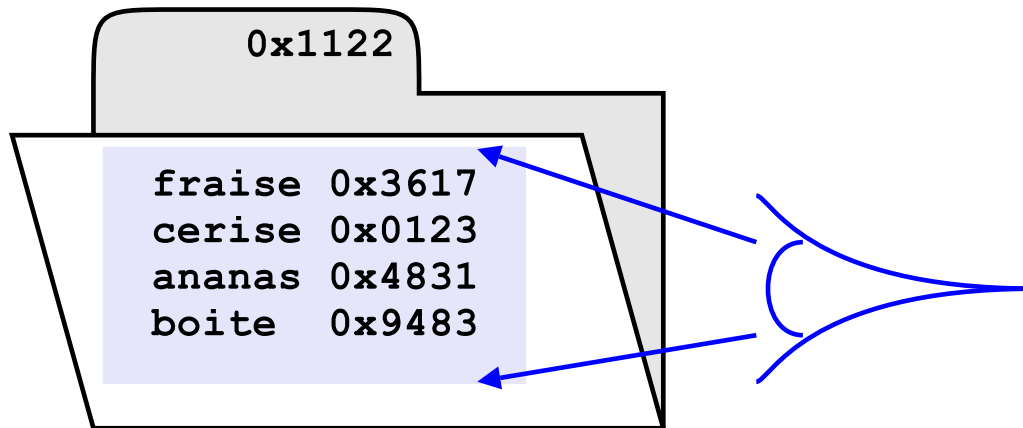
Syntaxe : pwd

```
% cd /etc
% pwd
/etc

% cd /usr/include
% pwd
/usr/include
```



Obtenir une liste d'objets, c'est lire le contenu d'un directory :



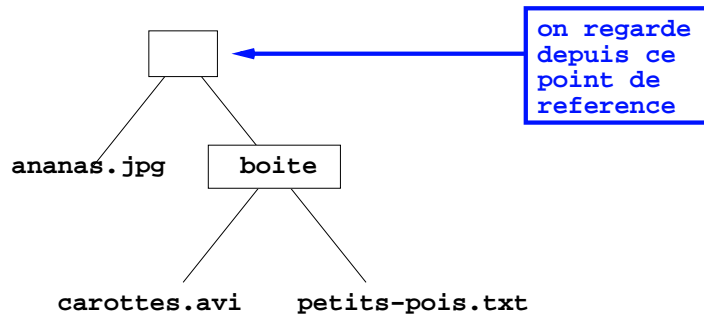
(en anglais *list*)

Syntaxe : `ls [options] objets`

Principales options (cumulables) :

- option « `-l` » : affichage au format long des informations relatives aux objets
- option « `-g` » : affichage des groupes propriétaires des objets
- option « `-R` » : liste récursive des objets indiqués
- option « `-d` » : affichage des noms des objets et non de leurs contenus
- option « `-F` » : affichage des objets avec un suffixe désignant le type de l'objet
- option « `-a` » : affichage des objets dont les noms commencent par « `.` »

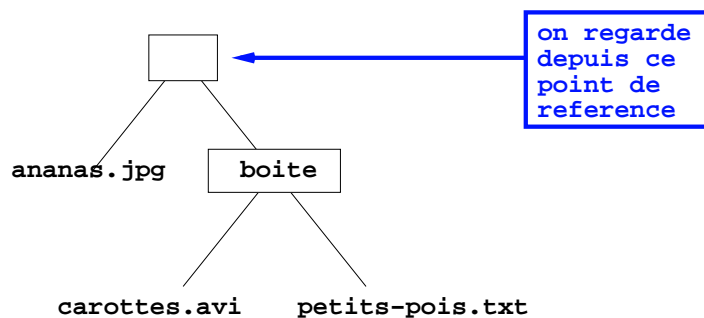
### ◇ Exemple 1 : commande seule



« ls » renvoie la liste des objets :

```
% ls
ananas.jpg  boite
```

### ◇ Exemple 2 : option « -l »



« ls -l » renvoie la liste des objets et de leurs informations :

```
% ls -l
total 6
-rw-r--r--  1 besancon ars      1035 Feb 13 14:55 ananas.jpg
drwxr-xr-x  2 besancon ars        512 May 16 20:18 boite
```

**ATTENTION** : Selon l'âge des objets, l'affichage n'est pas le même !

Objets vieux de moins de 6 mois :

```
% ls -l
total 6
-rw-r--r--    1 besancon ars      3506 Nov 27  2005 ananas.jpg
drwxr-xr-x    2 besancon ars        512 May 16 20:18 jardin
```

Objets vieux de plus de 6 mois : affichage de l'année mais pas de l'heure :

```
% ls -l
total 6
-rw-r--r--    1 besancon ars      3506 Nov 27  2005 ananas.jpg
drwxr-xr-x    2 besancon ars        512 May 16 20:18  jardin
```

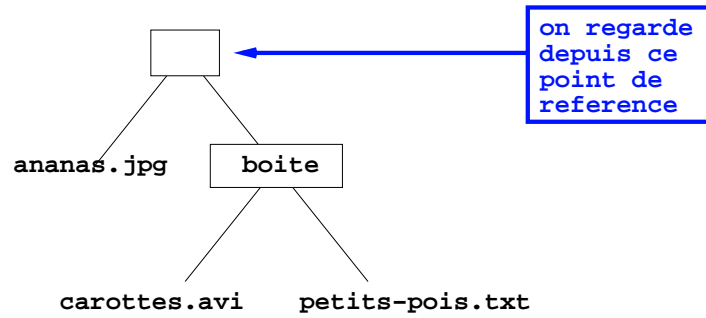
⇒ l'utilisation de « `ls -l` » sera difficile dans des scripts

Solaris : options « `-e` » ou « `-E` »

```
% ls -e poire.txt
-rw-r--r--    1 besancon ars      3506 Nov 27 19:10:37 2005 poire.txt
```

```
% ls -E poire.txt
-rw-r--r--    1 besancon ars      3506 2005-11-27 19:10:37.000000 +0100 poire.txt
```

### ◇ Exemple 3 : option « -R »



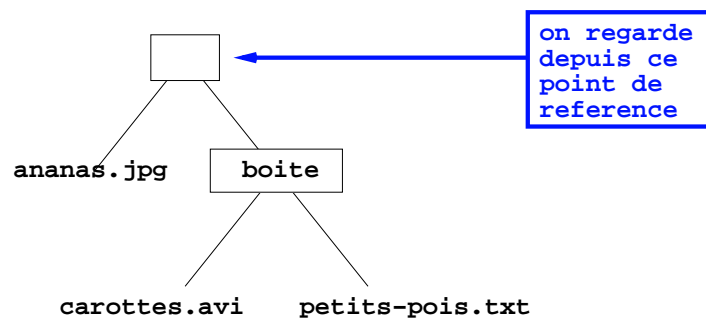
« ls -R » renvoie la liste des objets de la sous-arborescence :

```

% ls -R
.:
ananas.jpg  boite

./boite:
carottes.avi      petits-pois.txt
  
```

### ◇ Exemple 4 : combinaison de l'option « -l » et de l'option « -R »



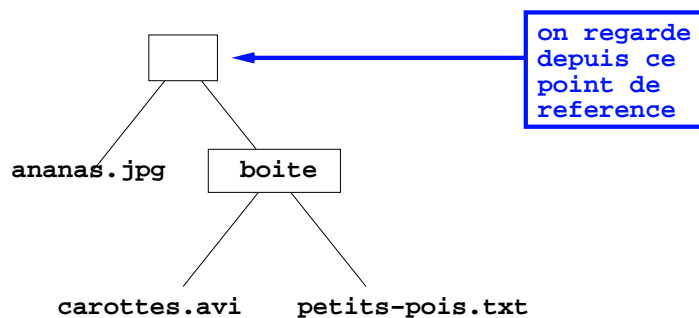
« ls -Rl » renvoie la liste des objets de la sous-arborescence et de leurs informations :

*suite sur transparent suivant*

```
% ls -Rl
.:
total 6
-rw-r--r--  1 besancon ars      1035 Feb 13 14:55 ananas.jpg
drwxr-xr-x  2 besancon ars        512 May 16 20:18 boite

./boite:
total 4
-rw-r--r--  1 besancon ars      315 Dec 14 09:35 carottes.avi
-rw-r--r--  1 besancon ars      613 Jan 23 22:18 petits-pois.txt
```

### ◇ Exemple 5 : option « -F »

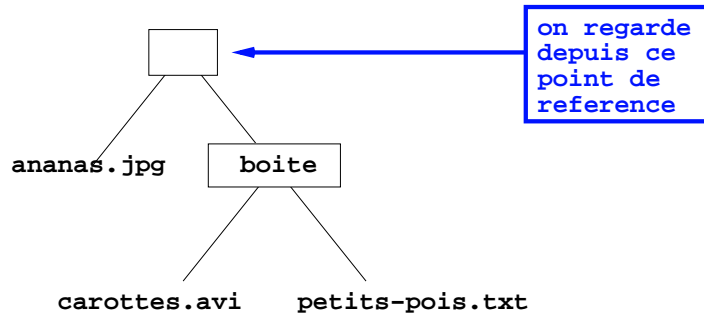


« ls -F » colle au nom de l'objet une indication sur sa nature :

- « / » pour un répertoire
- « \* » pour un exécutable
- « @ » pour un lien symbolique (voir page 171)
- etc.

```
% ls -F
ananas.jpg  boite/
```

## ◇ Exemple 6 : combinaison de l'option « -F » et de l'option « -l »



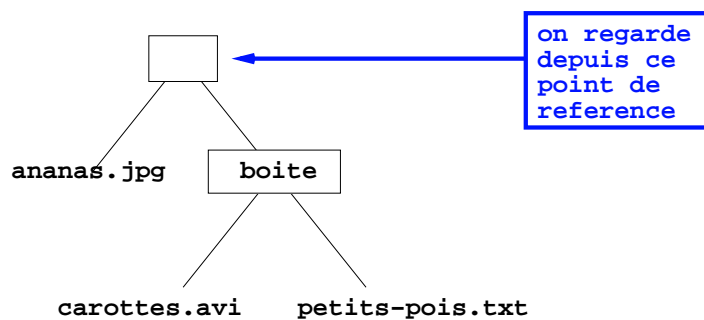
« `ls -lF` » colle au nom de l'objet une indication sur sa nature :

- « / » pour un répertoire
- « \* » pour un exécutable
- « @ » pour un lien symbolique (voir page 171)
- etc.

```

% ls -lF
total 6
-rw-r--r--  1 besancon ars      1035 Feb 13 14:55 ananas.jpg
drwxr-xr-x  2 besancon ars       512 May 16 20:18 boite/
  
```

## ◇ Exemple 7 : option « -a »



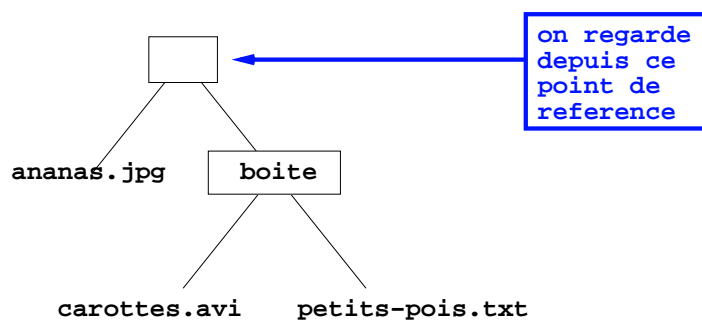
```

% ls -a
.      ..      ananas.jpg  boite
  
```

```
% cd $HOME
% ls -aF
./                .dia/            .kshrc*          .qt/
../              .dt/            .less            .rhosts
.ICAClient/     .dtprofile*    .lessrc          .shosts
.TTauthority    .exerc          .mailcap         .signature
.Xauthority     .fetchmail.pid .mailrc           .ssh/
.Xdefaults      .fetchmailrc   .mime.types      .sunw/
.Xresources     .fonts.cache-1 .mozilla/        .sversionrc*
.acrobat/       .foprc         .mpdefaults      .tcshrc@
.adobe/         .gimp-2.0/     .mushuser        .xine/
.antiword/     .gnome/        .mysql_history   .xinitrc@
.bash_history   .gnome2/      .netscape/     .xnvviewrc
.bash_login     .gphoto/      .plan            .xserverrc*
.bash_logout   .hushlogin/   .profile*       .xsession@
.bashrc        .ispell       .project         ananas.txt
.dbxinit       .java/        .psql_history    cerise.txt
```

Par défaut, la commande « ls » n'affiche pas les noms de objets commençant par « . » qui par convention sont des fichiers de configuration d'utilitaires.

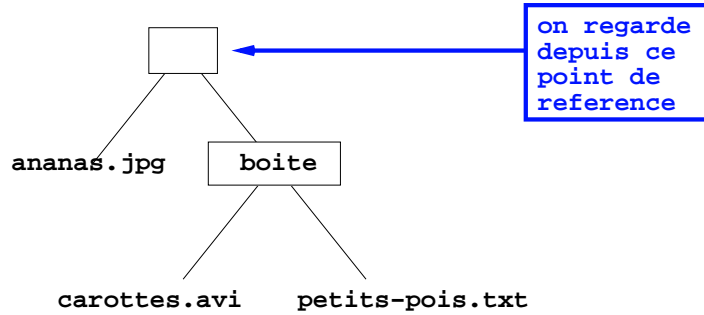
### ◇ Exemple 8 : différence entre contenant et contenu



Affichage du contenu :

```
% ls boite
carottes.avi      petits-pois.txt
```

```
% ls -l boite
total 4
-rw-r--r--  1 besancon ars      315 Dec 14 09:35 carottes.avi
-rw-r--r--  1 besancon ars      613 Jan 23 22:18 petits-pois.txt
```



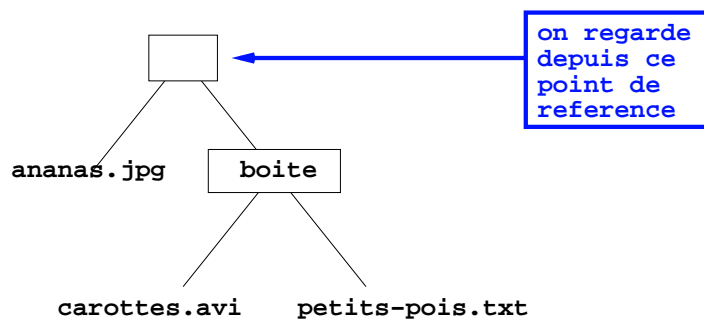
on regarde  
depuis ce  
point de  
reference

Affichage du contenant : utiliser l'option « -d » pour cela :

```
% ls -d boite
boite
```

Plus utile : combinaison de l'option « -d » et de l'option « -l » :

```
% ls -ld boite
drwxr-xr-x  2 besancon ars      512 May 16 20:18 boite
```



on regarde  
depuis ce  
point de  
reference

Affichage du contenant (suite) :

« ls » (sans option) renvoie la liste des objets **contenus dans le répertoire courant** :

```
% ls
ananas.jpg  boite
```

équivalent à « ls . » :

```
% ls .
ananas.jpg  boite
```



## Commande « dir.exe »

Commande UNIX	Commande Windows
<code>ls -C</code>	<code>dir.exe /d</code>
<code>ls -lR</code>	<code>dir.exe /s</code>
sans équivalence (pas de 8x3)	<code>dir.exe /x</code>
<code>ls -l</code>	<code>dir.exe /q</code>
<code>ls -l   more</code>	<code>dir.exe /p</code>

(en anglais *make directory*)

Syntaxe : `mkdir [options] répertoires`

```
% mkdir jardin
% ls -ld jardin
drwxr-xr-x  2 besancon ars          512 May 25 23:46 jardin
```

Taille minimale d'un répertoire (même vide) : 512 octets

Raison : 512 octets  $\equiv$  allocation minimale par le système pour cette catégorie d'objets (la taille sera toujours un multiple de 512)

Pour créer des répertoires emboîtés :

```
% mkdir repertoire1
% mkdir repertoire1/repertoire2
% mkdir repertoire1/repertoire2/repertoire3
```

Pas pratique !

Plus pratique : création directe de sous répertoires en cascade possible via option « -p » :

```
% mkdir -p repertoire1/repertoire2/repertoire3
% ls -R
.:
repertoire1

./repertoire1:
repertoire2

./repertoire1/repertoire2:
repertoire3

./repertoire1/repertoire2/repertoire3:
```

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

### §6.11 • (Windows : : création de répertoires : md.exe, mkdir.exe)

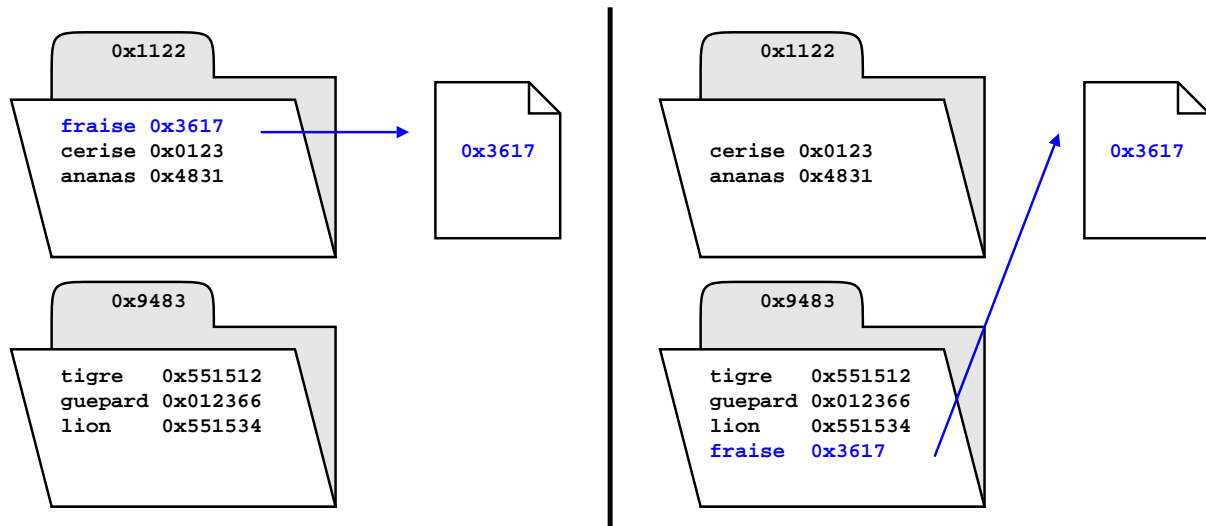
Commande « md.exe » ou « mkdir.exe ».

Commande UNIX	Commande Windows
mkdir dossier	md.exe dossier
mkdir dossier	mkdir.exe dossier

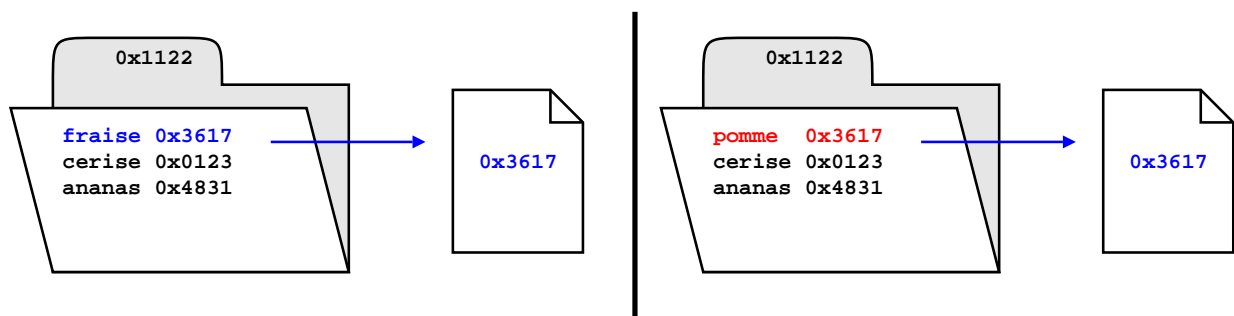
## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

## §6.12 • Déplacer et renommer des objets : mv

Déplacer un objet est le rattacher ailleurs dans l'arborescence à un autre répertoire :



Renommer un objet est changer son rattachement dans le répertoire :



UNIX fournit une seule commande pour ces opérations.

La commande permet à la fois :

- de changer le rattachement d'un objet
- de changer le nom du rattachement

(en anglais *move*)

Syntaxe générale :

```
mv [options] objets objet
```

Quelques options :

- « `-i` » : demande de confirmation à chaque écrasement d'objet

### ◇ Renommage d'objet :

Soit :

```
% ls -l
total 10
-rw-r--r--  1 besancon ars      1035 May 25 22:59 ananas.avi
-rw-r--r--  1 besancon ars      2893 May 25 22:59 banane.txt
```

On fait :

```
% mv ananas.avi film.avi
```

L'objet « `film.avi` » n'existait pas avant. Maintenant on a :

```
% ls -l
total 10
-rw-r--r--  1 besancon ars      2893 May 25 22:59 banane.txt
-rw-r--r--  1 besancon ars      1035 May 25 22:59 film.avi
```

◇ Déplacement d'objets :

```
% ls -l
total 12
-rw-r--r--  1 besancon ars      2893 May 25 22:59 banane.txt
drwxr-xr-x  2 besancon ars       512 May 25 23:03 cinematheque
-rw-r--r--  1 besancon ars     1035 May 25 22:59 film.avi
% mv film.avi cinematheque
% ls -l
total 8
-rw-r--r--  1 besancon ars      2893 May 25 22:59 banane.txt
drwxr-xr-x  2 besancon ars       512 May 25 23:05 cinematheque
% ls -lR
.:
total 8
-rw-r--r--  1 besancon ars      2893 May 25 22:59 banane.txt
drwxr-xr-x  2 besancon ars       512 May 25 23:05 cinematheque

./cinematheque:
total 4
-rw-r--r--  1 besancon ars     1035 May 25 22:59 film.avi
```

◇ Confirmation avec écrasement :

```
% ls -l
total 10
-rw-r--r--  1 besancon ars     1035 May 25 22:59 film.avi
-rw-r--r--  1 besancon ars     2893 May 25 23:16 mummy.avi

% mv -i film.avi mummy.avi
mv: overwrite mummy.avi (yes/no)? y

% ls -l
total 4
-rw-r--r--  1 besancon ars     1035 May 25 22:59 mummy.avi
```

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.13 • (Windows : : déplacer des objets : move.exe)

Commande « move.exe »

Commande UNIX	Commande Windows
mv objet dossier	move.exe [/Y   /-Y] [lecteur:][chemin]
mv dossier1 dossier2	mkdir.exe [/Y   /-Y] [lecteur:][chemin]

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.14 • (Windows : : renommer des objets : ren.exe, rename.exe)

Commande « ren.exe »

Commande « rename.exe »

Commande UNIX	Commande Windows
mv objet1 objet2	rename.exe [lecteur:][chemin]nom_de_fichi
mv objet1 objet2	ren.exe [lecteur:][chemin]nom_de_fichier1

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

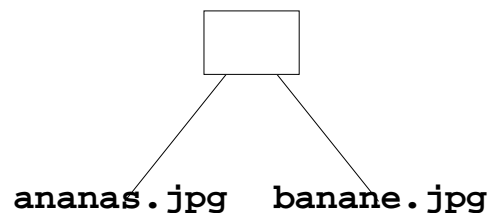
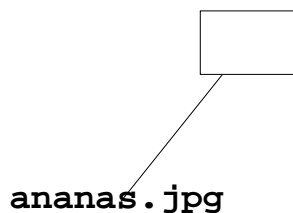
(en anglais *copy*)

Plusieurs syntaxes possibles :

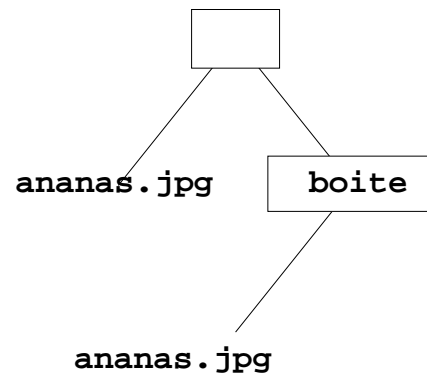
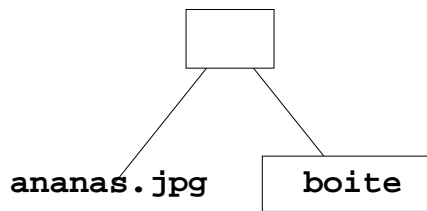
- 1 `cp [options] fichier1 fichier2`  
dupliquer l'objet de départ sous le nom de destination
- 2 `cp [options] fichiers dossier`  
dupliquer les fichiers dans le dossier indiqué
- 3 `cp -r [options] dossiers dossier`  
dupliquer les dossiers dans le dossier indiqué

Quelques options :

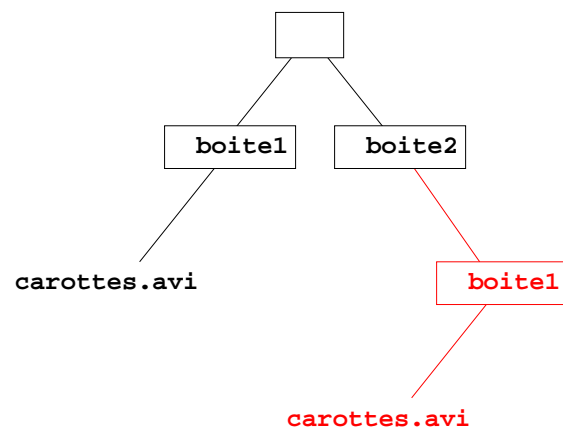
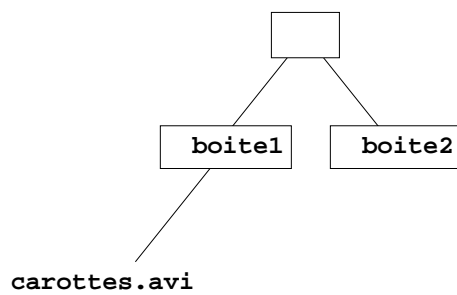
- « `-r` » : copie récursive
- « `-i` » : confirmation à chaque écrasement de fichier
- « `-p` » : conservation des dates (et des propriétaires utilisateur et groupe si commande lancée par l'administrateur)



```
% cp ananas.jpg banane.jpg
```



```
% cp ananas.jpg boite
```



```
% cp -r boite1 boite2
```



## La copie ne conserve pas les dates

```
% cp /etc/motd exemple.txt
% ls -l /etc/motd exemple.txt
-rw-r--r--  1 root      sys          49 Apr  7  2002 /etc/motd
-rw-r--r--  1 besancon ars          49 Jul  6 19:11 exemple.txt
```

⇒ option « -p » pour conserver les dates pendant la copie

```
% cp -p /etc/motd exemple.txt
% ls -l /etc/motd exemple.txt
-rw-r--r--  1 root      sys          49 Apr  7  2002 /etc/motd
-rw-r--r--  1 besancon ars          49 Apr  7  2002 exemple.txt
```

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

### §6.16 • (Windows : : Duplication d'un objet : copy.exe)

## Commande « copy.exe »

```
C:\>copy.exe /?
```

Copie un ou plusieurs fichiers sur un autre emplacement.

```
COPY [/V] [/N] [/Y | /-Y] [/Z] [/A | /B ] source [/A | /B]
      [+ source [/A | /B] [+ ...]] [cible [/A | /B]]
```

## Pas de copie récursive.

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.17 • (Windows : : Duplication d'un objet : xcopy.exe)

## Commande « xcopy.exe »

C:\&gt;xcopy.exe /?

Copie des fichiers et des arborescences de répertoires.

```
XCOPY source [destination] [/A | /M] [/D[:date]] [/P] [/S [/E]] [/V] [/W]
[/C] [/I] [/Q] [/F] [/L] [/H] [/R] [/T] [/U]
[/K] [/N] [/O] [/X] [/Y] [/Y] [/Z]
[/EXCLUDE:fich1[+fich2][+fich3]...]
```

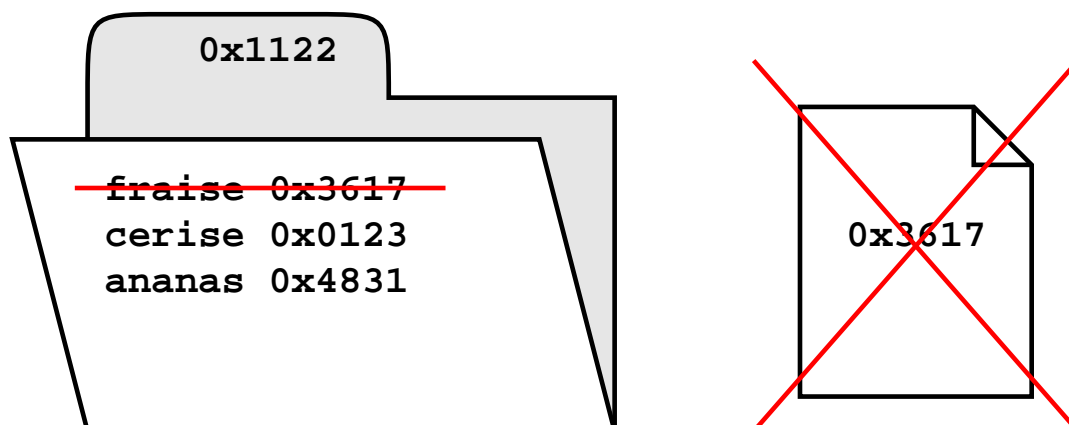
## Utilisation pratique :

- créer la destination : « md.exe dest »
- copie par : « xcopy \*.txt dest /e /c /h /k /o »

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.18 • Destruction d'un objet : rm

Détruire un objet, c'est le supprimer du directory qui l'associe à l'inode :



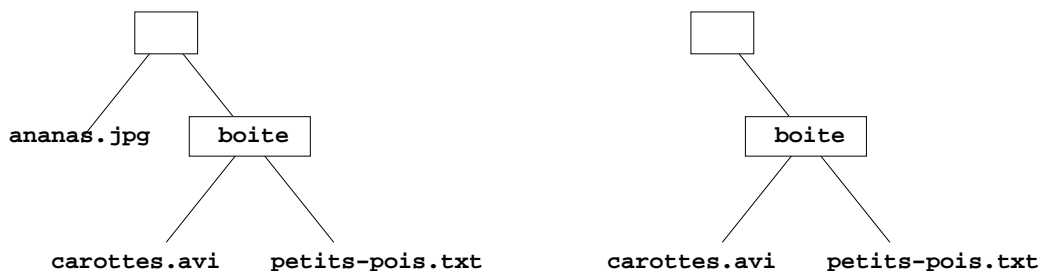
(en anglais *remove*)

Syntaxe : `rm [options] objets`

Quelques options :

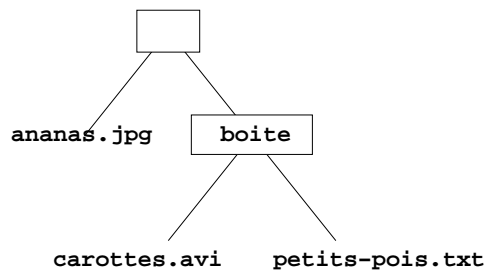
- option « `-i` » : confirmation à chaque suppression (garde fou)
- option « `-r` » : suppression récursive
- option « `-f` » : suppression en force d'un objet même si ses droits ne s'y prêtent pas

« `rm -rf répertoires` » permet de supprimer récursivement toute une arborescence sans demande de confirmation. Attention : dangereux.



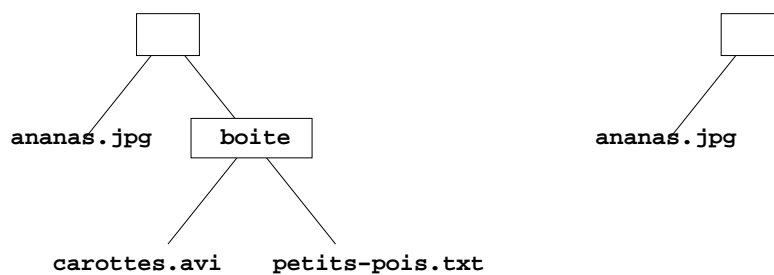
```
% ls
ananas.jpg  boite
```

```
% rm ananas.jpg
% ls
boite
```

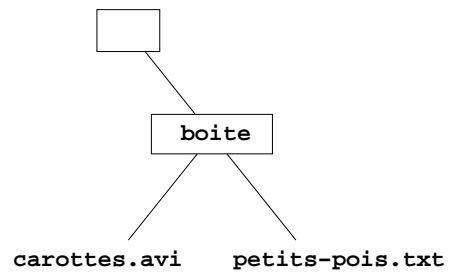
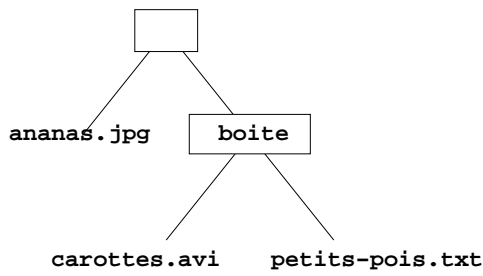


```
% rm boite
rm: boite is a directory

% rmdir boite
rmdir: boite: Directory not empty
% ls boite
carottes.avi      petits-pois.txt
```



```
% rm -rf boite
% ls
ananas.jpg
```



```

% rm -i ananas.jpg
rm: remove ananas.jpg? y
% ls
boite
  
```

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

### §6.19 • (Windows : : Destruction d'un objet : `del.exe`)

Commande UNIX	Commande Windows
<code>rm -i</code>	<code>del.exe /p</code>
<code>rm -r</code>	<code>del.exe /s</code>
<code>rm -f</code>	<code>del.exe /f</code>

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.20 • Suppression de répertoires : `rmdir`

(en anglais *remove directory*)

Syntaxe : `rmdir répertoires`

On ne peut effacer avec cette commande qu'un répertoire vide.

⇒ pénalisant

⇒ on préférera souvent la commande « `rm -rf` »

```
% cp /etc/motd dossier/fichier.txt
% rmdir dossier
rmdir: dossier: Directory not empty
```

Pas d'options dignes d'intérêt.

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

§6.21 • (Windows : : suppression d'un répertoire : `rd.exe`)

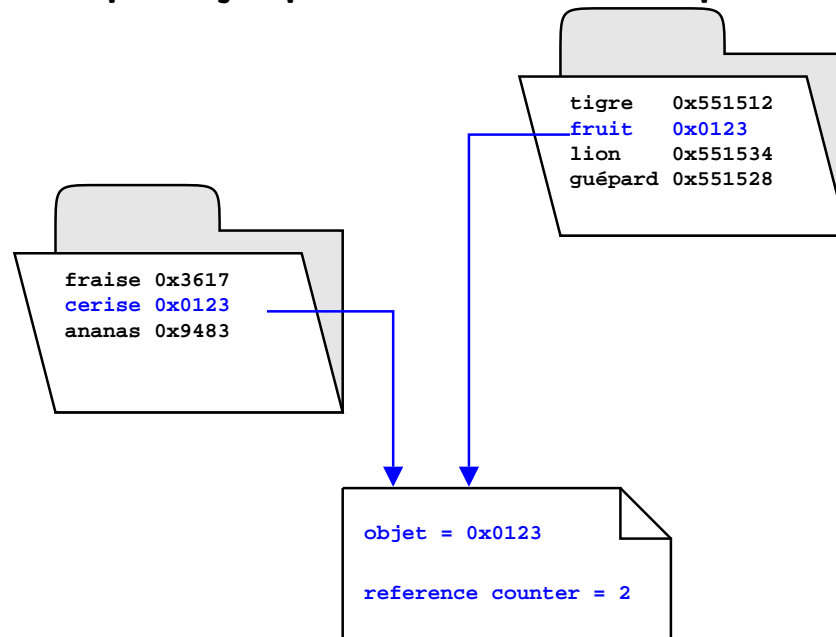
Commande « `rd.exe` »

Commande UNIX	Commande Windows
Pas d'équivalent car demande de confirmation inexistante sur UNIX	<code>rd rep</code>
<code>rmdir rep</code>	<code>rd /q rep</code>
<code>rm -rf rep</code>	<code>rd /s /q rep</code>

**Rappel : les noms sont stockés dans les répertoires.**

**Un nom est appelé un lien sur l'objet.**

**Sur UNIX à chaque objet peuvent être associés plusieurs noms.**



Dans l'inode d'un objet, il y a un compteur de liens :

- compteur incrémenté lors de la création d'un nouveau lien
- compteur décrémenté lors de la suppression d'un lien
- l'objet est détruit lorsque le dernier lien sur l'objet est supprimé

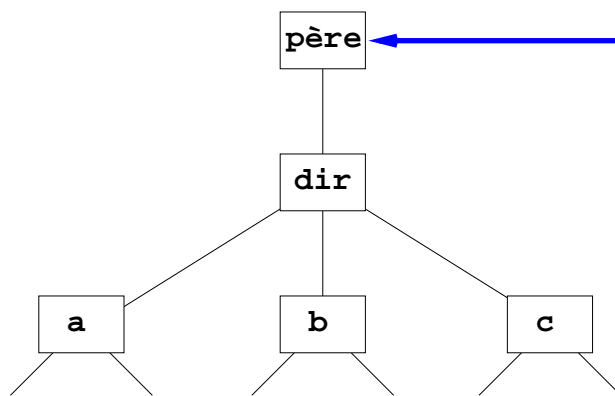
On voit les valeurs des compteurs de liens via la commande « `ls -l` » :

```
% ls -l
-rw-r--r--  1 besancon ars      39 Oct 26  2003 ananas
-rw-r--r--  1 besancon ars      35 Jul  3 17:38 banane
drwxr-xr-x  4 besancon ars    512 Jul  4 15:48 cerise
drwxr-xr-x  2 besancon ars    512 Nov 20  2003 endive
drwxr-xr-x  3 besancon ars    512 Jul  5 00:36 fraise
```

**Rappel :**

- « `.` » est un lien
- « `..` » est un lien

Soit l'arborescence :



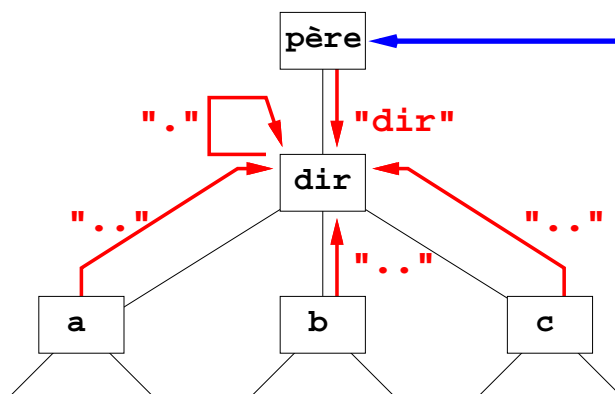
on regarde  
depuis ce  
point de  
référence

et l'affichage :

```
% ls -l
drwxr-xr-x  5 besancon ars          512 Jul  5 00:29 dir
```

Pourquoi a-t-on l'indication de 5 liens sur « dir » ?

Il y a en effet 5 liens sur l'objet nommé « dir » :



on regarde  
depuis ce  
point de  
référence

Ces 5 noms sont :

- 1 lien « /chemin/vers/dir »
- 2 lien « /chemin/vers/dir/. »
- 3 lien « /chemin/vers/dir/a/.. »
- 4 lien « /chemin/vers/dir/b/.. »
- 5 lien « /chemin/vers/dir/c/.. »



Preuve via l'utilisation de l'option « `-i` » de « `ls` » qui affiche les numéros d'inodes :

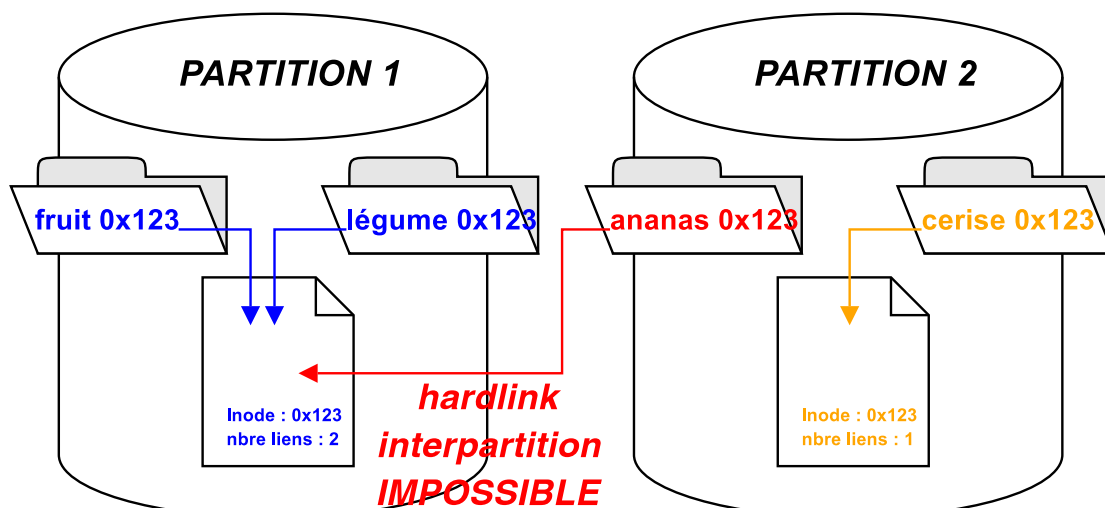
```
% ls -ldi dir dir/. dir/a/.. dir/b/.. dir/c/..
550907 drwxr-xr-x 5 besancon ars      512 Jul  5 00:29 dir
550907 drwxr-xr-x 5 besancon ars      512 Jul  5 00:29 dir/.
550907 drwxr-xr-x 5 besancon ars      512 Jul  5 00:29 dir/a/..
550907 drwxr-xr-x 5 besancon ars      512 Jul  5 00:29 dir/b/..
550907 drwxr-xr-x 5 besancon ars      512 Jul  5 00:29 dir/c/..
```

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

### §6.23 • Lien hard sur objets : 1n

(en anglais *link*)

Le lien **hard** utilise le numéro d'inode pour trouver l'objet. Le numéro est unique par partition. ⇒ un lien hard reste interne à une partition.



Les interdits :

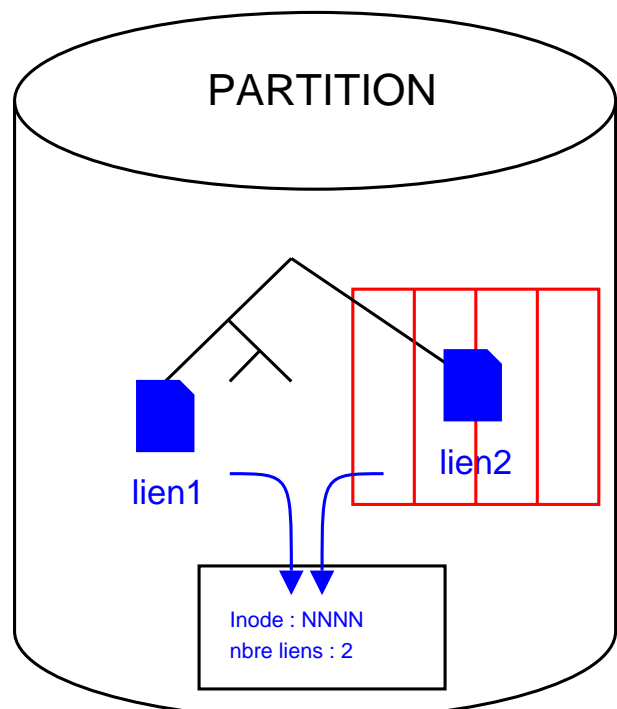
- on ne peut pas faire de hard link vers une autre partition (car impossibilité d'adresser l'inode d'une autre partition depuis un répertoire)
- on ne peut pas faire de hard link vers un répertoire (car sinon boucles invisibles impossibles à détecter dans l'arborescence)

A quoi sert un lien hard ?

Exemple : environnement chrooté (sera revu plus tard)

**Principe du chroot** : il restreint l'accès au contenu d'une partie d'arborescence (dite la *cage*) et on ne peut pas accéder au contenu extérieur de la cage

**Pourquoi utiliser un lien hard ?** : avec un lien hard, un objet peut être à l'intérieur et à l'extérieur de la cage du chroot selon le lien utilisé



(en anglais *link*)

La commande à utiliser est : ln original synonyme

```
% ls -l ananas.jpg
-rw-r--r--  1 besancon ars  9919 Jul 14 10:15 ananas.jpg

% ln ananas.jpg fruit.jpg

% ls -l ananas.jpg fruit.jpg
-rw-r--r--  2 besancon ars  9919 Jul 14 10:15 ananas.jpg
-rw-r--r--  2 besancon ars  9919 Jul 14 10:15 fruit.jpg

% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 ananas.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 fruit.jpg
```

Suppression d'un lien hard par la commande « rm »

```
% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 ananas.jpg
357 -rw-r--r--  2 besancon ars  9919 Jul 14 10:15 fruit.jpg

% rm ananas.jpg

% ls -li fruit.jpg
357 -rw-r--r--  1 besancon ars  9919 Jul 14 10:15 fruit.jpg
```

## Place occupée

```
% ls -li ananas.jpg fruit.jpg
total 8
357 -rw-r--r--  2 besancon ars      9919 Jul 14 10:15 ananas.jpg
357 -rw-r--r--  2 besancon ars      9919 Jul 14 10:15 fruit.jpg

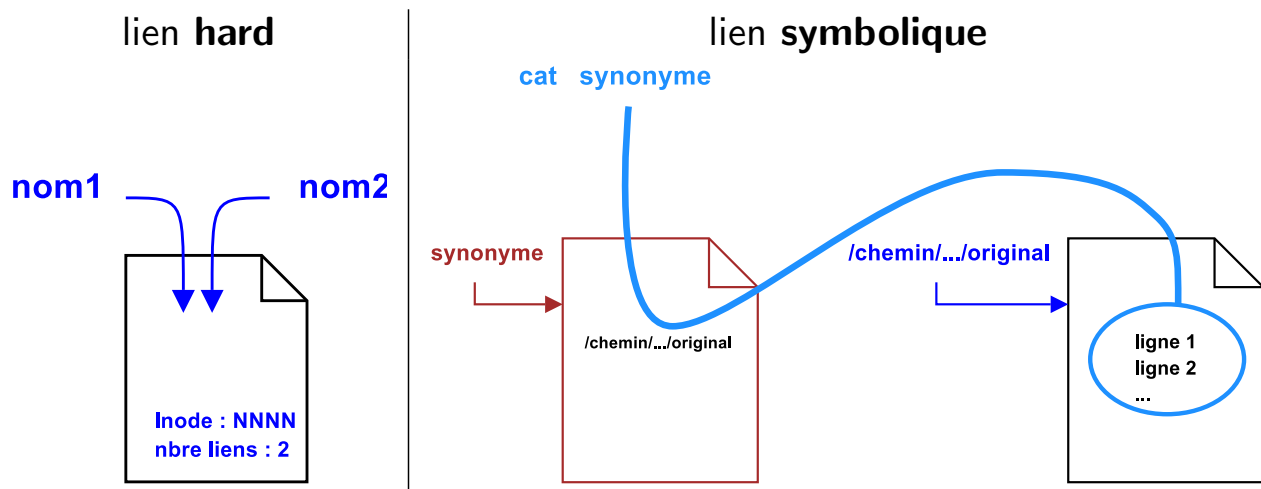
% du -k .
3      .
% rm fruit.jpg
% du -k .
3      .
```

En aucune façon, on ne double la place consommée !

## Chapitre 6 • Commandes de manipulation de base d'objets UNIX

### §6.24 • Lien symbolique sur objets : ln -s

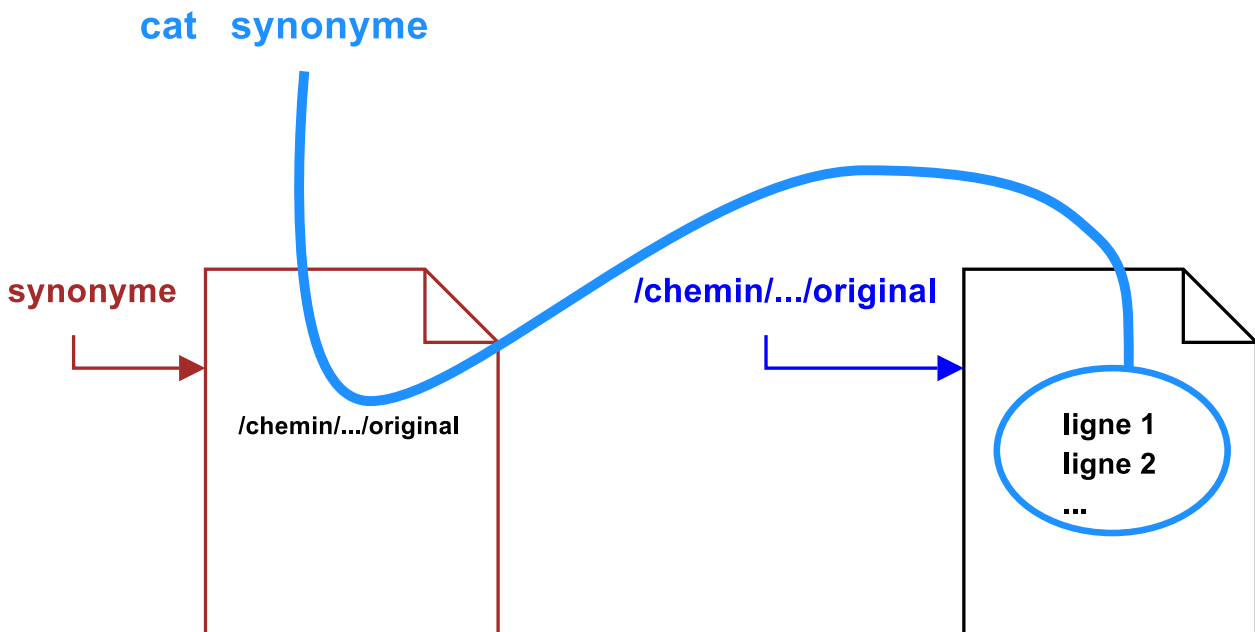
ATTENTION : le mot lien a deux sens sur UNIX :



ATTENTION : ce sont des notions différentes !

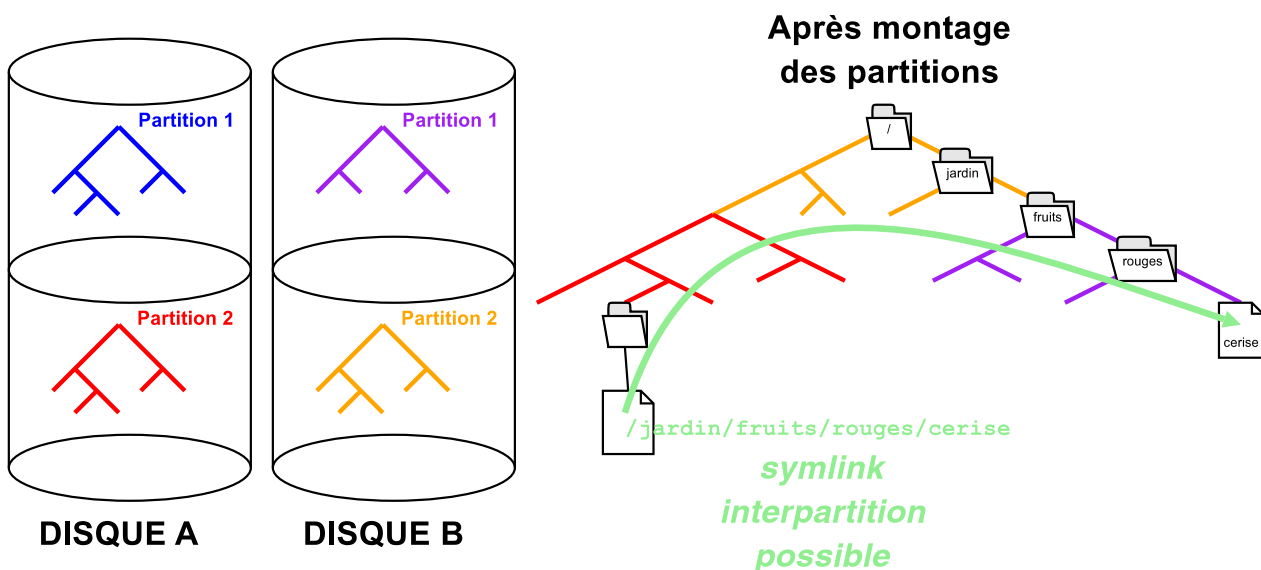
(en anglais *symbolic link*)

Un lien **symbolique** est un fichier spécial contenant le chemin d'un autre objet.



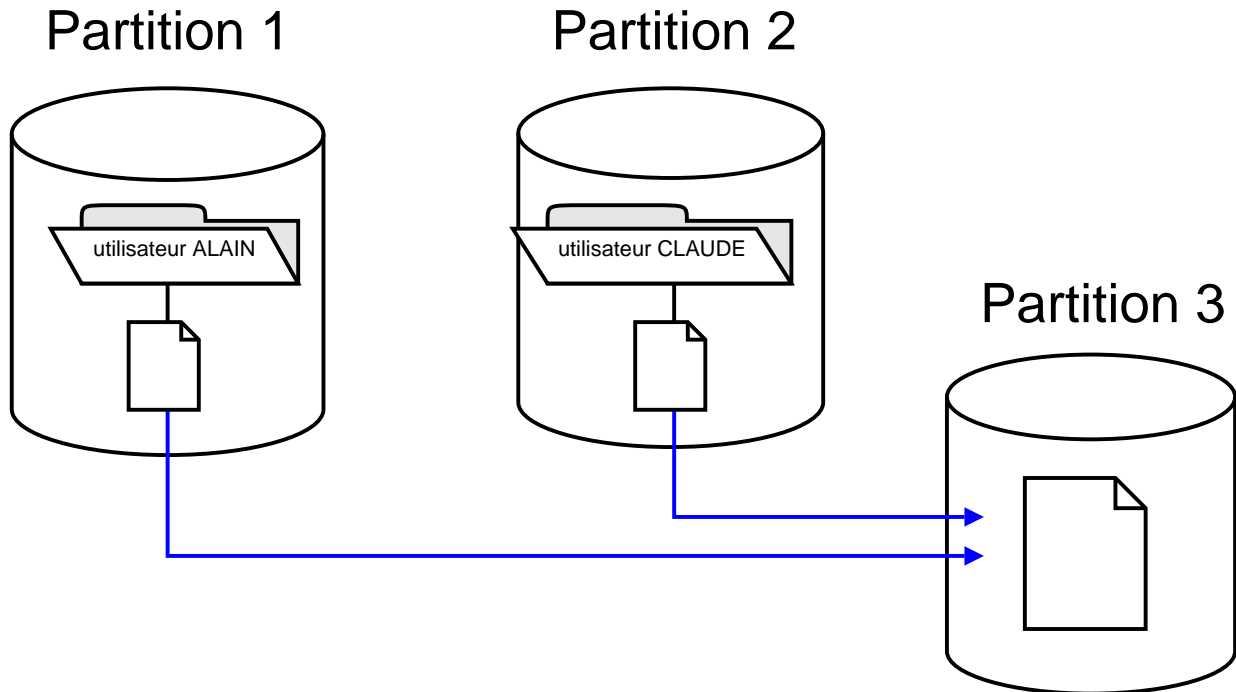
On dit « lien symbolique », « symbolic link », « symlink ».

Un lien **symbolique** est non limité à une partition d'un disque dur parce qu'utilisant le chemin d'un objet et non pas son numéro d'inode.



A quoi sert un lien symbolique ?

Exemple : fichier de configuration commun à tous les utilisateurs



Pourquoi ne peut-on pas employer des liens hard dans cet exemple ?

La commande à utiliser est : `ln -s original synonyme`

```
% ls -l ananas.jpg
-rw-r--r-- 1 besancon ars 9919 Jul 14 10:15 ananas.jpg
% ln -s ananas.jpg fruit.jpg

% ls -li ananas.jpg fruit.jpg
357 -rw-r--r-- 1 besancon ars 9919 Jul 14 10:15 ananas.jpg
358 lrwxr-xr-x 1 besancon ars 8 Oct 17 18:26 fruit.jpg -> ananas.jpg

% ls -lL ananas.jpg fruit.jpg
-rw-r--r-- 1 besancon ars 9919 Jul 14 10:15 ananas.jpg
-rw-r--r-- 1 besancon ars 9919 Jul 14 10:15 fruit.jpg
```

## Suppression d'un lien symbolique par « rm »

```
% ls -li ananas.jpg fruit.jpg
357 -rw-r--r--  1 besancon ars   9919 Jul 14 10:15 ananas.jpg
358 lrwxr-xr-x  1 besancon ars     8 Oct 17 18:26 fruit.jpg -> ananas.jpg

% rm ananas.jpg
% ls -liL fruit.jpg
358 lrwxr-xr-x  1 besancon ars     8 Oct 17 18:26 fruit.jpg -> ananas.jpg
% cat fruit.jpg
cat: fruit.jpg: No such file or directory
```

Les systèmes UNIX imposent les droits `lrwxr-xr-x` sur le lien (selon l'UNIX cela pourra être à la place `lrwxrwxrwx`).

**Ils ne peuvent pas être modifiés.**

**On ne peut que changer les droits d'un fichier pointé par un lien symbolique :**

```
% ls -l ananas.jpg fruit.jpg
-rw-r--r--  1 besancon ars   9919 Jul 14 10:15 ananas.jpg
lrwxr-xr-x  1 besancon ars     8 Oct 17 18:26 fruit.jpg -> ananas.jpg

% chmod 600 fruit.jpg
% ls -l
-rw-----  1 besancon ars   9919 Jul 14 10:15 ananas.jpg
lrwxr-xr-x  1 besancon ars     8 Oct 17 18:26 fruit.jpg -> ananas.jpg

% ls -lL ananas.jpg fruit.jpg
-rw-----  1 besancon ars   9919 Jul 14 10:15 ananas.jpg
-rw-----  1 besancon ars   9919 Jul 14 10:15 fruit.jpg
```

Sous Windows notion de *raccourci*, *shortcut*.

## Un raccourci Windows n'est pas équivalent à un lien symbolique UNIX !

- pas de commande standard pour générer un raccourci ; uniquement via interface graphique ou API (VBS, etc.) a priori
- impossibilité d'accéder via des commandes en ligne à l'objet via le raccourci comme on le ferait sous UNIX (voir ci-après) ; il faut passer par l'interface graphique à la place

### ◇ Echec d'un « cd » sur le raccourci sur un dossier :

```
C:\Documents and Settings\besancon\My Documents>dir
Volume in drive C is Windows XP
Volume Serial Number is 0C5C-E708

Directory of C:\Documents and Settings\besancon\My Documents

06/07/2004  20:53    <DIR>          .
06/07/2004  20:53    <DIR>          ..
06/07/2004  20:52                725 Shortcut-to-My-Pictures.lnk
                1 File(s)                725 bytes
                2 Dir(s)          551 477 248 bytes free

C:\Documents and Settings\besancon\My Documents>cd Shortcut-to-My-Pictures.lnk

The directory name is invalid.
```



### ◇ Echec d'un « dir » sur le raccourci sur un dossier :

```
C:\Documents and Settings\besancon\My Documents>dir Shortcut-to-My-Pictures.lnk
Volume in drive C is Windows XP
Volume Serial Number is 0C5C-E708

Directory of C:\Documents and Settings\besancon\My Documents

06/07/2004  20:52                725 Shortcut-to-My-Pictures.lnk
             1 File(s)                    725 bytes
             0 Dir(s)                    551 477 248 bytes free
```

### ◇ Echec d'un « type » sur le raccourci sur un fichier texte :

```
C:\Documents and Settings\besancon\My Documents>dir
Volume in drive C is Windows XP
Volume Serial Number is 0C5C-E708

Directory of C:\Documents and Settings\besancon\My Documents

06/07/2004  21:06    <DIR>          .
06/07/2004  21:06    <DIR>          ..
06/07/2004  21:00                1 870 a.txt
06/07/2004  21:06                757 b.txt.lnk
...

C:\Documents and Settings\besancon\My Documents>type b.txt.lnk
L  ?q?  +  Fc  á?lÖèc-?É?sIíc-?á.álic-?N  ?  e?k ?
PaO- O:i?ó +00Y? /C:\  \ 1  -0?p? \DOCUME~1  D ? ? n+N0Auu0lô
q  Documents  and  Settings  ? @ 1  @0,V? besancon ( ?
? n+00qsu0lôq  besancon  ? d 1  S0sk? MYDOCU~1  0 ? ? n+00qsu0lôq
My  Documents  ? ? ? ? n+besancon  ? 6 2 N  u0?y  a.txt " ? ?
n+u0-ûu0?ÿq  a . t x t  q n  ? ? ? 7  m  ? ? t \ ? ?  Windows
XP C:\Documents and Settings\besancon\My Documents\a.txt  . \ a . t x t / C : \
Documents  and  Settings  \  besancon  \  My  Docum
ents  `  ?  áX  best  =+A-?+=N2bb++-î,33lYl-+?a· PV=+A-?+=Nb
b++-î,3ElYl-+?a· PV+
```

Le système NTFS de Windows offre la notion de « *junction* » mais il y a peu d'utilitaires pour les utiliser.

On trouve :

- « linkd.exe », « delrp.exe » dans le resource kit Windows 2K/XP/2K3
- « junction.exe » d'URL  
<http://www.sysinternals.com/ntw2k/source/misc.shtml#junc>

### ◇ Création d'une junction :

```
C:\Documents and Settings\besancon\My Documents>junction tools2 tools
```

```
Junction v1.03 - Win2K junction creator and reparse point viewer
Copyright (C) 2000-2002 Mark Russinovich
Systems Internals - http://www.sysinternals.com
```

```
Created: C:\Documents and Settings\besancon\My Documents\tools2
Targetted at: C:\Documents and Settings\besancon\My Documents\tools
```

Et on voit bien la junction « tools2 » :

```
C:\Documents and Settings\besancon\My Documents>dir
```

```
Volume in drive C is Windows XP
Volume Serial Number is 0C5C-E708
```

```
Directory of C:\Documents and Settings\besancon\My Documents
```

```
06/07/2004  22:14    <DIR>          .
...
06/07/2004  22:13    <DIR>          tools
06/07/2004  22:14    <JUNCTION>    tools2
...
```

### ◇ Utilisation d'une junction :

```
C:\Documents and Settings\besancon\My Documents>dir tools2
Volume in drive C is Windows XP
Volume Serial Number is 0C5C-E708

Directory of C:\Documents and Settings\besancon\My Documents\tools2

06/07/2004  22:13    <DIR>          .
06/07/2004  22:13    <DIR>          ..
15/01/2000  09:34                749 README.TXT
                1 File(s)          749 bytes
                2 Dir(s)        423 448 576 bytes free

C:\Documents and Settings\besancon\My Documents>
```

### ◇ Utilisation d'une junction (2) :

```
C:\Documents and Settings\besancon\My Documents>junction tools2

Junction v1.03 - Win2K junction creator and repase point viewer
Copyright (C) 2000-2002 Mark Russinovich
Systems Internals - http://www.sysinternals.com

C:\Documents and Settings\besancon\My Documents\tools2: JUNCTION
Substitute Name: C:\Documents and Settings\besancon\My Documents\tools
```

### ◇ Destruction de la junction :

```
C:\Documents and Settings\besancon\My Documents>junction -d tools2

Junction v1.03 - Win2K junction creator and repase point viewer
Copyright (C) 2000-2002 Mark Russinovich
Systems Internals - http://www.sysinternals.com

Deleted tools2.
```

## Chapitre 7

***Gestion de versions de fichiers***

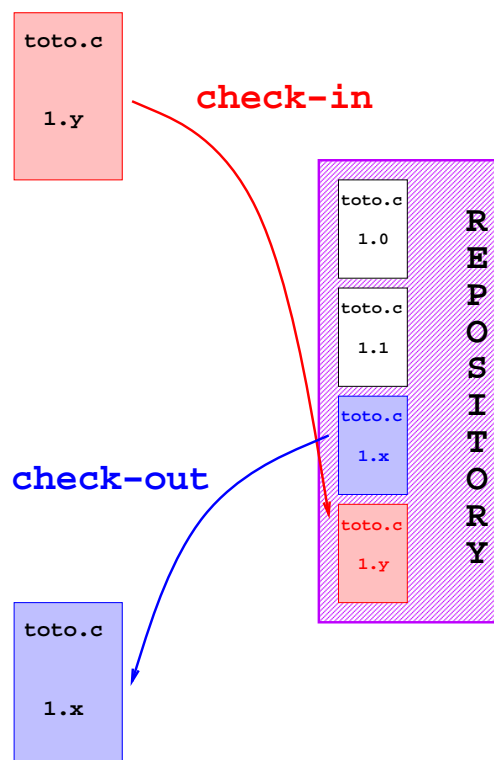
## Chapitre 7 • Gestion de versions de fichiers

## §7.1 • Introduction

Principe général à tous les outils de gestion de versions de fichiers :

On stocke en fait les « diff » entre versions successives du fichier.

Seuls les noms du répertoire de repository et les noms des commandes de check-in et check-out changeront d'un package de gestion à un autre.



SCCS  $\equiv$  *Source Code Control System*  
1975

Nom du repository	« SCCS »
Création dans le repository	« sccs create programme.c »
Check-out en read-write	« sccs edit programme.c »
Check-in	« sccs delta programme.c »
Check-out en read-only	« sccs get programme.c »
Comparaison avec le repository	« sccs diffs programme.c »
Historique des versions	« sccs prt programme.c »

On a :

```
sccs deledit == sccs delta + sccs edit
sccs delget  == sccs delta + sccs get
```

RCS  $\equiv$  *Revision Control System*

« ftp://ftp.lip6.fr/pub/gnu/rcs/ »

Nom du repository	« RCS »
Création dans le repository	« rcs -i programme.c »
Check-out en read-write	« co -l programme.c »
Check-in	« ci programme.c »
Check-out en read-only	« co programme.c »
Comparaison avec le repository	« rcsdiff programme.c »
Historique des versions	« rlog programme.c »

On a :

```
ci -l == ci + co -l
```

CVS  $\equiv$  *Concurrent Version System*

Construit au dessus de RCS (pour sa gestion interne des fichiers mais les commandes RCS ne sont pas utilisées)

Utilisé par de nombreuses équipes de développeurs de programmes sur Internet.

⇒ CVS utilisera le terme de *projet*

⇒ CVS peut fonctionner en réseau, le repository peut être sur une machine distante

Pour utiliser CVS, on doit définir 2 variables d'environnement :

- variable « CVSROOT » : chemin du repository par défaut
- variable « EDITOR » : éditeur de texte par défaut (« vi » par défaut)

Pour travailler sous CVS :

- créer un repository ou se connecter à un repository

```
% CVSROOT=/chemin/vers/mon/projet
% export CVSROOT
% cvsinit
```

La commande « cvsinit » crée les répertoires et les fichiers d'administration nécessaires.

- commande « cvs checkout filename »  
Une copie du fichier indiqué est extraite du repository.
- commande « cvs add filename »  
Ajoute le fichier indiqué à l'arborescence du repository.
- commande « cvs remove filename »  
Supprime un fichier du repository.
- commande « cvs commit »  
Entérine les actions des « cvs add » et « cvs remove ».  
Incorpore au repository vos modifications.
- commande « cvs update »  
Pour récupérer sur son disque l'état actuel du repository.

- commande « `cv diff` »  
Montre les différences entre votre copie locale et le repository.
- commande « `cv export` »  
?
- commande « `cv history` »  
Affiche l'historique des modifications.
- commande « `cv log` » ou commande « `cv status` »  
Affiche des informations sur le module du répertoire de travail courant.

### ◇ Pour travailler avec CVS à travers le réseau

2 contextes :

- on veut suivre un projet en tant que spectateur ; on ne modifiera pas les fichiers  
⇒ CVS anonyme
- on veut suivre un projet en tant qu'acteur actif ; on modifiera des fichiers  
⇒ CVS authentifié

### ◇ CVS anonyme

On veut suivre un projet en tant que spectateur ; on ne modifiera pas les fichiers.

La phase de connexion est du type suivant :

```
% CVSROOT=:pserver:anonymous@anoncvs.example.com:/cvs/gnome
% export CVSROOT
% cvs login
CVS password: <-- taper retour chariot
```

On procède ensuite avec les commandes normales de récupération de fichiers.

### ◇ CVS authentifié

On veut suivre un projet en tant qu'acteur actif ; on modifiera des fichiers.

On utilisera SSH pour sécuriser la connexion :

```
% CVS_RSH=ssh
% export CVS_RSH
% CVSROOT=:ext:user@cvs.example.com:/chemin/vers/repository
% export CVSROOT
```

On procède ensuite avec les commandes normales de récupération de fichiers.



**A completer...**

## Chapitre 8

# *Commandes de manipulation de base d'objets UNIX (suite)*

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.1 • Affichage du contenu d'un fichier texte : cat

(en anglais *concatenate*)

Syntaxe : `cat fichiers`

Par exemple :

```
% cat exemple.txt
```

```
This system is for the use of authorized users only. Individuals using this computer system without authority, or in excess of their authority, are subject to having all of their activities on this system monitored and recorded by system personnel.
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.2 • (Windows : : Affichage du contenu d'un fichier texte : type.exe)

Commande « `type.exe` »

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.3 • Affichage du contenu d'un fichier texte : `more`

(en anglais *more*)

En cas de texte très long, la commande « `cat` » n'est pas pratique. On lui préférera la commande « `more` » pour son affichage page d'écran par page d'écran.

Syntaxe : `more fichiers`

- Caractère « `q` » pour quitter (en anglais *quit*)
- Caractère espace pour avancer d'une page d'écran
- Caractère « `b` » pour revenir en arrière d'une page (en anglais *backward*)
- Caractère « `f` » pour avancer d'une page d'écran (en anglais *forward*)

La commande « `man` » affiche en fait les pages du manuel au moyen de la commande « `more` »

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.4 • (Windows : : Affichage du contenu d'un fichier texte : `more.exe`)

Commande « `more.exe` »

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.5 • Affichage du contenu d'un fichier texte : `less`

(en anglais *less*)

La commande « `less` » possède quelques fonctionnalités agréables de plus que la commande « `more` ». Par exemple : remonter dans le fichier (à l'origine « `more` » ne le faisait pas).

Syntaxe : `less fichiers`

Même méthode d'utilisation que pour « `more` ».

**ATTENTION** : il existe une variable d'environnement (voir page 513) appelée « `LESSOPEN` » qui complique le fonctionnement de la commande en fait (sera vu en TP).

⇒ Désactiver la variable en pratique (faire « `unset LESSOPEN` », voir page 510).

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.6 • Comptage de lignes dans un fichier : `wc`

(en anglais *word count*)

Syntaxe : `wc [option] fichiers`

Quelques options intéressantes :

- « `-c` » : nombre de caractères uniquement (en anglais *character*)
- « `-w` » : nombre de mots uniquement (en anglais *word*)
- « `-l` » : nombre de lignes uniquement (en anglais *line*)

Par exemple :

```
% wc exemple.txt
   3   16   82 exemple.txt
% wc -l exemple.txt
   3 exemple.txt
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *difference*)

Syntaxe : `diff [options] fichier1 fichier2`

Objet : réaliser la comparaison ligne à ligne du fichier texte « `fichier2` » par rapport au fichier texte « `fichier1` ».

Deux options intéressantes :

- option « `-c` » : affiche de quelques lignes du contexte (en anglais *contextual*)
- option « `-u` » : mode unifié (en anglais *unified*)

### ◇ Exemple 1 : utilisation sans paramètre

```
% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.

% cat fichier2
1 Blabla bla bla.
2 Deux fautes d'ortographe ici.
3 Encore du blabla bla bla.

% diff fichier1 fichier2
2c2
< Deux fotes d'ortographe ici.
---
> Deux fautes d'orthographe ici.
```

Interprétation de « `2c2` » : la ligne 2 de « `fichier2` » est changée par rapport à la ligne 2 de « `fichier1` »

## ◇ Exemple 2 : décalage de lignes

```
% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.

% cat fichier2
1
2 Blabla bla bla.
3 Deux fautes d'ortographe ici.
4 Encore du blabla bla bla.

% diff fichier1 fichier2
0a1
>
2c3
< Deux fotes d'ortographe ici.
---
> Deux fautes d'ortographe ici.
```

Interprétation de « 0a1 » : la ligne 1 de « fichier2 » est ajoutée par rapport à la ligne 0 de « fichier1 »

Interprétation de 2c3 : la ligne 3 de « fichier2 » est changée par rapport à la ligne 2 de « fichier1 »

## ◇ Exemple 3 : option « -c »

```
% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.

% cat fichier2
1 Blabla bla bla.
2 Deux fautes d'ortographe ici.
3 Encore du blabla bla bla.

% diff -c fichier1 fichier2
*** fichier1  Sun Sep  9 19:06:13 2001
--- fichier2  Sun Sep  9 19:06:24 2001
*****
*** 2 ****
! Deux fotes d'ortographe ici.
--- 2 ----
! Deux fautes d'orthographe ici.
```

Interprétation de « \*\*\* 2 \*\*\*\* » et « --- 2 ---- » :  
la ligne 2 de « fichier2 » est changée par rapport à la ligne 2 de « fichier1 ».

## ◇ Exemple 4 : option « -u »

```

% cat fichier1
1 Blabla bla bla.
2 Deux fotes d'ortographe ici.
3 Encore du blabla bla bla.

% cat fichier2
1 Blabla bla bla.
2 Deux fautes d'ortographe ici.
3 Encore du blabla bla bla.

% diff -u fichier1 fichier2
--- fichier1      Fri Sep 16 22:05:15 2005
+++ fichier2      Fri Sep 16 22:05:20 2005
@@ -1,3 +1,3 @@
 Blabla bla bla.
-Deux fotes d'ortographe ici.
+Deux fautes d'ortographe ici.
 Encore du blabla bla bla.

```

L'affichage est un peu plus lisible.

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.8 • Comparaison de deux fichiers binaires : cmp

(en anglais *compare*)

Syntaxe : `cmp fichier1 fichier2`

Objet : réaliser la comparaison octet à octet du fichier binaire

« fichier2 » par rapport au fichier binaire « fichier1 ».

Utilité par exemple : comparer des binaires d'un système à la recherche de programmes piratés, etc.

Exemple :

```

% cmp program1.exe programme2.exe
program1.exe programme2.exe differ: char 3174, line 9

```

Moralité : les deux fichiers sont différents

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.9 • Extraction des premières lignes de fichiers : head

(en anglais *head*)

Syntaxe : `head [-nombre] fichier`

#### ◇ Exemple : utilisation intéressante

```
% head -3 /etc/motd
SunOS Release 4.1.4 (EXCALIBUR.LPS.ENS.FR [1.1]): Fri Aug 8 17:43:56 GMT 1997
This system is for the use of authorized users only. Individuals using
this computer system without authority, or in excess of their authority,
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.10 • Extraction des dernières lignes de fichiers : tail

(en anglais *tail*)

Plusieurs syntaxes :

- `tail [-nombre] fichier` : les N dernières lignes
- `tail [+nombre] fichier` : de la Nième ligne à la fin du fichier
- `tail [-f] fichier` : affichage « en live »

#### ◇ Exemple : utilisation intéressante

```
% tail -3 /etc/motd
advised that if such monitoring reveals possible evidence of criminal
activity, system personnel may provide the evidence of such monitoring
to law enforcement officials.
```



### ◇ Exemple 3 : très pratique

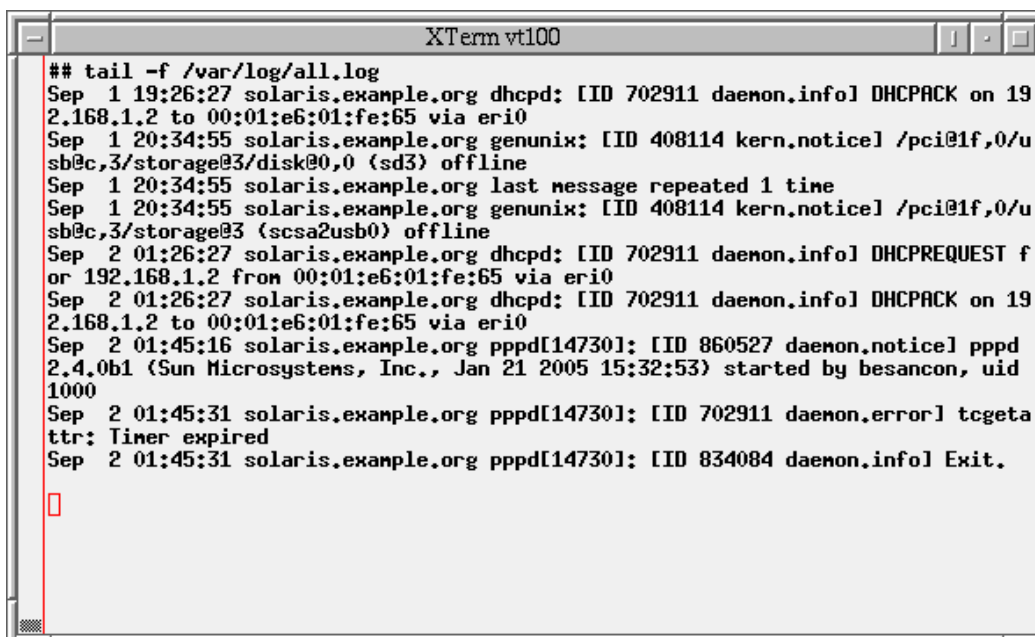
```
% tail +3 /etc/motd
this computer system without authority, or in excess of their authority,
are subject to having all of their activities on this system monitored
and recorded by system personnel.
```

In the course of monitoring individuals improperly using this system, or in the course of system maintenance, the activities of authorized users may also be monitored.

Anyone using this system expressly consents to such monitoring and is advised that if such monitoring reveals possible evidence of criminal activity, system personnel may provide the evidence of such monitoring to law enforcement officials.

### ◇ Exemple 4 : Affichage des dernières lignes en temps réel

Syntaxe : `tail -f fichier`



```
XTerm vt100
## tail -f /var/log/all.log
Sep  1 19:26:27 solaris.example.org dhcpd: [IID 702911 daemon.info] DHCPACK on 19
2.168.1.2 to 00:01:e6:01:fe:65 via eri0
Sep  1 20:34:55 solaris.example.org genunix: [IID 408114 kern.notice] /pci@1f,0/u
sb@c,3/storage@3/disk@0,0 (sd3) offline
Sep  1 20:34:55 solaris.example.org last message repeated 1 time
Sep  1 20:34:55 solaris.example.org genunix: [IID 408114 kern.notice] /pci@1f,0/u
sb@c,3/storage@3 (scsa2usb0) offline
Sep  2 01:26:27 solaris.example.org dhcpd: [IID 702911 daemon.info] DHCPREQUEST f
or 192.168.1.2 from 00:01:e6:01:fe:65 via eri0
Sep  2 01:26:27 solaris.example.org dhcpd: [IID 702911 daemon.info] DHCPACK on 19
2.168.1.2 to 00:01:e6:01:fe:65 via eri0
Sep  2 01:45:16 solaris.example.org pppd[14730]: [IID 860527 daemon.notice] pppd
2.4.0b1 (Sun Microsystems, Inc., Jan 21 2005 15:32:53) started by besancon, uid
1000
Sep  2 01:45:31 solaris.example.org pppd[14730]: [IID 702911 daemon.error] tcgeta
ttr: timer expired
Sep  2 01:45:31 solaris.example.org pppd[14730]: [IID 834084 daemon.info] Exit.
```

Quitter en faisant Ctrl-C.

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *cut*)

Syntaxe : `cut [options] fichiers`

Quelques options :

- « `-c` » : découpage selon des positions de caractères (en anglais *character*)
- « `-f` » : découpage selon des positions de mots (en anglais *field*)
- « `-d` » : indique le délimiteur de mots (en anglais *delimiter*)

Exemples :

- extraction sur chaque ligne de caractères pris isolément :  
`cut -c 1,8,27 fichier`
- extraction sur chaque ligne de caractères d'une position 1 à une position 2 :  
`cut -c 25-42 fichier`
- extraction sur chaque ligne (constituée de mots séparés par un certain délimiteur) de mots pris isolément :  
`cut -d: -f 1,5 fichier`
- extraction sur chaque ligne (constituée de mots séparés par un certain délimiteur) du mot *i* au mot *j* :  
`cut -d: -f 4-7 fichier`

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *sort*)

Syntaxe : `sort [options] fichiers`

Quelques options :

- option « `-n` » : tri numérique (en anglais *numerical*)
- option « `-r` » : tri par ordre décroissant (en anglais *reverse*)
- option « `-t` » : permet de spécifier le séparateur de mots
- option « `-k` » : spécifie la clef de tri (en anglais *key*) ; on peut indiquer plusieurs clefs de tri

### ◇ Exemple 1

```
% cat exemple.txt
arbre
12
ascenceur
2
ordinateur
```

```
% sort exemple.txt
12
2
arbre
ascenceur
ordinateur
```

## ◇ Exemple 2

```
% cat exemple.txt
```

```
12
2
3
33
22
```

```
% sort exemple.txt
```

```
12
2
22
3
33
```

```
% sort -n exemple.txt
```

```
2
3
12
22
33
```

```
% sort -rn exemple.txt
```

```
33
22
12
3
2
```

## ◇ Exemple 3

```
% cat exemple.txt
```

```
or:100000
argent:40000
bois:15
```

```
% sort exemple.txt
```

```
argent:40000
bois:15
or:100000
```

```
% sort -t : -k 2 exemple.txt
```

```
or:100000
bois:15
argent:40000
```

```
% sort -t : -k 2 -n exemple.txt
```

```
bois:15
argent:40000
or:100000
```

### ◇ Exemple 4 : trier des adresses IP

Soit le fichier à trier :

```
192.168.1.1      ananas.example.com
192.168.1.4      poire.example.org
127.0.0.1       localhost
192.168.1.3      cerise.example.org
134.157.46.129  serveur.formation.jussieu.fr
192.168.1.100   kiwi.example.org
192.168.1.2     banane.example.org
192.168.2.1     freebox.example.org
```

On a donc :

```
% sort -t . -k 1n,1 -k 2n,2 -k 3n,3 -k 4n,4 exemple.txt
127.0.0.1      localhost
134.157.46.129 serveur.formation.jussieu.fr
192.168.1.1    ananas.example.com
192.168.1.2    banane.example.org
192.168.1.3    cerise.example.org
192.168.1.4    poire.example.org
192.168.1.100 kiwi.example.org
192.168.2.1    freebox.example.org
```

Importance de bien préciser les clefs de tri :

```
% cat exemple.txt
```

```
2.2.3.0
2.10.20.0
2.10.3.0
10.2.2.0
2.10.100.0
2.10.100.5
2.10.100.43
2.10.10.8
```

```
% sort exemple.txt
```

```
10.2.2.0      <- mal classé
2.10.10.8
2.10.100.0
2.10.100.43
2.10.100.5   <- mal classé
2.10.20.0    <- mal classé
2.10.3.0     <- mal classé
2.2.3.0      <- mal classé
```

```
% sort -t . exemple.txt
```

```
10.2.2.0      <- mal classé
2.10.10.8
2.10.100.0
2.10.100.43
2.10.100.5   <- mal classé
2.10.20.0    <- mal classé
2.10.3.0     <- mal classé
2.2.3.0      <- mal classé
```

```
% sort -t . -k 1n,1 exemple.txt
```

```
2.10.10.8
2.10.100.0
2.10.100.43
2.10.100.5   <- mal classé
2.10.20.0    <- mal classé
2.10.3.0     <- mal classé
2.2.3.0      <- mal classé
10.2.2.0
```

## Importance de bien préciser les clefs de tri (suite) :

```
% sort -t . -k 1n,1 -k 2n,2 exemple.txt
2.2.3.0
2.10.10.8
2.10.100.0
2.10.100.43  <- mal classé
2.10.100.5
2.10.20.0    <- mal classé
2.10.3.0     <- mal classé
10.2.2.0
```

```
% sort -t . -k 1n,1 -k 2n,2 -k 3n,3 exemple.txt
2.2.3.0
2.10.3.0
2.10.10.8
2.10.20.0
2.10.100.0
2.10.100.43  <- mal classé
2.10.100.5
10.2.2.0
```

## Importance de bien préciser les clefs de tri (suite) :

### La solution pour trier ces adresses IP :

```
% sort -t . -k 1n,1 -k 2n,2 -k 3n,3 -k 4n,4 exemple.txt
2.2.3.0
2.10.3.0
2.10.10.8
2.10.20.0
2.10.100.0
2.10.100.5
2.10.100.43
10.2.2.0
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.13 • (Windows : : Tri d'un fichier : `sort.exe`)

```
C:\documents and settings\besancon\mes documents>sort /?  
SORT [/R] [/+n] [/M kilo-octets] [/L locale] [/RE octets_enregistrement]  
  [[lecteur1:][chemin1]nom_fichier1] [/T [lecteur2:][chemin2]]  
  [/O [lecteur3:][chemin3]nom_fichier3]
```

Point de vue unixien sur le `sort` de Windows :

- considération préhistorique de la place mémoire
- pas de possibilité de tri sur des champs mais uniquement sur des positions de caractères
- pas de tri numérique

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.14 • Elimination des lignes redondantes d'un fichier : `uniq`

(en anglais *uniq*)

Syntaxe : `uniq [options] fichier`

Objet : élimine les lignes **consécutives** redondantes

Quelques options :

- option « `-c` » : précède chaque ligne du résultat du nombre d'occurrences de cette ligne dans le fichier original (en anglais *count*)

◇ Exemple 1

```
% cat exemple.txt
Ceci est un test.
Ceci est un test.
TEST
unix
TEST
TEST
```

```
% uniq exemple.txt
Ceci est un test.
TEST
unix
TEST
```

◇ Exemple 2

```
% cat exemple.txt
Ceci est un test.
Ceci est un test.
TEST
unix
TEST
TEST
logiciel
```

```
% uniq -c exemple.txt
 2 Ceci est un test.
 1 TEST
 1 unix
 2 TEST
 1 logiciel
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.15 • Création d'un fichier vide : `touch`

(en anglais *touch*)

Syntaxe : `touch fichiers`

**ATTENTION** : cette commande ne permet de créer un fichier vide que si le fichier mentionné n'existe pas déjà !

◇ Exemple

```
% ls -l exemple.txt
exemple.txt: No such file or directory
```

```
% touch exemple.txt
```

```
% ls -l exemple.txt
```

```
-rw-r--r--  1 besancon ars          0 Sep 27 12:58 exemple.txt
```



## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *touch*)

Syntaxe : touch [options] [-t time] objet

Un objet UNIX a **trois** dates parmi ses attributs (ce sera revu plus loin).

Quelques options :

- option « -a » : modification de la date d'accès du fichier (a ≡ accesstime)
- option « -m » : modification de la date de modification du fichier (m ≡ mtime)
- option « -t time » : indique une date à mettre autre que la date de l'instant ;  
format : AAAAMMJJhhmm.ss

### ◇ Exemples

```
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          49 Sep 27 13:07 exemple.txt

% touch -m -t 199901012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          49 Jan  1  1999 exemple.txt

% touch -m -t 09012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          49 Sep  1 22:33 exemple.txt
```

# Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.17 • Création d'objets temporaires : /tmp

(en anglais *temporary*)

Le répertoire /tmp sert à stocker des objets temporaires.

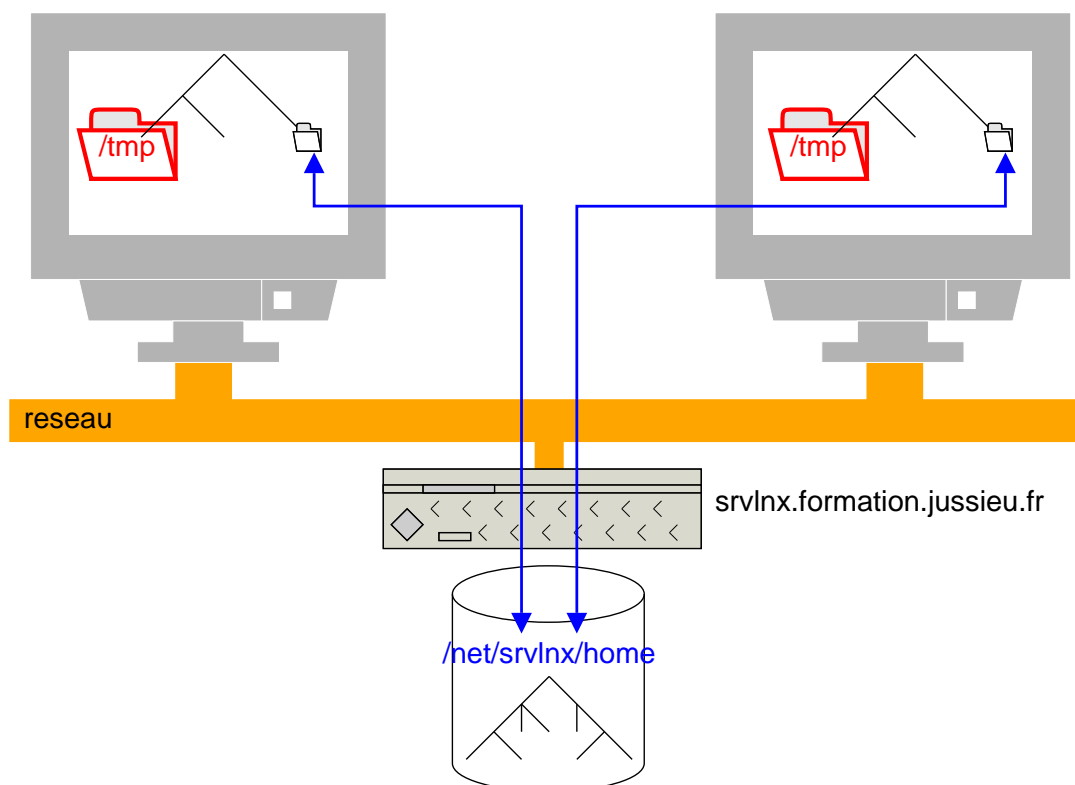
```
% ls -ld /tmp
drwxrwxrwt 12 root      sys          2648 Sep 28 13:02 /tmp
```

Les droits d'accès de /tmp sont **1777** exprimé en octal. Leurs significations :

- signification de 777 : tout le monde sur la machine peut créer, modifier, effacer des objets
- signification de 1000 : un utilisateur ne peut effacer que les objets qui lui appartiennent

Ces droits d'accès seront revus et expliqués page 317.

**ATTENTION** : le répertoire « /tmp » est **local** à chaque machine :



## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.18 • (Windows : : variable temp, répertoire temp)

Il existe sous Windows :

- répertoire public « C:\winnt\temp » sous Windows 2000
- répertoire public « C:\windows\temp » sous Windows XP et Windows 2003
- pour chaque utilisateur d'un Windows 2000 server ou mieux, répertoire temporaire donné par la variable « %temp% », en fait « C:\Documents and settings\utilisateur\local settings\temp\ »

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.19 • Manipulation des noms d'objets : basename

(en anglais *base of name*)

Syntaxe : `basename fichier [suffixe]`

Objet : supprimer un suffixe (extension) du nom d'un objet

```
% basename document.doc .doc
document
```

### ◇ Utilisation archi classique :

Changer l'extension « `.txt` » de tous les fichiers du répertoire courant en l'extension « `.doc` » :

```
for i in *.txt
do
  mv $i `basename $i .txt`.doc
done
```

(la syntaxe de cet exemple sera revue et expliquée page 605).

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *file*)

Syntaxe : `file objets`

Cette commande permet de deviner à quelle application est lié l'objet. Elle s'appuie pour rendre un avis sur la reconnaissance de motifs connus dans le contenu de l'objet.

Pour voir les motifs, se reporter au fichier « `/etc/magic` »

Version améliorée de la commande (l'amélioration porte sur le nombre de types de fichiers reconnus) :

```
ftp://ftp.astron.com/pub/file/file-4.09.tar.gz
```

## Exemple d'utilisation :

Soit des fichiers pris dans le cache d'un navigateur web

De quels types sont ces fichiers aux noms sans extension ?

```
% ls -l
total 720110
-rw----- 1 besancon ars      23185 Dec 18 18:41 C3866435d01
-rw----- 1 besancon ars  357515264 Dec 18 18:33 DF0D61DCd01
-rw----- 1 besancon ars   2306816 Dec 18 18:41 _CACHE_001_
-rw----- 1 besancon ars   2670592 Dec 18 18:41 _CACHE_002_
-rw----- 1 besancon ars   135168 Dec 18 18:01 _CACHE_MAP_

% file *
C3866435d01: HTML document text
DF0D61DCd01: Zip archive data, at least v2.0 to extract
_CACHE_001_: MP32, Mono
_CACHE_002_: data
_CACHE_MAP_: pfm?
```

## Quelques exemples non exhaustifs (2) :

```
% file inconnu
inconnu: JPEG image data, JFIF standard 1.02, resolution (DPI), 72 x 72

% file inconnu
inconnu: ASCII text

% file inconnu
inconnu: TeX DVI file (TeX output 2002.08.10:1903)

% file inconnu
inconnu: PostScript document text conforming at level 2.0

% file /usr/bin/ls
/usr/bin/ls: ELF 32-bit MSB executable, SPARC, version 1 (SYSV),
dynamically linked (uses shared libs), stripped

% file /usr/lib/libc.a
/usr/lib/libc.a: current ar archive

% file /usr/lib/libc.so.1
/usr/lib/libc.so.1: ELF 32-bit MSB shared object, SPARC, version 1 (SYSV),
not stripped
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *octal dump*)

Syntaxe : `od [options] objet`

Principales options (cumulables) :

- option « `-c` » : affichage en ascii (plus pseudo codes C)
- option « `-b` » : affichage en base 8
- option « `-x` » : affichage en base 16

Quelques exemples non exhaustifs (2) :

```
% cat exemple.txt
abcde
```

```
% od -b exemple.txt
0000000 141 142 143 144 145 012
0000006
```

```
% od -c exemple.txt
0000000  a  b  c  d  e  \n
0000006
```

```
% od -x exemple.txt
0000000 6162 6364 650a
0000006
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *translate*)

Commande de base sur tous les UNIX.

Syntaxe : `tr [options] jeu1 [jeu2]`

Deux utilisations possibles de la commande :

- remplacer un par un chaque caractère du jeu 1 par le caractère en même position dans le jeu 2
- effacer les caractères du jeu 1 (pas de jeu2 mentionné)

**On notera qu'aucun nom de fichiers n'est à donner sur la ligne de commande.**

Soit le fichier contenant les lignes suivantes :

```
toto
cheval
```

### ◇ Exemple 1

On veut convertir les lettres o en lettres e :

```
% tr 'o' 'e' < exemple.txt
tete
cheval
```

### ◇ Exemple 2

On veut convertir les lettres minuscules en lettres majuscules :

```
% tr '[a-z]' '[A-Z]' < exemple.txt
TOTO
CHEVAL
```

## ◇ Exemple 3

Meilleure façon de convertir les lettres minuscules en lettres majuscules (ou vice versa) :

```
% cat exemple.txt
```

```
aoieàôùéèè
```

```
% tr '[:lower:]' '[:upper:]' < exemple.txt
```

```
AOIEÀÔÛÉÊÈ
```

Nouvelles écritures :

- « [:lower:] » : en anglais *lower case* : lettres minuscules
- « [:upper:] » : en anglais *upper case* : lettres majuscules
- « [:digit:] » : chiffres de la base 10
- « [:xdigit:] » : chiffres de la base 16
- « [:punct:] » : signes de ponctuation
- « [:alnum:] » : XXX
- « [:alpha:] » : XXX
- « [:graph:] » : XXX
- « [:print:] » : XXX
- « [:blank:] » : XXX

**ATTENTION** : Cela nécessite la variable d'environnement `LC_CTYPE` positionnée à une valeur correcte (voir page 513 pour les variables d'environnement) :

```
% echo $LC_CTYPE
```

```
en_US
```

```
% tr '[:lower:]' '[:upper:]' < exemple.txt
```

```
AOIEÀÔÛÉÊÈ
```

```
% unset LC_CTYPE
```

```
% tr '[:lower:]' '[:upper:]' < exemple.txt
```

```
AOIEàôùéèè
```



## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *disk filesystems*)

Syntaxe : `df [options] [répertoires]`

Quelques options intéressantes :

- « `-k` » : affichage des capacités en kilo octets
- « `-i` » : affichage des capacités en inodes (sur Solaris faire « `-o i` »)
- « `-h` » : affichage sous forme plus lisible (en anglais *human readable*) ; affichage variable ⇒ peu utilisable en programmation

Si l'on indique un répertoire, la commande affiche le remplissage du disque dur (local ou réseau) contenant ce répertoire.

Exemple pris sur une machine du réseau de la Formation Permanente :

```
% df -k
Filesystem      1k-blocks    Used Available Use% Mounted on
/dev/hda1        1143208    693050   391091   64% /
serveur:/net/serveur/home
                  1015695    783819   170935   82% /.automount/serveur/net/
serveur/home
serveur:/var/mail 246167     84838   136713   38% /.automount/serveur/var/
mail
```

## Exemple pris sur une machine Solaris :

```
% df
/                (/dev/dsk/c0t0d0s0 ):21867748 blocks  2209818 files
/devices         (/devices           ):          0 blocks          0 files
/system/contract (ctfs              ):          0 blocks 2147483620 files
/proc           (proc              ):          0 blocks    9892 files
/etc/mnttab     (mnttab            ):          0 blocks          0 files
/etc/svc/volatile (swap             ): 3186528 blocks   90949 files
/system/object  (objfs            ):          0 blocks 2147483450 files
/dev/fd         (fd                ):          0 blocks          0 files
/tmp            (swap             ): 3186528 blocks   90949 files
/var/run        (swap             ): 3186528 blocks   90949 files
/extra          (/dev/dsk/c0t0d0s7 ):79432732 blocks  6870541 files
/entrepot       (/dev/dsk/c0t2d0s2 ):11126010 blocks  6415734 files
/users          (/entrepot/users   ):11126010 blocks  6415734 files
/src            (/entrepot/src     ):11126010 blocks  6415734 files
/ars            (/entrepot/projets/ars):11126010 blocks  6415734 files
```

On notera le type d'affichage sans l'option « -k ».

## Exemple pris sur une machine Solaris :

```
% df -k
Filesystem          kbytes  used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0  20648041 9714167 10727394    48%  /
/devices            0         0         0         0%  /devices
ctfs                0         0         0         0%  /system/contract
proc               0         0         0         0%  /proc
mnttab             0         0         0         0%  /etc/mnttab
swap              1594208    952 1593256     1%  /etc/svc/volatile
objfs              0         0         0         0%  /system/object
fd                 0         0         0         0%  /dev/fd
swap              1685728   92472 1593256     6%  /tmp
swap              1593312     56 1593256     1%  /var/run
/dev/dsk/c0t0d0s7  56091807 16375441 39155448    30%  /extra
/dev/dsk/c0t2d0s2  57708710 52145705 4985918    92%  /entrepot
/entrepot/users    57708710 52145705 4985918    92%  /users
/entrepot/src      57708710 52145705 4985918    92%  /src
/entrepot/projets/ars
                    57708710 52145705 4985918    92%  /ars
```

## Exemple pris sur une machine Solaris :

```
% df -o i
df: operation not applicable for FSType autofs
df: operation not applicable for FSType ctfs
df: operation not applicable for FSType devfs
df: operation not applicable for FSType fd
df: operation not applicable for FSType lofs
df: operation not applicable for FSType mntfs
df: operation not applicable for FSType objfs
df: operation not applicable for FSType proc
df: operation not applicable for FSType tmpfs
Filesystem          iused   ifree   %iused  Mounted on
/dev/dsk/c0t0d0s0   324582 2209818   13%    /
/dev/dsk/c0t0d0s7     9459 6870541    0%    /extra
/dev/dsk/c0t2d0s2   662665 6415735    9%    /entrepot
```

Chapitre 8 • Commandes de manipulation de base d'objets UNIX  
(suite)

## §8.24 • (Windows : : Information sur le remplissage des disques : df.exe)

Récupérer le package cmdutils.zip sur le site

<http://paulsadowski.com/>. Il contient un binaire df.exe montrant disques locaux et réseau :

```
C:\>net use z: \\winserveur\c$
```

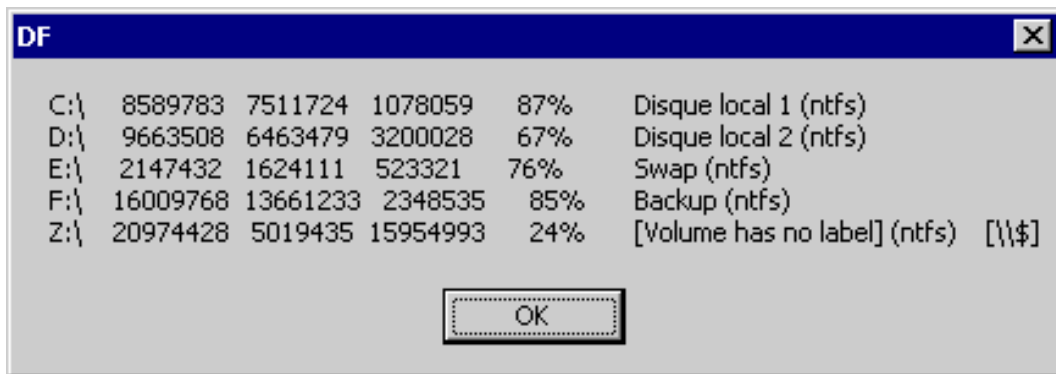
La commande s'est terminée correctement.

```
C:\>df.exe -b
```

```
C:\ 8589783 7511724 1078059 87% Disque local 1 (ntfs)
D:\ 9663508 6463479 3200028 67% Disque local 2 (ntfs)
E:\ 2147432 1624111 523321 76% Swap (ntfs)
F:\ 16009768 13661233 2348535 85% Backup (ntfs)
Z:\ 20974428 5020499 15953928 24% [Volume has no label]
```

Affichage dans fenêtre graphique possible :

C:\>`df.exe -b -w`



## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *disk usage*)

Syntaxe : `du [-s] [-k] répertoires`

Principales options (cumulables) :

- option « `-s` » : affichage uniquement du total (en anglais *sum*)
- option « `-k` » : affichage des totaux exprimés en kilo octets (en anglais *kilobytes*)
- « `-h` » : affichage sous forme plus lisible (en anglais *human readable*) ; affichage variable ⇒ peu utilisable en programmation

**Attention : ne pas confondre « `df` » et « `du` ».**

◇ Exemple 1 :

Affichage récursif avec des messages d'erreur :

```
% du -k picqueno
152    picqueno/.kde/share/config
du: cannot change to directory `picqueno/.kde/share/fonts': Permission de
du: cannot change to directory `picqueno/.kde/share/apps': Permission den
du: cannot change to directory `picqueno/.kde/share/mimelnk': Permission
du: cannot change to directory `picqueno/.kde/share/services': Permission
du: cannot change to directory `picqueno/.kde/share/icons': Permission de
176    picqueno/.kde/share
184    picqueno/.kde
588    picqueno/.mcp/trader-cache
592    picqueno/.mcp
20     picqueno/tp/2
32     picqueno/tp/4
4      picqueno/tp/1
4      picqueno/tp/3
68     picqueno/tp/5
776    picqueno/tp
1672   picqueno
```

◇ Exemple 2 :

Affichage récursif avec suppression des messages d'erreur :

```
% du -k picqueno 2>/dev/null
152    picqueno/.kde/share/config
176    picqueno/.kde/share
184    picqueno/.kde
588    picqueno/.mcp/trader-cache
592    picqueno/.mcp
20     picqueno/tp/2
32     picqueno/tp/4
4      picqueno/tp/1
4      picqueno/tp/3
68     picqueno/tp/5
776    picqueno/tp
1672   picqueno
```

### ◇ Exemple 3 :

Affichage du total avec des messages d'erreur :

```
% du -k -s picqueno
du: cannot change to directory `picqueno/.kde/share/fonts': Permission de
du: cannot change to directory `picqueno/.kde/share/apps': Permission der
du: cannot change to directory `picqueno/.kde/share/mimelnk': Permission
du: cannot change to directory `picqueno/.kde/share/services': Permission
du: cannot change to directory `picqueno/.kde/share/icons': Permission de
1672    picqueno
```

### ◇ Exemple 4 :

Affichage du total avec suppression des messages d'erreur :

```
% du -k -s picqueno 2>/dev/null
1672    picqueno
```

**Vous ne devez pas laisser votre compte se remplir de fichiers. Les disques durs n'ont pas une capacité infinie et hors de question de stocker toute la documentation disponible sur Internet chez vous !**

La commande « `du -k $HOME` » vous donnera la taille disque que votre homedirectory occupe. La commande passe en revue tous les répertoires et en affiche la taille.

Le résultat affiché est exprimé en kilo octets (1 ko = 1024 octets).

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.26 • (Windows : : Calcul de la place disque occupée : `diruse.exe`)

La commande « `diruse.exe` » (resource kit Windows 2000, support kit Windows XP/2003) calcule la place occupée dans une arborescence :

```
C:\>diruse.exe /s /k C:\Docume~1
```

Size (kb)	Files	Directory
0.00	0	\DOCUME~1
0.00	0	\DOCUME~1\Administrator
0.00	0	\DOCUME~1\Administrator\Application Data
0.00	0	\DOCUME~1\Administrator\Application Data\Microsoft
...		
0.02	1	\DOCUME~1\Default User\Start Menu\Programs\Startup
36.14	12	\DOCUME~1\Default User\Templates
732377.45	4711	SUB-TOTAL: \DOCUME~1
732377.45	4711	TOTAL: \DOCUME~1

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.27 • (Windows : : Calcul de la place disque occupée : `diskuse.exe`)

La commande « `diskuse.exe` » (resource kit 2000/XP/2003) calcule la place occupée dans une arborescence mais la calcule par utilisateur :

```
C:\>diskuse.exe /s /e:nul /t "c:\Documents and settings"
```

```
DiskUse
```

```
Version 1.3
```

```
Scanning Path .\.....
.....
.....
.....
.....
.....
.....
```

```
Resolving Names....
```

```
Sorting....
```

WINXP	Administrator	1653
WINXP	besancon	705532503
WINXP	root	36876743
BUILTIN	Administrators	6375184

Pas d'équivalent immédiat sous UNIX.

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.28 • Compression de fichiers : `compress`, `uncompress`, `zcat`

Syntaxes de quelques commandes de compression ou décompression :

- `compress [options] fichiers`
- `uncompress [options] fichiers.Z`
- `zcat fichiers.Z`

Le fichier compressé s'appelle après compression « `fichier.Z` ».

Le fichier décompressé retrouve son nom « `fichier` ».



## ◇ Exemples

```
% ls -l access_log
-rw-r--r--  1 besancon ars  19834224 Jul  6 18:36 access_log

% compress access_log

% compress -v access_log
access_log: Compression: 85.42% -- replaced with access_log.Z

% ls -l access_log.Z
-rw-r--r--  1 besancon ars   2890847 Jul  6 18:36 access_log.Z

% zcat access_log.Z
nfs2.institut.math.jussieu.fr - - [01/Jul/2004:02:02:12 +0200]
...
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.29 • Compression de fichiers : `gzip`, `gunzip`, `gzcat`

Cette série de commandes compresse mieux les fichiers que la famille autour de `compress`.

Syntaxes de quelques commandes de compression ou décompression :

- `gzip [options] fichiers`
- `gunzip [options] fichiers.gz`
- `gzcat fichiers.gz`

Le fichier compressé s'appelle après compression « `fichier.gz` ».

Le fichier décompressé retrouve son nom « `fichier` ».

De plus en plus répandu. Au cas où absent :

`ftp://ftp.lip6.fr/pub/gnu/gzip/`

**Attention :** dans les salles de TP, il faut utiliser « `zcat` » au lieu de « `gzcat` ».

## ◇ Exemples

```
% ls -l access_log
-rw-r--r--  1 besancon ars  19834224 Jul  6 18:36 access_log

% gzip access_log

% gzip -v access_log
access_log:          92.9% -- replaced with access_log.gz

% ls -l access_log.gz
-rw-r--r--  1 besancon ars   1396359 Jul  6 18:36 access_log.gz

% gzcat access_log.gz
nfs2.institut.math.jussieu.fr - - [01/Jul/2004:02:02:12 +0200]
...
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.30 • Compression de fichiers : bzip2, bunzip2, bzcat

Cette série de commandes compresse mieux les fichiers que la famille autour de `compress`.

Syntaxes de quelques commandes de compression ou décompression :

- `bzip2 [options] fichiers`
- `bunzip2 [options] fichiers.bz2`
- `bzcat fichiers.bz2`

Le fichier compressé s'appelle après compression « `fichier.bz2` ».

Le fichier décompressé retrouve son nom « `fichier` ».

Pas encore très répandu. Cf <http://sources.redhat.com/bzip2/>

◇ Exemples

```
% ls -l access_log
-rw-r--r--  1 besancon ars  19834224 Jul  6 18:36 access_log

% bzip2 access_log

% bzip2 -v access_log
access_log: 21.835:1,  0.366 bits/byte, 95.42% saved, 19834224 in, 9083

% ls -l access_log.bz2
-rw-r--r--  1 besancon ars    908376 Jul  6 18:36 access_log.bz2

% bzip2 -v access_log.bz2
nfs2.institut.math.jussieu.fr - - [01/Jul/2004:02:02:12 +0200]
...
```

◇ Temps de compression plus long

```
% ls -l access_log
-rw-r--r--  1 besancon ars  19834224 Jul  6 18:36 access_log

% time gzip -v access_log
access_log:          92.9% -- replaced with access_log.gz

real    0m3.229s
user    0m2.980s
sys     0m0.160s

% ls -l access_log
-rw-r--r--  1 besancon ars  19834224 Jul  6 18:36 access_log

% time bzip2 -v access_log
access_log: 21.835:1,  0.366 bits/byte, 95.42% saved, 19834224 in, 9083

real    1m13.341s
user    1m12.270s
sys     0m0.240s
```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

(en anglais *tape archive*)

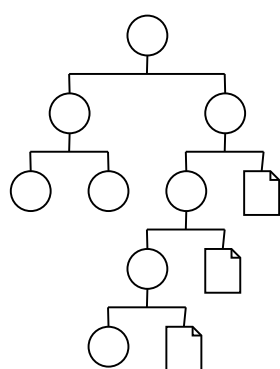
La commande `tar` permet d'archiver dans un seul fichier une arborescence.

Selon l'action que l'on veut faire, la syntaxe est la suivante :

- Création d'une archive : « `tar -cvf archive.tar objets` »
- Affichage du contenu d'une archive : « `tar -tvf archive.tar` »
- Extraction de l'archive complète : « `tar -xvf archive.tar` »
- Extraction d'un ou plusieurs objets de l'archive :  
« `tar -xvf archive.tar objets` »

Attention : la commande autorise d'écrire les options sans le signe « - » devant. Par exemple on peut rencontrer :

« `tar cvf archive.tar objets` »

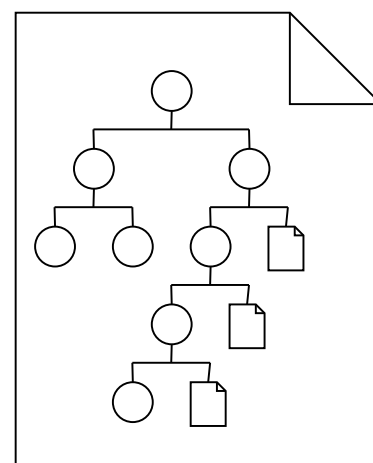


objets

`tar cvf archive.tar objets`

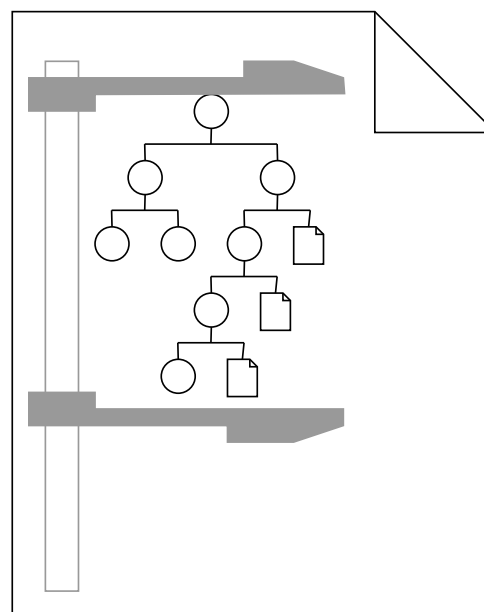
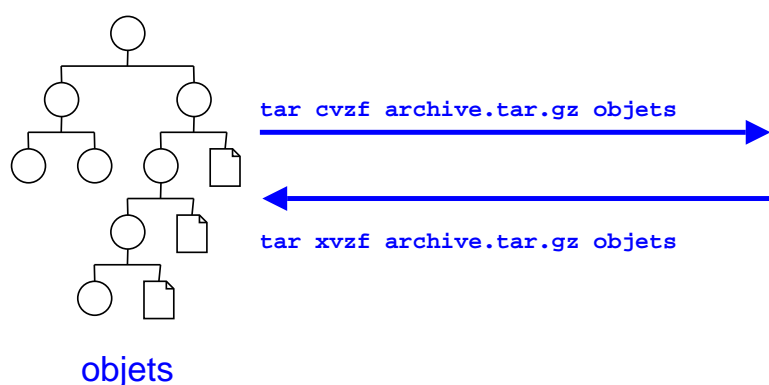


`tar xvf archive.tar objets`



archive.tar

Vous pouvez selon les systèmes UNIX compresser l'archive au fur et à mesure de sa construction :



#### ■ Utilisation de `compress` : option « Z »

- Création d'une archive : `tar cvZf archive.tar.Z objets`
- Affichage du contenu d'une archive : `tar tvZf archive.tar.Z`
- Extraction de l'archive complète : `tar xvZf archive.tar.Z`
- Extraction d'un ou plusieurs objets de l'archive :  
`tar xvZf archive.tar.Z objets`

#### ■ Utilisation de `gzip` : option « z »

- Création d'une archive : `tar cvzf archive.tar.gz objets`
- Affichage du contenu d'une archive : `tar tvzf archive.tar.gz`
- Extraction de l'archive complète : `tar xvzf archive.tar.gz`
- Extraction d'un ou plusieurs objets de l'archive :  
`tar xvzf archive.tar.gz objets`

#### ■ Utilisation de `bzip2` : option « j »

- Création d'une archive : `tar cvjf archive.tar.bz2 objets`
- Affichage du contenu d'une archive : `tar tvjf archive.tar.bz2`
- Extraction de l'archive complète : `tar xvjf archive.tar.bz2`
- Extraction d'un ou plusieurs objets de l'archive :  
`tar xvjf archive.tar.bz2 objets`

Un peu de jargon informatique : un fichier « `.tar.gz` » s'appelle un « *tarball* ».

Extension traditionnelle : « `.tar` »

Extension courante : « `.tgz` » équivalent à « `.tar.gz` »

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.32 • Commandes issues du monde Windows : zip, unzip

Syntaxes de quelques commandes de compression ou décompression :

- `zip [options] fichier`
- `unzip [options] fichier.zip`

URL : <http://www.info-zip.org/>

◇ Exemple 1 : création d'une archive

```
% zip -r archive.zip fichier1 fichier2
adding: fichier1 (deflated 63%)
adding: fichier2 (deflated 66%)
```

◇ Exemple 2 : consultation de la table des matières de l'archive

```
% unzip -l archive.zip
Archive:  archive.zip
Length      Date       Time       Name
-----
   3213  10-25-03  01:50     fichier1
  10371  10-25-03  01:50     fichier2
-----
 13584
                2 files
```

◇ Exemple 3 : extraction du contenu entier de l'archive

```
% unzip archive.zip
Archive:  archive.zip
  inflating: fichier1
  inflating: fichier2
```

◇ Exemple 4 : extraction du contenu partiel de l'archive

```
% unzip archive.zip fichier2
Archive:  archive.zip
  inflating: fichier2
```

**Attention : utiliser l'option « -a » pour extraire des fichiers texte avec conversion des fins de ligne de MSDOS vers UNIX**

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.33 • (Windows : : PowerArchiver Command Line)

Version 3.50 gratuite et téléchargeable sur

<http://www.powerarchiver.com/>

Version 4 payante

- « PComp.exe » compresse sous les formats : ZIP, CAB, LHA, BH (BlakHole), JAR (JavaARchiver), TAR, TAR.GZ (GZIPed TAR), TAR.BZ2 (BZIPed TAR)
- « PExt.exe » extrait les formats : ZIP, RAR, ARJ, CAB, LHA(LZH), ARC, ACE, GZIP, BZIP2, TAR (TAR.GZ, TAR.BZ2), UUE, XXE, ZOO, JAR (JavaARchiver) et autres formats auto-extractibles.

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.34 • Impression : lpr, lpq, lprm

(en anglais *line printer*, *line printer queue*, *line printer remove*)

Une impression nécessite de connaître le nom de l'imprimante et d'avoir un fichier au bon format à imprimer.

Pour imprimer, utiliser la commande `lpr -Pimprimante fichiers`

Pour consulter la queue d'impression, utiliser la commande

`lpq -Pimprimante`

Pour retirer un fichier de la queue d'impression, utiliser la commande

`lprm -Pimprimante numéro-dans-la-queue-renvoyé-par-lpq`



A la formation permanente, les noms des imprimantes sont :

- pièce 213 : « **hp4250-213** »
- pièce 214 : « **hp4100-214** »
- pièce 216 : « **hp4250-216** »
- pièce 217 : « **hp4250-217** »

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.35 • Impression de fichiers texte : a2ps

(en anglais *ascii to postscript*)

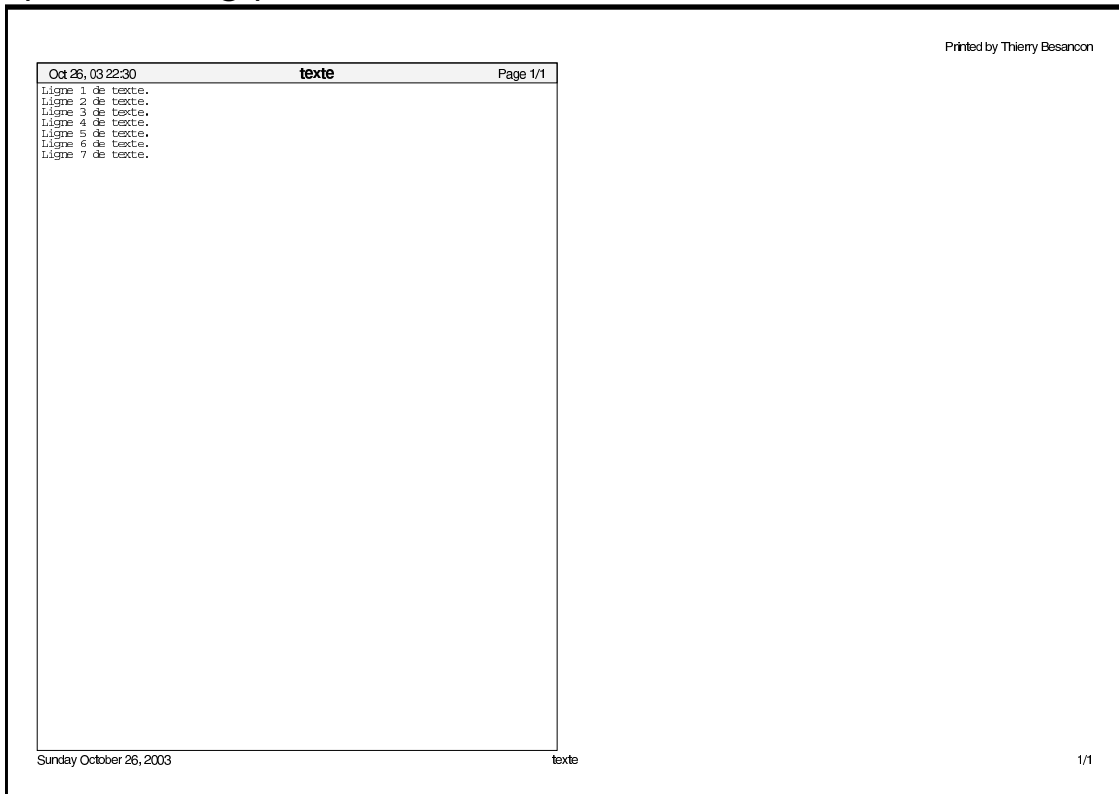
Pour convertir du texte vers le format PostScript compris par l'imprimante. Nombreuses options de la commande a2ps.

Pour imprimer du texte dans la salle de TP de la Formation Permanente :

```
% a2ps -P 216-hp fichier
[a2ps (plain): 1 page on 1 sheet]
[Total: 1 page on 1 sheet] sent to the standard output
```

Disponible à l'URL : <http://www.inf.enst.fr/~demaille/a2ps/>

Exemple de listing produit :



## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.36 • Impression de fichiers texte : `enscript`

(le nom provient d'un logiciel de la marque Adobe aux fonctionnalités reprises par le logiciel GNU qui a repris le nom pour marquer sa compatibilité avec le logiciel original)

Pour convertir du texte vers le format PostScript compris par l'imprimante. Nombreuses options de la commande `enscript`.

Disponible à l'URL : <http://www.iki.fi/~mtr/genscript/>

Exemple de listing produit :

```

10/03/04 23:06:37
fichier.txt 1
ligne 1 du texte
ligne 2 du texte
ligne 3 du texte
ligne 4 du texte
ligne 5 du texte
ligne 6 du texte

```

## Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

### §8.37 • Utilitaires pour disquettes PC : `mtools`, `mcopy`

Sur des machines équipées de lecteur de disquettes, on peut transférer des fichiers depuis et vers leur lecteur de disquette. Un logiciel appelé **mtools** permet d'utiliser les disquettes en offrant des commandes UNIX avec la logique des commandes connues du DOS.

Récupérer le logiciel sur <http://mtools.linux.lu>

La commande de base à utiliser est `mcopy`.

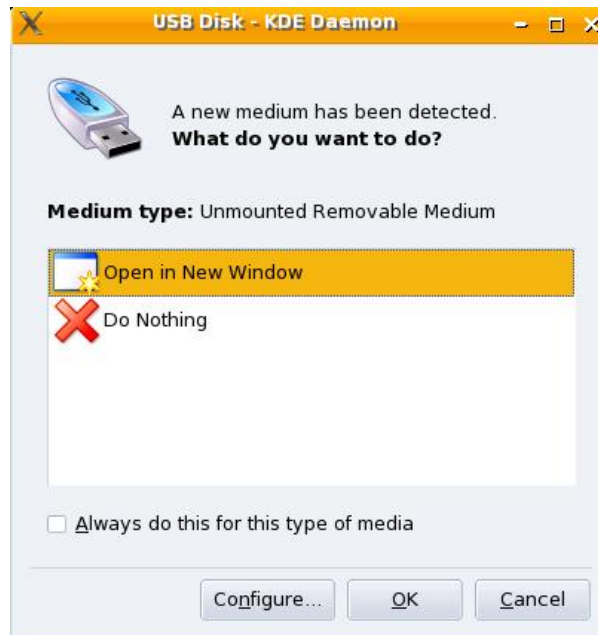
Transfert d'UNIX vers la disquette	<code>mcopy fichier a:</code>
Transfert de la disquette vers UNIX	<code>mcopy a:fichier .</code>
Affichage du contenu de la disquette	<code>mdir a:</code>

Se reporter à l'URL <http://www.loria.fr/~giese/doc/mtools.html> pour plus de détails sur les commandes disponibles.

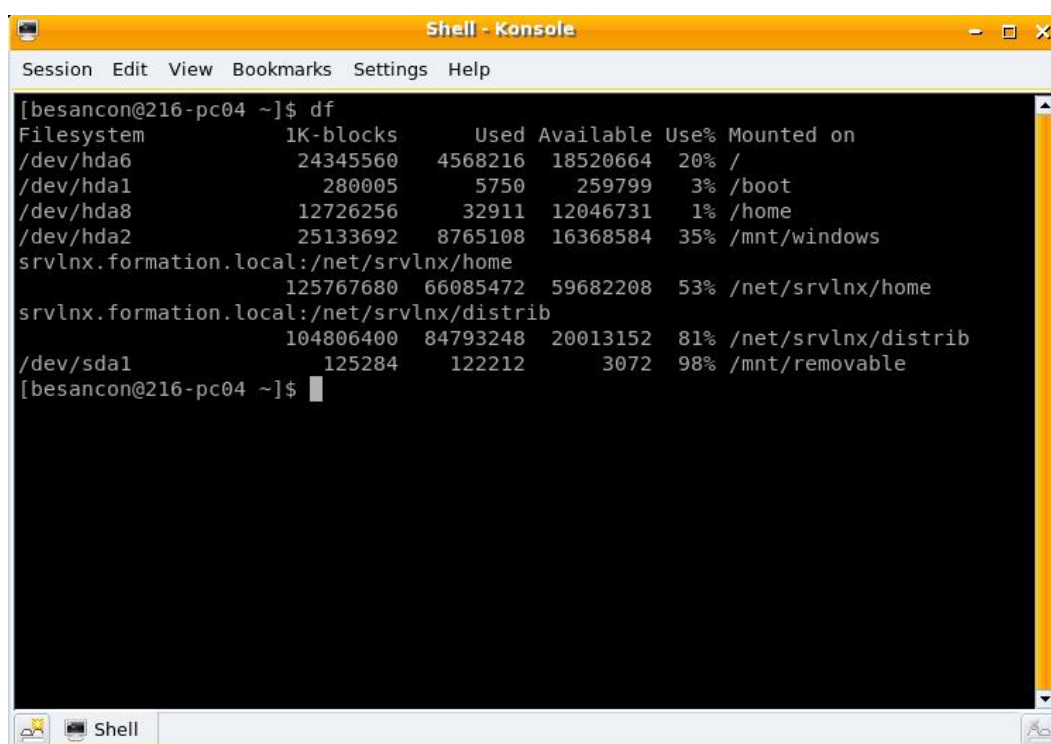
# Chapitre 8 • Commandes de manipulation de base d'objets UNIX (suite)

§8.38 • Utilisation de clefs USB

Les clefs USB sont directement reconnues par le bureau de l'environnement de multifenêtrage KDE de la salle de TP de la Formation Permanente :



La clef apparait en tant qu'icone sur le bureau et en tant que « /mnt/removable » dans l'arborescence des fichiers.



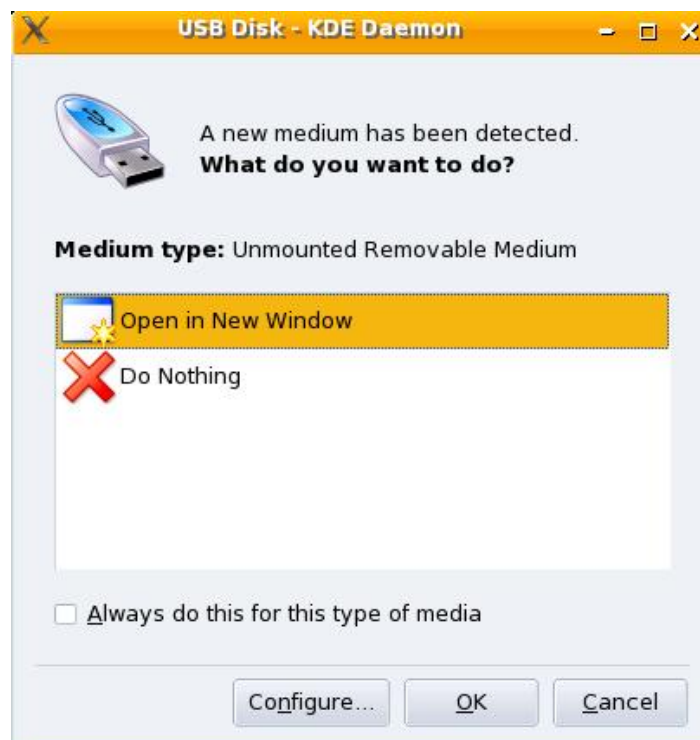
```
[besancon@216-pc04 ~]$ df
Filesystem            1K-blocks    Used Available Use% Mounted on
/dev/hda6              24345560   4568216 18520664 20% /
/dev/hda1              280005     5750   259799  3% /boot
/dev/hda8             12726256   32911 12046731  1% /home
/dev/hda2             25133692   8765108 16368584 35% /mnt/windows
srvlnx.formation.local:/net/srvlnx/home
125767680 66085472 59682208 53% /net/srvlnx/home
srvlnx.formation.local:/net/srvlnx/distrib
104806400 84793248 20013152 81% /net/srvlnx/distrib
/dev/sda1              125284     122212    3072 98% /mnt/removable
[besancon@216-pc04 ~]$
```

La clef USB doit être formatée en FAT ou FAT32.

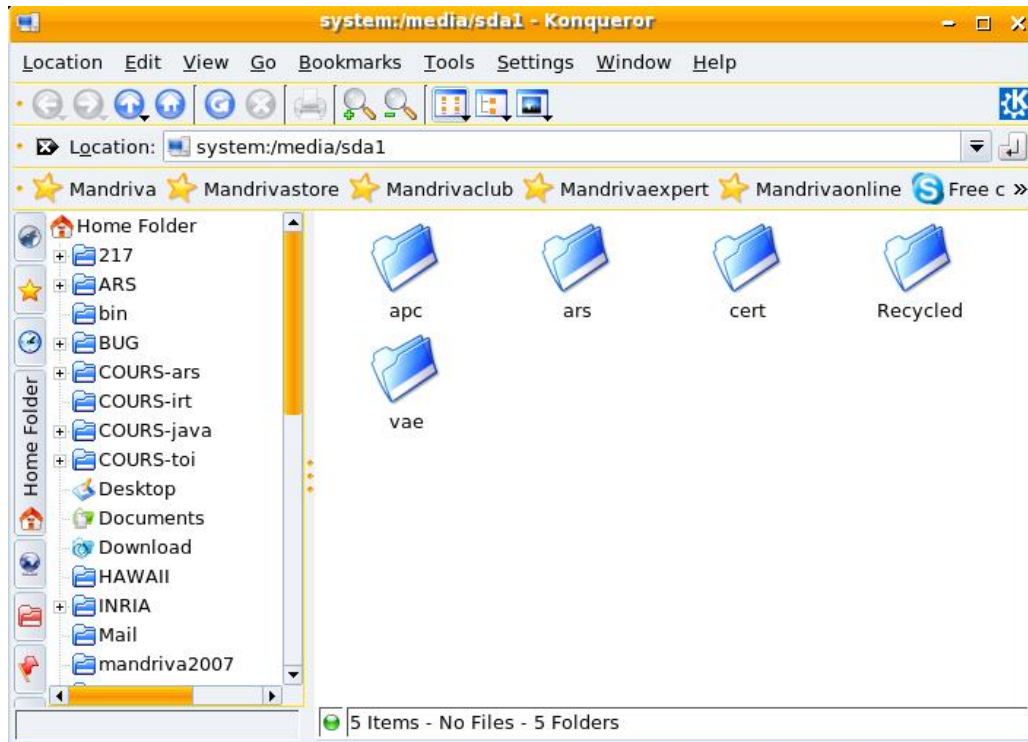
La clef USB doit être formatée en mode disque dur et doit comporter par conséquent une table de partition.

Une clef USB formatée en mode disquette (c'est-à-dire sans table de partition) ne sera pas reconnue par le système LINUX de la salle de TP de la Formation Permanente.

#### ◇ Mandriva 2007 : insertion de la clef



## ◇ Mandriva 2007 : l'explorateur de fichiers s'ouvre



## ◇ Mandriva 2007 : montage automatique de la clef

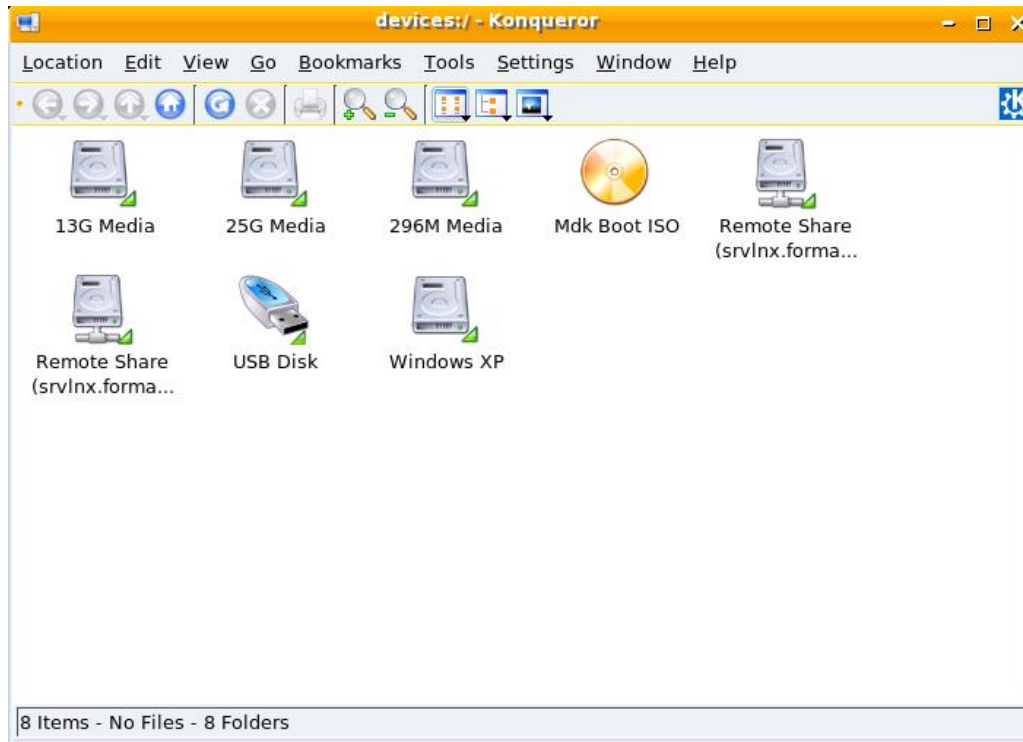
```

[besancon@216-pc04 ~]$ df
Filesystem            1K-blocks    Used Available Use% Mounted on
/dev/hda6              24345560   4568216 18520664 20% /
/dev/hda1               280005      5750   259799   3% /boot
/dev/hda8              12726256   32911 12046731  1% /home
/dev/hda2              25133692   8765108 16368584 35% /mnt/windows
srvlnx.formation.local:/net/srvlnx/home
125767680 66085472 59682208 53% /net/srvlnx/home
srvlnx.formation.local:/net/srvlnx/distrib
104806400 84793248 20013152 81% /net/srvlnx/distrib
/dev/sda1              125284     122212    3072 98% /mnt/removable
[besancon@216-pc04 ~]$

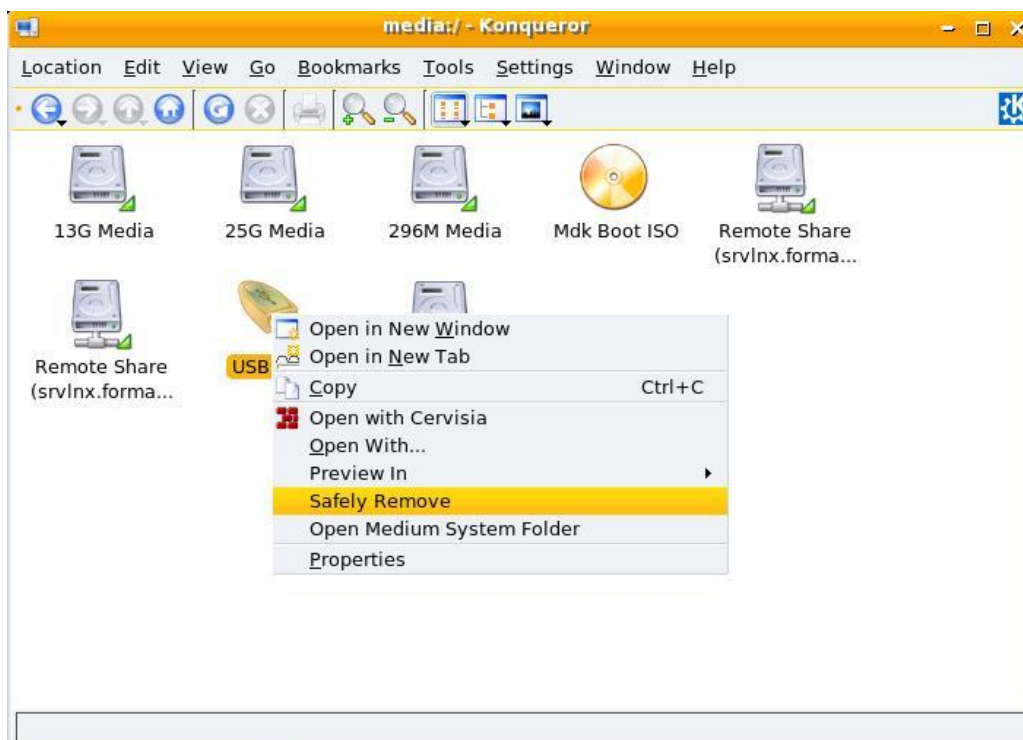
```

La clef est montée en tant que « /mnt/removable ».

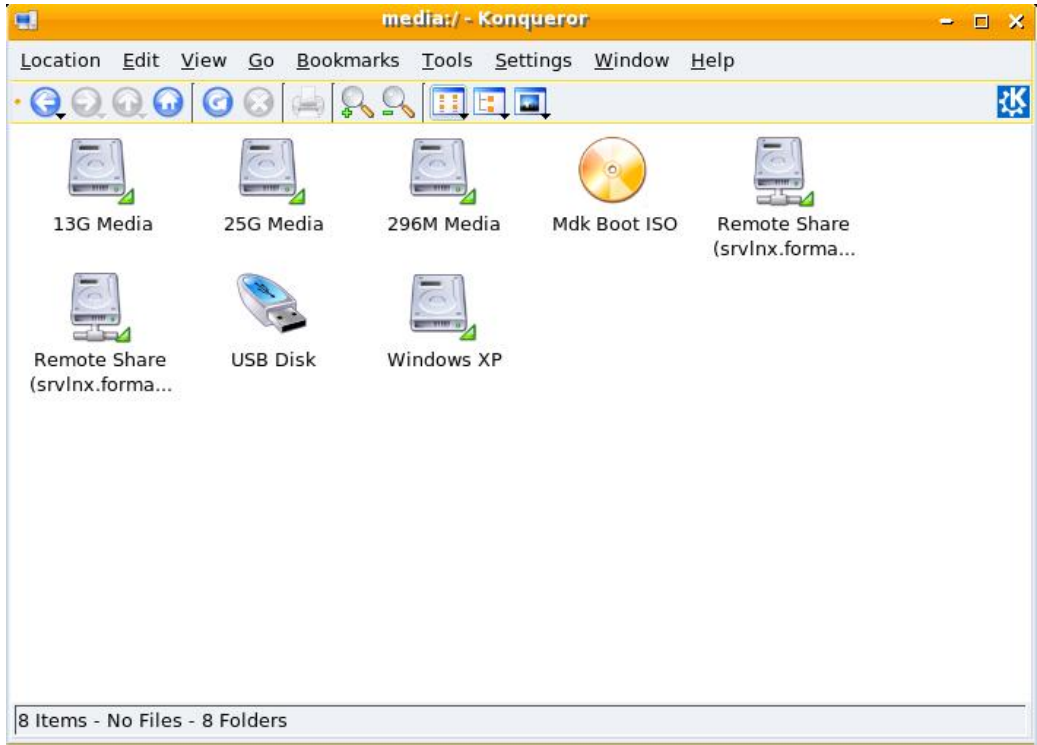
## ◇ Mandriva 2007 : présence de la clef dans les devices



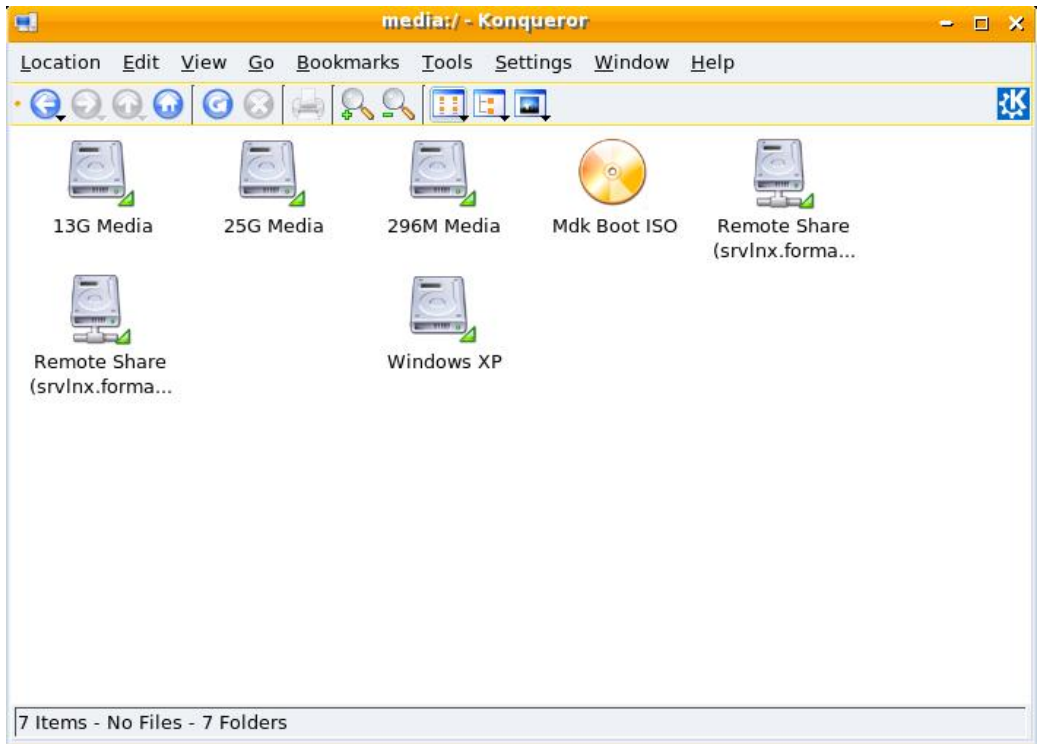
## ◇ Mandriva 2007 : démontage de la clef (1)



◇ Mandriva 2007 : démontage de la clef (2)

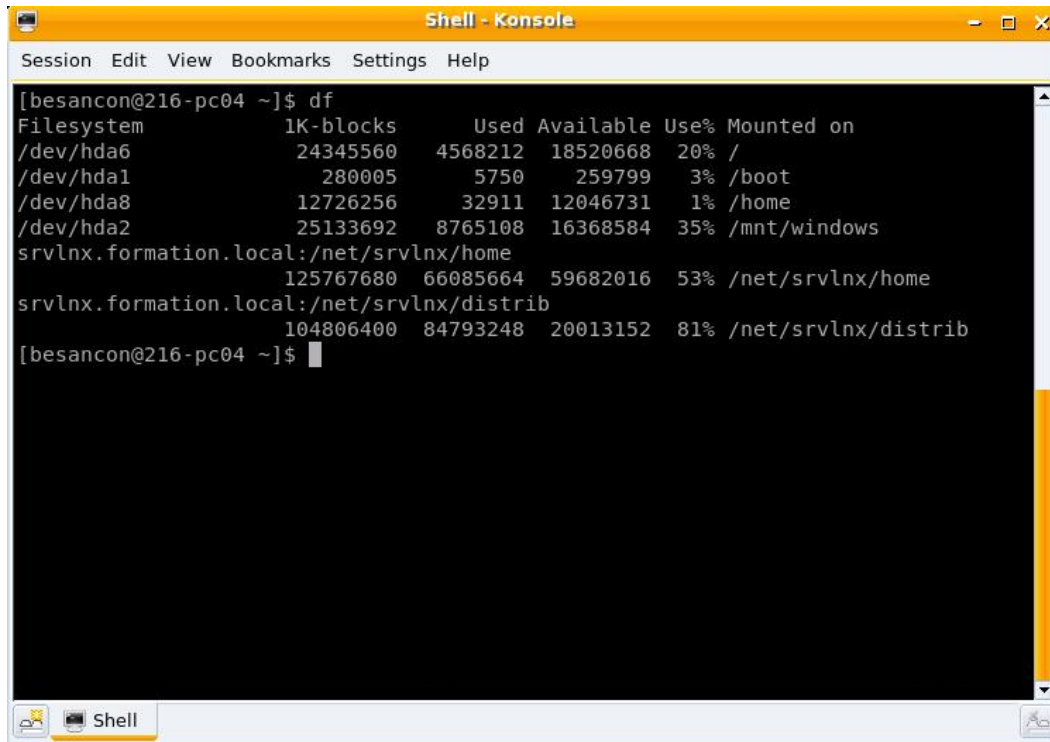


◇ Mandriva 2007 : démontage de la clef (3)





## ◇ Mandriva 2007 : démontage de la clef (4)



```
[besancon@216-pc04 ~]$ df
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/hda6              24345560    4568212  18520668  20% /
/dev/hda1               280005         5750    259799    3% /boot
/dev/hda8              12726256     32911  12046731    1% /home
/dev/hda2              25133692     8765108 16368584   35% /mnt/windows
srvlnx.formation.local:/net/srvlnx/home
125767680    66085664  59682016   53% /net/srvlnx/home
srvlnx.formation.local:/net/srvlnx/distrib
104806400    84793248  20013152   81% /net/srvlnx/distrib
[besancon@216-pc04 ~]$
```

## Chapitre 9

***Attributs des objets UNIX***

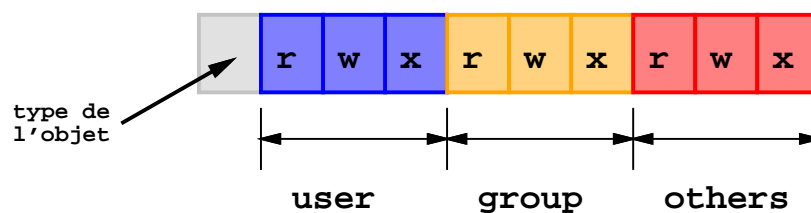
## Chapitre 9 • Attributs des objets UNIX

## §9.1 • Définition des droits d'accès d'un objet

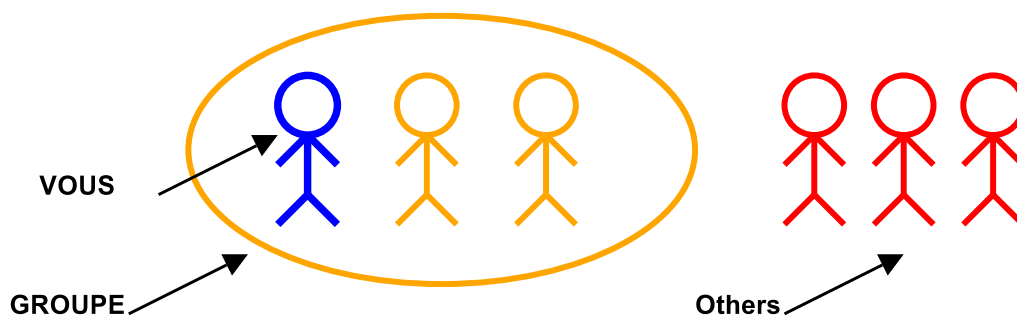
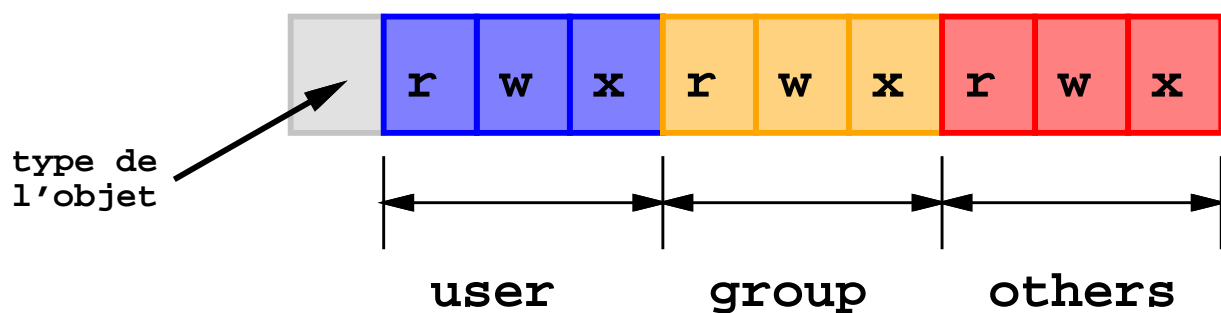
Les droits d'accès à un objet sont stockés dans une structure dite *inode*. Cette structure n'est pas manipulable directement.

Les droits d'accès des objets sont indiqués dans les 10 premiers caractères de chaque ligne affichée par « `ls -l` » :

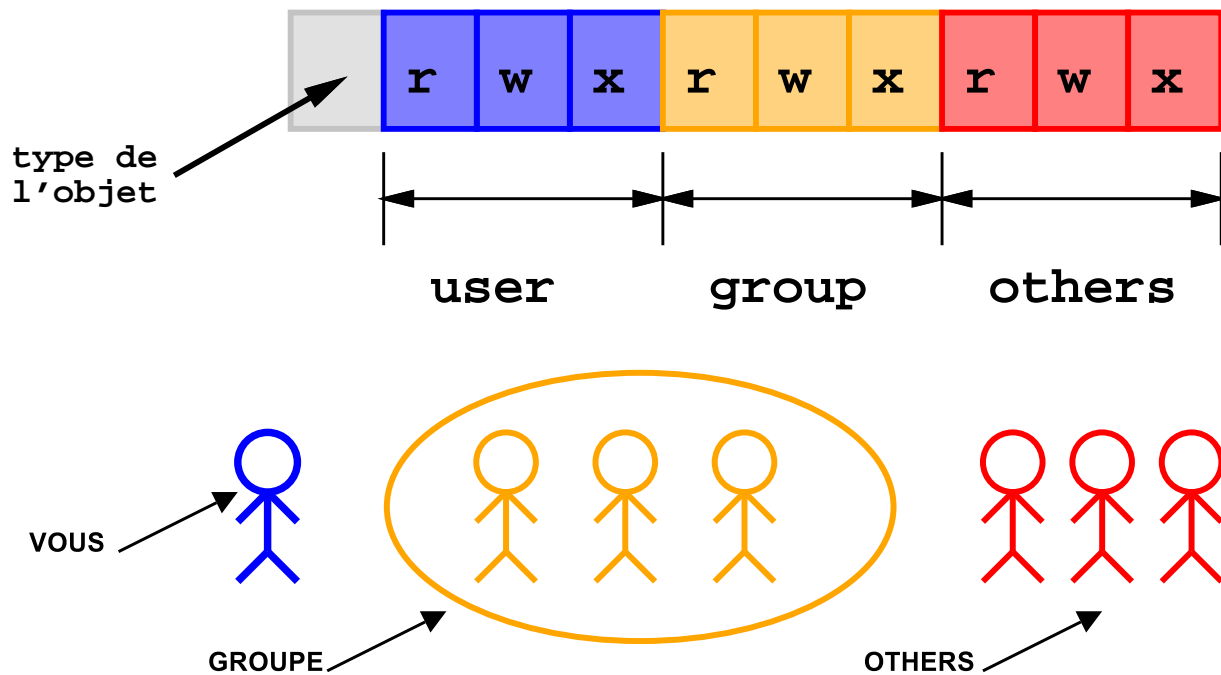
```
% ls -l
total 16
-rw-r--r-- 1 besancon ars      15524 Sep 15 15:17 exemple.txt
```



Cas le plus courant :

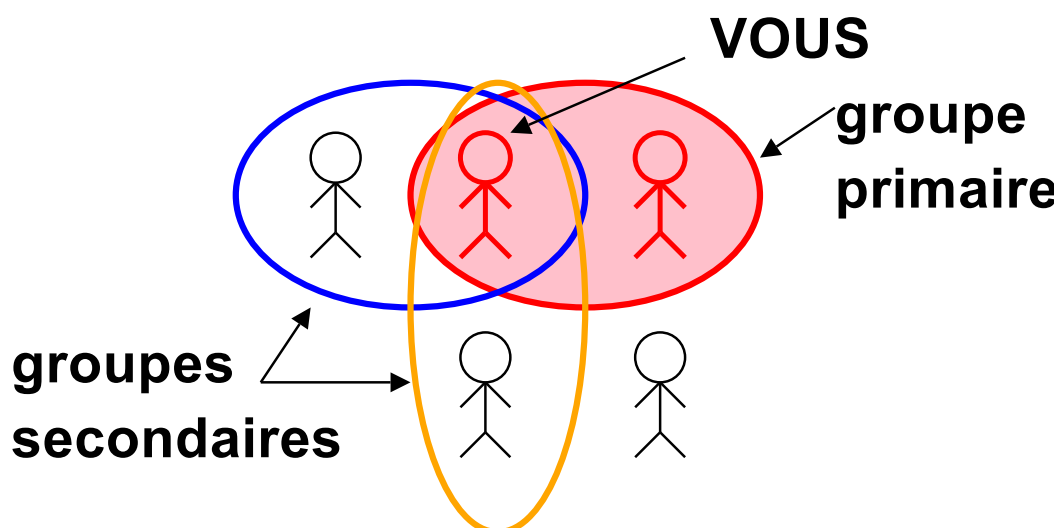


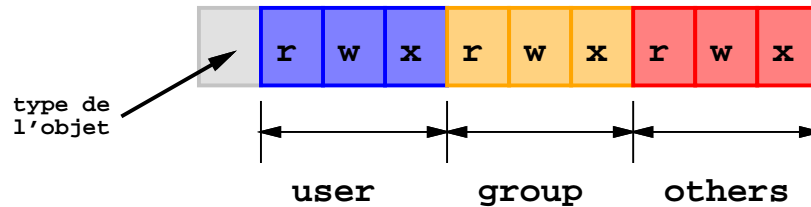
Cas moins courant mais possible :



Le principe :

- Un utilisateur appartient à un groupe primaire
- Un utilisateur peut appartenir à des groupes secondaires
- Un objet a un propriétaire utilisateur et un propriétaire groupe





Il existe trois droits d'accès associés à chaque objet :

- droits du propriétaire ( $u \equiv \text{user}$ )
- droits des membres du groupe ( $g \equiv \text{group}$ )
- droits des autres utilisateurs ( $o \equiv \text{others}$ )

Il existe trois types de permissions :

- droit en lecture ( $r \equiv \text{read}$ )
- droit en écriture ( $w \equiv \text{write}$ )
- droit en exécution ( $x \equiv \text{execute access}$ )

## Chapitre 9 • Attributs des objets UNIX

### §9.2 • Changements des droits d'accès d'un objet : `chmod`

(en anglais *change modes*)

Syntaxe : `chmod [options] modes objets`

Option « `-R` » pour changer récursivement les droits des objets d'une arborescence.

La précision des modes dans la commande peut prendre deux formes :

- forme symbolique :
  - « `u` » (*user*), « `g` » (*group*), « `o` » (*others*) ou « `a` » (*all*)
  - « `+` » ou « `-` » ou « `=` »
  - permissions (`r`, `w` ou `x`)
- forme numérique :
  - Les permissions sont exprimées en base huit ou octale.
  - Par exemple : `rwX r-X r-X`  $\equiv$  `755`

Droits	Valeur base 2	Valeur base 8
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwX	111	7

C'est pourquoi on a par exemple :

`rwX r-x r-x`  $\equiv$  755

`rw- r-- r--`  $\equiv$  644

`rw- --- ---`  $\equiv$  600

```
% ls -lg exemple.txt
-rw-r--r--  1 besancon ars      249 Sep 20 22:43 exemple.txt

% chmod g+w exemple.txt

% ls -lg exemple.txt
-rw-rw-r--  1 besancon ars      249 Sep 20 22:43 exemple.txt

% chmod o=wx exemple.txt

% ls -lg exemple.txt
-rw-rw--wx  1 besancon ars      249 Sep 20 22:43 exemple.txt

% chmod 640 exemple.txt

% ls -lg exemple.txt
-rw-r-----  1 besancon ars      249 Sep 20 22:43 exemple.txt
```

## Chapitre 9 • Attributs des objets UNIX

§9.3 • Droits d'accès par défaut lors de création d'objets : `umask`

(en anglais *user mask*)

2 syntaxes possibles :

- connaître les droits par défaut lors de la création d'objets :

Syntaxe : `umask`

```
% umask
```

```
022
```

- positionner les droits d'accès par défaut :

Syntaxe : `umask [modes-par-défaut]`

**On utilise une notation octale : on indique les bits qui ne seront pas positionnés lors de la création des objets.**

Droits par défaut	Valeur base 2	Valeur base 8
---	111	7
--x	110	6
-w-	101	5
-wx	100	4
r--	011	3
r-x	010	2
rw-	001	1
rwX	000	0

On veut par défaut les droits `rwX r-x r-x`  $\equiv$  `umask 022`

On veut par défaut les droits `rw- r-- r--`  $\equiv$  `umask 133`

On veut par défaut les droits `rw- --- ---`  $\equiv$  `umask 177`

Le mode paranoïaque  $\equiv$  droits par défaut `rwX --- ---`  $\equiv$  `umask 077`

**ATTENTION** : le réglage de umask indique les bits autorisés au mieux, il n'indique pas les bits qui seront obtenus.

Voir exemples ci après.

### ◇ Exemple 1

```
% umask
```

```
022
```

```
% vi prog.c
```

```
% ls -l prog.c
```

```
-rw-r--r--  1 besancon ars          127 Oct 12 14:45 prog.c
```

```
% gcc prog.c -o prog.exe
```

```
% ls -l prog.exe
```

```
-rwxr-xr-x  1 besancon ars        6076 Oct 12 14:45 prog.exe
```

### ◇ Exemple 2

```
% umask 027

% vi prog.c

% ls -l prog.c
-rw-r----- 1 besancon ars      127 Oct 12 14:45 prog.c

% gcc prog.c -o prog.exe

% ls -l prog.exe
-rwxr-x---  1 besancon ars      6076 Oct 12 14:45 prog.exe
```

### ◇ Exemple 3

```
% umask 000

% vi prog.c

% ls -l prog.c
-rw-rw-rw-  1 besancon ars      127 Oct 12 14:45 prog.c

% gcc prog.c -o prog.exe

% ls -l prog.exe
-rwxrwxrwx  1 besancon ars      6076 Oct 12 14:45 prog.exe
```



**ATTENTION : un réglage de umask dure le temps d'une session shell !** (voir plus loin comment rendre le réglage permanent)

**ATTENTION : le umask de l'administrateur doit être 022 au pire, 077 au mieux !**

## Chapitre 9 • Attributs des objets UNIX

### §9.4 • Régler le umask de façon permanente

Hypothèse : on est sous Bourne Shell ou sous BASH.

Objectif : on veut régler son « umask » de façon permanente.

Solution :

- On utilise le fichier « `$HOME/.profile` » avec un lien symbolique « `$HOME/.bashrc` » dessus.  
On règle ainsi (sh ou bash) :
  - cas du shell interactif de login
  - cas du shell interactif non de login
  - cas du shell non interactif
- On ajoute dans le fichier « `$HOME/.profile` »  
`umask 022`

## Chapitre 9 • Attributs des objets UNIX

### §9.5 • Attribut spécial de fichier : bit setuid

Il existe un attribut spécial de fichier réservé à la gestion du système : le bit **setuid** (4000 en octal).

Avec ce bit positionné, le programme est exécuté avec les droits de l'utilisateur propriétaire.

```
% ls -lg prog1.exe prog2.exe
-rwxr-xr-x  1 besancon ars      249 Sep 20 22:43 prog1.exe
-rwxr-xr-x  1 besancon ars      249 Sep 20 22:43 prog2.exe
% chmod u+s prog1.exe
% chmod 4711 prog2.exe
% ls -lg prog1.exe prog2.exe
-rwsr-xr-x  1 besancon ars      249 Sep 20 22:43 prog1.exe
-rws--x--x  1 besancon ars      249 Sep 20 22:43 prog2.exe
```

Attention à l'affichage du bit setuid !  
Classiquement :

```
% gcc prog.c -o prog.exe
% ls -l prog.exe
-rwxr-xr-x  1 besancon ars      6204 Jan 24 20:22 prog.exe
% chmod u+s prog.exe
% ls -l prog.exe
-rwsr-xr-x  1 besancon ars      6204 Jan 24 20:22 prog.exe
```

Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          127 Jan 24 20:21 exemple.txt
% chmod u+s exemple.txt
chmod: WARNING: exemple.txt: Execute permission required for
set-ID on execution
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          127 Jan 24 20:21 exemple.txt
% chmod 4644 exemple.txt
% ls -l exemple.txt
-rwSr--r--  1 besancon ars          127 Jan 24 20:21 exemple.txt
```

Bref :

- affichage « S » ≡ « bit 04000 seul »
- affichage « s » ≡ « bit x + bit S »

**ATTENTION : le bit setuid ne fonctionne pas avec un shell script.** (voir page 560 pour ce qu'est un shell script)

**Il ne fonctionne qu'avec un exécutable binaire.**

Une solution sera proposée dans le tôme 3 (commande « sudo »).

## Chapitre 9 • Attributs des objets UNIX

## §9.6 • Attribut spécial de fichier : bit setgid

Il existe un attribut spécial de fichier réservé à la gestion du système : le bit **setgid** (**2000** en octal).

Avec ce bit positionné, le programme est exécuté avec les droits du groupe propriétaire

```
% ls -lgF prog1.exe prog2.exe
-rwxr-xr-x  1 besancon ars      249 Sep 20 22:43 prog1.exe
-rwxr-xr-x  1 besancon ars      249 Sep 20 22:43 prog2.exe
% chmod g+s prog1.exe
% chmod 2711 prog2.exe
% ls -lgF prog1.exe prog2.exe
-rwxr-sr-x  1 besancon ars      249 Sep 20 22:43 prog1.exe
-rwx--s--x  1 besancon ars      249 Sep 20 22:43 prog2.exe
```

Attention à l'affichage du bit setgid !

Classiquement :

```
% gcc prog.c -o prog.exe
% ls -l prog.exe
-rwxr-xr-x  1 besancon ars      6204 Jan 24 20:22 prog.exe
% chmod g+s prog.exe
% ls -l prog.exe
-rwxr-sr-x  1 besancon ars      6204 Jan 24 20:22 prog.exe
```

Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          127 Jan 24 20:21 exemple.txt
% chmod g+s exemple.txt
chmod: WARNING: exemple.txt: Execute permission required for set-:
on execution
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          127 Jan 24 20:21 exemple.txt
% chmod 2644 exemple.txt
% ls -l exemple.txt
-rw-r-lr--  1 besancon ars          127 Jan 24 20:21 exemple.txt
```

Bref :

- affichage « l » ≡ « bit 02000 seul »
- affichage « s » ≡ « bit x + bit l »

**ATTENTION : le bit setgid ne fonctionne pas avec un shell script.** (voir page 560 pour ce qu'est un shell script)

**Il ne fonctionne qu'avec un exécutable binaire.**

Une solution sera proposée dans le tôme 3 (commande « sudo »).

## Chapitre 9 • Attributs des objets UNIX

## §9.7 • Attribut spécial de répertoire : sticky bit

Il existe un attribut spécial de répertoire réservé à la gestion du système : le **sticky bit** (**1000** en octal).

Avec ce bit positionné, on ne peut effacer d'un répertoire que ses propres fichiers et pas ceux des autres.

Exemple d'utilisation : le répertoire système de stockage des fichiers temporaires « /tmp »

```
% ls -ld /tmp
drwxrwxrwt 11 root root      2580 Aug  2 15:40 /tmp
% cd /tmp
% ls -l
-rw-r--r--  1 besancon adm      90252 Jul 29 16:21 outlook-express-0042.
-rw-r--r--  1 besancon adm     255596 Jul 29 16:21 outlook-express-0049.
-rw-r--r--  1 root      root        639 Sep  1 23:44 pear.con
-rw-r--r--  1 root      root        614 Sep  1 23:09 php.errors
-rw-r--r--  1 apache   apache   35383 Aug  8 01:49 ps
% rm ps
rm: ps: override protection 644 (yes/no)? y
rm: ps not removed: Permission denied
% rm php.errors
rm: php.errors: override protection 644 (yes/no)? y
rm: php.errors not removed: Permission denied
% ls -l
...
-rw-r--r--  1 root      root        614 Sep  1 23:09 php.errors
-rw-r--r--  1 apache   apache   35383 Aug  8 01:49 ps
...
```

Historiquement : le sticky bit positionné sur un exécutable le chargeait en mémoire virtuelle et ne l'effaçait pas de la zone de swap si bien que le recharger se faisait rapidement.

Mécanisme abandonné (avant 1990).

Bit libre récupéré pour le mécanisme connu maintenant.

Attention à l'affichage du sticky bit !  
Classiquement :

```
% mkdir -p exemple.dir
% ls -ld exemple.dir
drwxr-xr-x  2 besancon ars          117 Jan 25 11:09 exemple.dir
% chmod 1777 exemple.dir
% ls -lgd exemple.dir
drwxrwxrwt  2 besancon ars          117 Jan 25 11:09 exemple.dir
```

Moins classiquement :

```
% mkdir -p exemple.dir
% ls -ld exemple.dir
drwxr-xr-x  2 besancon ars          117 Jan 25 11:09 exemple.dir
% chmod 1700 /tmp/exemple.dir
% ls -lgd /tmp/exemple.dir
drwx-----T  2 besancon ars          117 Jan 25 11:12 exemple.dir
```

Bref :

- affichage « T »  $\equiv$  « bit 01000 seul »
- affichage « t »  $\equiv$  « bit x + bit T »

## Chapitre 9 • Attributs des objets UNIX

### §9.8 • Attributs de date d'un objet : mtime, atime, ctime

**Sur UNIX, à chaque objet sont associées 3 dates stockées dans une structure dite *inode*. Cette structure n'est pas manipulable directement.**

Ces 3 dates sont :

- date de dernière modification dite *mtime* ( $\equiv$  modification time)
- date de dernier accès dite *atime* ( $\equiv$  access time)
- date de dernière modification des attributs dite *ctime* ( $\equiv$  change time)

Ces dates sont affichables via la commande « `ls` ».



**Attention** : par défaut, « `ls -l` » affiche

- une date de moins de 6 mois sous la forme : « Mois Jour Heure :Minute »
- une date de plus de 6 mois sous la forme : « Mois Jour Année »

```
% ls -l
drwxr-xr-x   3 besancon ars      1024 Oct 23  2005 jardin
-rw-r--r--   1 besancon ars     29749 Apr  5 20:50 ananas.avi
```

Sur Solaris, option « `-e` » pour un affichage normalisé :

```
% ls -e
drwxr-xr-x   3 besancon ars      1024 Oct 23 19:35:51 2005 jardin
-rw-r--r--   1 besancon ars     29749 Apr  5 20:50:28 2006 ananas.av
```

Au niveau de la commande « `ls` » :

- option « `-l` » : format long (affichage du mtime par défaut)
- option « `-t` » : tri décroissant par date (mtime par défaut)
- option « `-r` » : tri par ordre inverse
- « `ls -lt` » : classement par ordre chronologique décroissant des mtimes
- « `ls -ltu` » : classement par ordre chronologique décroissant des atimes
- « `ls -ltc` » : classement par ordre chronologique décroissant des ctimes

Syntaxe : `date [options] [+format]`

Quelques cas utiles :

- « `date` » : la date de l'instant courant
- « `date -u` » : la date GMT de l'instant courant
- « `date '+%Y%m%d'` » : date du jour sous la forme « AAAAMMJJ »  
Voir page de manuel de la fonction C « `strftime` »

Syntaxe : `touch [options] [-t time] objet`

Quelques options :

- option « `-a` » : modification de la date d'accès de l'objet (`a` ≡ `atime`)
- option « `-m` » : modification de la date de modification de l'objet (`m` ≡ `mtime`)
- option « `-t time` » : indique une date autre que la date du moment à mettre ;  
format : « AAAAMMJJhhmm.ss »

## ◇ Exemple

```
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          49 Sep 27 13:07 exemple.txt

% touch -m -t 199901012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          49 Jan  1  1999 exemple.txt

% touch -m -t 09012233 exemple.txt
% ls -l exemple.txt
-rw-r--r--  1 besancon ars          49 Sep  1 22:33 exemple.txt
```

## Chapitre 10

# *Expressions régulières et commandes UNIX associées*

Expression régulière = regular expression = regexp

Besoin pratique : faire des recherches dans des fichiers d'enregistrements de base de données, d'inventaires, de comptabilité ... Bref, des fichiers ayant souvent une **structure forte**.

Pour cela, plusieurs programmes UNIX utilisent des **critères** de reconnaissance de motifs de chaînes de caractères.

Un exemple de motif :

*« les lignes commençant par la lettre a »*

**Une expression régulière est la traduction en langage UNIX du motif de recherche**, ce qui permettra de le reconnaître au sein d'un texte dans un fichier.

Ainsi le motif précédent donne la regexp :

`^a`

La regexp traduit le motif à rechercher sous la forme d'une **suite de contraintes à satisfaire toutes**.

Pour construire les contraintes, on dispose de différents types d'écritures :

- écritures décrivant des valeurs de caractères
- écritures décrivant des positions de caractères
- écritures décrivant des répétitions de caractères

### ◇ Contraintes sur des valeurs de caractères

caractère	désigne la contrainte pour un caractère d'être ce caractère
[caractères]	désigne la contrainte pour un caractère d'être parmi la liste indiquée
[^caractères]	désigne la contrainte pour un caractère de ne pas figurer dans la liste indiquée
.	désigne la contrainte pour un caractère d'être quelconque

### Rappel sur les codes ASCII :

Faire « man ascii »

0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	AC
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	S
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SY
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	R
32	SP	33	!	34	"	35	#	36	\$	37	%	38	&
40	(	41	)	42	*	43	+	44	,	45	-	46	.
48	0	49	1	50	2	51	3	52	4	53	5	54	(
56	8	57	9	58	:	59	;	60	<	61	=	62	>
64	@	65	A	66	B	67	C	68	D	69	E	70	F
72	H	73	I	74	J	75	K	76	L	77	M	78	N
80	P	81	Q	82	R	83	S	84	T	85	U	86	V
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^
96	`	97	a	98	b	99	c	100	d	101	e	102	f
104	h	105	i	106	j	107	k	108	l	109	m	110	n
112	p	113	q	114	r	115	s	116	t	117	u	118	v
120	x	121	y	122	z	123	{	124		125	}	126	~

## Exemples :

- « [a-z] » : le caractère peut prendre une valeur entre « a » et « z »
- « [A-Z] » : le caractère peut prendre une valeur entre « A » et « Z »
- « [0-9] » : le caractère peut prendre une valeur entre « 0 » et « 9 »
- « [a-zA-Z0-9] » = « [A-Za-z0-9] » = « [0-9A-Za-z] » (8 combinaisons au total ; peu importe l'ordre des trois intervalles) : le caractère peut prendre une valeur entre « a » et « z » ou entre « A » et « Z » ou entre « 0 » et « 9 »
- « [a-z-] » : le caractère peut prendre une valeur entre « a » et « z » ou la valeur « - » en plus
- « [a-z\[ \]] » : le caractère peut prendre une valeur entre « a » et « z » ou les valeurs « [ » ou « ] » en plus
- « [a-z\\] » : le caractère peut prendre une valeur entre « a » et « z » ou la valeur « \ » en plus
- « [^a-z] » : les valeurs entre « a » et « z » sont interdites pour le caractère ; le caractère vaudra tout sauf une minuscule

◇ Contraintes sur les positions de caractères

^	désigne la contrainte d'être en début de ligne
\$	désigne la contrainte d'être en fin de ligne

### ◇ Répétitions d'une contrainte

{min,max}	indique que la contrainte mentionnée juste avant doit être satisfaite entre min et max fois
*	indique que la contrainte mentionnée juste avant doit être satisfaite autant de fois que possible en pratique (de 0 à autant que l'on veut)

Par exemple :

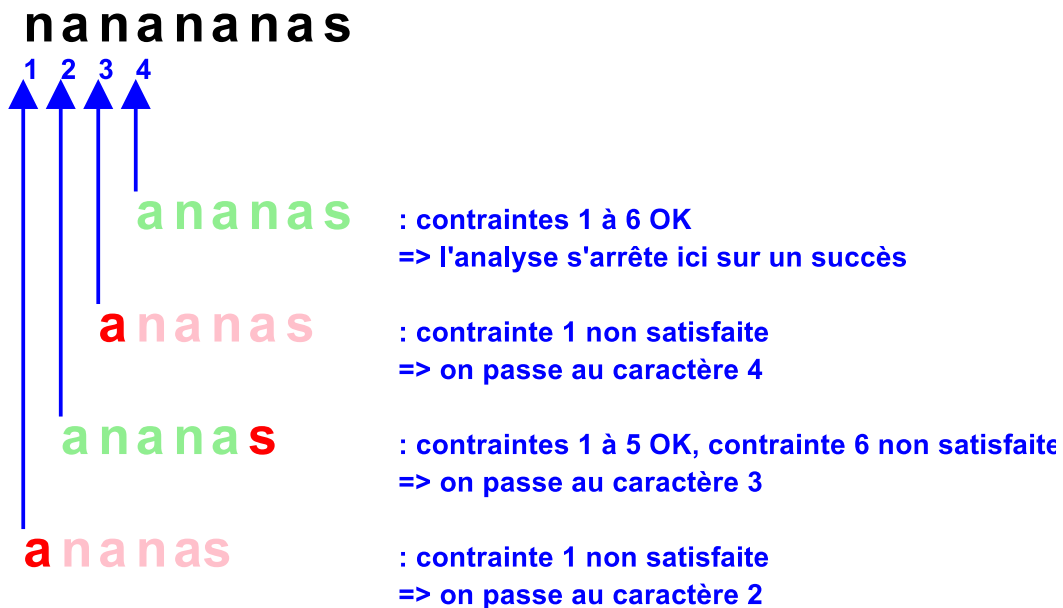
- « {3,7} » : contrainte précédente répétée entre 3 et 7 fois
- « {3,} » : contrainte précédente répétée au moins 3 fois
- « {5} » : contrainte précédente répétée 5 fois
- « {1} » : contrainte précédente répétée 1 fois ; cette écriture n'a pas d'intérêt : autant écrire la contrainte précédente simplement toute seule

## ATTENTION : principes FONDAMENTAUX des regexps

- on analyse chaque ligne indépendamment de la précédente et de la suivante (sauf mention contraire mais rarissime)
- on analyse de gauche à droite, caractère par caractère en cherchant si l'on vérifie la regexp ; passage au caractère suivant à droite si l'on ne vérifie pas la regexp sur la position courante
- via la regexp, **on essaye de vérifier (matcher) la chaîne de caractères la plus longue possible**
- une contrainte peut être rendue muette selon le contexte pour satisfaire la regexp au total

Analyse de la gauche vers la droite

Exemple : rechercher le mot « ananas » dans le texte suivant :



Application des regexps :

- sélectionner des lignes de fichiers texte qui satisfont la regexp  
⇒ c'est l'objet de la commande UNIX « **grep** »
- faire des remplacements du texte satisfaisant la regexp par un autre texte qui peut-être déduit du texte initial  
⇒ c'est l'objet de la commande UNIX « **sed** »



## ◇ Exemple 1

Le motif à satisfaire :

*la ligne contient le mot elephant*

Traduction en regexp :

elephant

En effet, c'est une suite de 8 contraintes à satisfaire toutes :

- contrainte 1 : un caractère doit valoir « e »
- contrainte 2 : un caractère doit valoir « l »
- contrainte 3 : un caractère doit valoir « e »
- contrainte 4 : un caractère doit valoir « p »
- contrainte 5 : un caractère doit valoir « h »
- contrainte 6 : un caractère doit valoir « a »
- contrainte 7 : un caractère doit valoir « n »
- contrainte 8 : un caractère doit valoir « t »

## ◇ Exemple 2

Le motif à satisfaire :

*la ligne se termine par 2 caractères en minuscule suivis d'un chiffre*

Traduction en regexp :

[a-z][a-z][0-9]\$

ou

[a-z]{2}[0-9]\$

En effet, c'est une suite de 4 contraintes à satisfaire toutes :

- contrainte 1 : un caractère doit valoir une lettre minuscule, soit « [a-z] »
- contrainte 2 : un caractère doit valoir une lettre minuscule, soit « [a-z] »
- contrainte 3 : un caractère doit valoir un chiffre soit « [0-9] »
- contrainte 4 : être en fin de ligne soit « \$ »

## ◇ Exemple 3

Le motif à satisfaire :

« mot1: mot2: mot3: » en début de ligne où

- mot1 commence par une lettre majuscule
- mot2 se termine par un chiffre
- mot3 est quelconque

Processus de traduction :

- 1 « en début de ligne »  
⇒ la regexp est de la forme « ^ »
- 2 « mot1 commence par une lettre majuscule »  
⇒ mot1 commence par une lettre majuscule et le reste des lettres est quelconque  
⇒ mot1 commence par une lettre majuscule et le reste des lettres est quelconque sans pour autant valoir le caractère : qui sépare les mots  
⇒ la regexp est de la forme « [A-Z] [^:]\* »
- 3 « mot1 : mot2 »  
⇒ la regexp est de la forme « : »

- 4 « mot2 se termine par un chiffre »  
⇒ mot2 commence par des caractères quelconques et se termine par un chiffre  
⇒ mot2 commence par des caractères quelconques sans pour autant valoir : qui sépare les mots et se termine par un chiffre  
⇒ la regexp est de la forme « [^:]\* [0-9] »
- 5 « mot2 : mot3 : »  
⇒ la regexp est de la forme « : »
- 6 « mot3 est quelconque »  
⇒ mot3 est composé de caractères quelconques  
⇒ mot3 est composé de caractères quelconques sans pour autant valoir le caractère : qui sépare les mots  
⇒ la regexp est de la forme « [^:]\* »
- 7 « mot3 : »  
⇒ la regexp est de la forme « : »

Résultat final, assemblage des résultats intermédiaires :

$^ [A-Z] [^:]* : [^:]* [0-9] : [^:]* :$

◇ Exemple 4

Soit la regexp :

```
^[^abc]*
```

Soit le fichier contenant les lignes suivantes :

```
ascenceur
berceau
chameau
elephant
```

La regexp sélectionne les lignes suivantes :

```
ascenceur
berceau
chameau
elephant
```

Pourquoi ?

**Principe de rendre muette une contrainte pour satisfaire la regexp globalement**

◇ Exemple 5

Soit la regexp :

```
(cerise|ananas)
```

Cette regexp permet de chercher les lignes contenant le mot « ananas » ou « cerise ».

Impossible à traduire avec les constructions basiques de la page 330.

On parle de « *extended regexp* ».

Forme générale : « (regexp1|regexp2|...|regexpN) »

(le mot grep vient de « g/re/p » dans vi)

Syntaxe : `grep [options] regexp fichiers`

Quelques options intéressantes :

- option « `-i` » : pas de différenciation entre lettres minuscules et majuscules
- option « `-n` » : affichage des numéros de ligne
- option « `-l` » : n'affiche que les noms de fichiers
- option « `-v` » : affichage des lignes ne contenant pas la chaîne précisée

### ◇ Exemple 1

Soit le fichier :

```
Ecrivons toto en minuscules ici.  
Et ici ToTo en minuscules et majuscules.  
Mais là on ne met pas la regexp de l'exemple.
```

On voit :

```
% grep toto exemple.txt  
Ecrivons toto en minuscules ici.
```

### ◇ Exemple 2

Soit le fichier :

```
Ecrivons toto en minuscules ici.  
Et ici ToTo en minuscules et majuscules.  
Mais là on ne met pas la regexp de l'exemple.
```

On voit :

```
% grep -in toto exemple.txt  
1:Ecrivons toto en minuscules ici.  
2:Et ici ToTo en minuscules et majuscules.
```

### ◇ Exemple 3

Soit le fichier :

```
Ecrivons toto en minuscules ici.  
Et ici ToTo en minuscules et majuscules.  
Mais là on ne met pas la regexp de l'exemple.
```

On voit :

```
% grep -inv toto exemple.txt  
3:Mais là on ne met pas la chaîne de l'exemple.
```

En pratique :

- écrire la regexp comprise entre deux apostrophes (l'explication des apostrophes sera vue au niveau du cours sur la pratique du shell)
- sur Solaris :  
« `egrep [options] 'regexp' fichiers` »
- sur LINUX :  
« `grep -E [options] 'regexp' fichiers` »

### ◇ Options LINUX non standard

- Option « `-A nombre` » : affiche *nombre* de lignes après la regexp (en anglais **after**)
- Option « `-B nombre` » : affiche *nombre* de lignes avant la regexp (en anglais **before**)

Exemples à suivre avec un fichier contenant des lignes du type :

```
; Inventaire: xxxxxx
; Modele: xxxxxx
; Proprietaire: xxxxxx
; Emplacement: xxxxxx
; Prise: xxxxxx
; Spanning-tree: xxxxxx
; Utilisateur: xxxxxx
```

On veut les modèles possédés par l'équipe ECP6 :

```
% grep -B 1 -i "proprietaire: ecp6" data
; Modele: PC NEC, Iiyama plat 21" AS4611UT, 1Go, Windows2000
; Proprietaire: ECP6
--
; Modele: PC NEC, NEC 19", 512Mo, Linux
; Proprietaire: ECP6
--
; Modele: PC Dell, Dell 19", 64Mo, Windows2000
; Proprietaire: ECP6
--
etc.
```

On veut les emplacements des matériels de l'équipe ECP6 :

```
% grep -A 1 -i "proprietaire: ecp6" data
; Proprietaire: ECP6
; Emplacement: Chevaleret, 1E3
--
; Proprietaire: ECP6
; Emplacement: Chevaleret, 1E16
--
; Proprietaire: ECP6
; Emplacement: Chevaleret, 1E10
--
etc.
```

On veut les emplacements et les modèles des matériels de l'équipe ECP6 :

```
% grep -A 1 -B 1 -i "proprietaire: ecp6" data
; Modele: PC NEC, Iiyama plat 21" AS4611UT, 1Go, Windows2000
; Proprietaire: ECP6
; Emplacement: Chevaleret, 1E3
--
; Modele: PC NEC, NEC 19", 512Mo, Linux
; Proprietaire: ECP6
; Emplacement: Chevaleret, 1E16
--
; Modele: PC Dell, Dell 17", Windows98
; Proprietaire: ECP6
; Emplacement: Chevaleret, 1E10
--
etc.
```

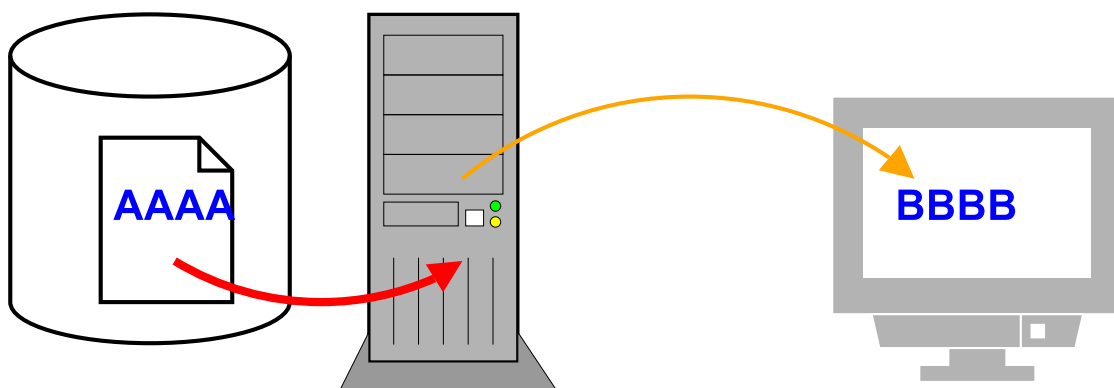
## Chapitre 10 • Expressions régulières et commandes UNIX associées

### §10.3 • Modification à la volée de contenu de fichiers : sed

(en anglais *stream editor*)

Syntaxe : `sed [options] fichiers`

La commande « `sed` » agit sur un **flux**.



```
sed -e 's/AAAA/BBBB/' exemple.txt
```



Qu'est qu'un flux ?

*Un flux est une quantité de texte envoyé à l'affichage par une commande.*

Exemples typiques de flux :

- le résultat d'une commande « `ls` »
- le résultat d'un « `cat exemple.txt` »

**sed modifie un flux, pas un contenu de fichier :  
après l'application de la commande, le fichier utilisé  
est inchangé.**

Quelques options intéressantes :

- option « `-e commande` » : commande à exécuter sur chaque ligne du flux
- option « `-f script` » : précision d'un fichier dans lequel prendre les commandes à appliquer sur chaque ligne de texte

Une commande prend l'une des formes suivantes :

```
[adresse1 [, adresse2]] fonction[argument]  
[/regexp1/ [, /regexp2/]] fonction[argument]
```

(avec les crochets « `[]` » désignant l'aspect facultatif de l'objet)

On construit une adresse de ligne grâce à :

- son numéro de ligne
- le caractère « `$` » pour désigner la dernière ligne

Les fonctions proposées dans sed :

- suppression de ligne : « d » (en anglais *delete*)
- affichage de ligne : « p » (en anglais *print*)  
La fonction « p » n'est intéressante que couplée à l'option « -n » de « sed » car l'option « -n » supprime l'affiche par défaut des lignes.
- substitution au sein des lignes :  
« s/regexp/remplacement/modifiers » (en anglais *substitute*)

Des compléments au fonctionnement de la commande sed seront donnés en TP.

### ◇ Exemple 1

Soit le fichier « exemple.txt » :

```
Ceci est la premiere ligne.  
Ceci est la deuxieme ligne.  
Ceci est la troisieme ligne.  
Ceci est la quatrieme ligne.  
Ceci est la cinquieme ligne.
```

On veut supprimer les 2 premieres lignes du fichier **lors de son affichage** :

```
% sed -e '1,2d' exemple.txt  
Ceci est la troisieme ligne.  
Ceci est la quatrieme ligne.  
Ceci est la cinquieme ligne.
```

### ◇ Exemple 2

Soit le fichier « exemple.txt » :

```
Ceci est la premiere ligne.  
Ceci est la deuxieme ligne.  
Ceci est la troisieme ligne.  
Ceci est la quatrieme ligne.  
Ceci est la cinquieme ligne.
```

On ne veut garder que les 2 premieres lignes du fichier **lors de son affichage** :

```
% sed -e '3,$d' exemple.txt  
Ceci est la premiere ligne.  
Ceci est la deuxieme ligne.
```

### ◇ Exemple 3

Soit le fichier « exemple.txt » :

```
moteur;ferrari;30  
moteur;porsche;epuise  
carrosserie;porsche;epuise  
moteur;ford;40  
moteur;skoda;epuise
```

On veut remplacer le mot « epuise » du fichier par 0 **lors de son affichage** :

```
% sed -e 's/epuise/0/' exemple.txt  
moteur;ferrari;30  
moteur;porsche;0  
carrosserie;porsche;0  
moteur;ford;40  
moteur;skoda;0
```

#### ◇ Exemple 4

Soit le fichier « exemple.txt » :

```
moteur;ferrari;30
moteur;porsche;epuise
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;epuise
```

On veut remplacer le mot « epuise » du fichier par 0 pour les lignes parlant de moteur **lors de son affichage** :

```
% sed -e '/moteur/s/epuise/0/' exemple.txt
moteur;ferrari;30
moteur;porsche;0
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;0
```

#### ◇ Exemple 5

Soit le fichier « exemple.txt » :

```
Les courses de chevaux se font a Vincennes.
Les chevaux ont 4 jambes. Les chevaux sont des equides.
```

On veut remplacer le mot « chevaux » du fichier par « chevaux » **lors de son affichage** :

```
% sed -e 's/chevals/chevaux/' exemple.txt
Les courses de chevaux se font a Vincennes.
Les chevaux ont 4 jambes. Les chevaux sont des equides.
```

⇒ Le remplacement n'a pas lieu sur tous les mots « chevaux » au sein d'une ligne !

### ◇ Exemple 5 (suite)

Soit le fichier « exemple.txt » :

```
Les courses de chevaux se font a Vincennes.  
Les chevaux ont 4 jambes. Les chevaux sont des equides.
```

On veut remplacer le mot « chevaux » du fichier par « chevaux » **lors de son affichage** :

```
% sed -e 's/chevals/chevaux/g' exemple.txt  
Les courses de chevaux se font a Vincennes.  
Les chevaux ont 4 jambes. Les chevaux sont des equides.
```

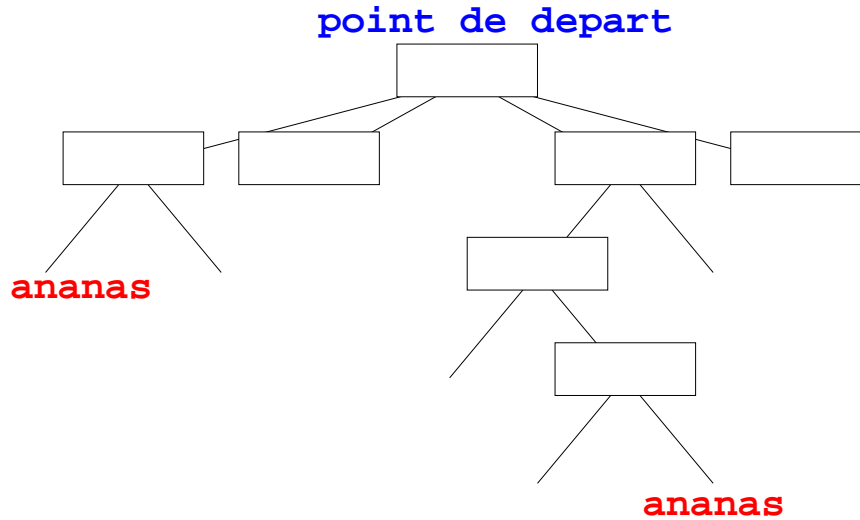
⇒ Le remplacement a lieu sur tous les mots « chevaux » au sein d'une ligne.

## Chapitre 11

# *Commande de recherche d'objets : find*

(en anglais *find*)

On recherche à **partir des répertoires indiqués** les objets répondant aux critères exprimés par des expressions.



Syntaxe : `find répertoires expressions`

Les expressions indiquent :

- des conditions
- des actions à effectuer

### ◇ Options pour rechercher sur un nom :

Syntaxe : « -name nom »

Le nom sera spécifié avec des métacaractères du shell (voir page 506)

Autre syntaxe possible (seulement LINUX) : « -iname nom » (lettres minuscules ou majuscules indifférenciées)

Le nom sera spécifié avec des métacaractères du shell (voir page 506)

Exemple : rechercher des objets d'extension « .c » :

```
% find . -name \*.c -print
% find . -name '*.c' -print
```

### ◇ Options pour rechercher sur des droits d'accès :

Plusieurs syntaxes possibles :

- Syntaxe : « -perm permissions »  
Les permissions doivent être strictement celles indiquées.
- Syntaxe : « -perm -permissions »  
Les bits indiqués doivent tous exister parmi les permissions des objets recherchés.
- Syntaxe (seulement LINUX) : « -perm +permissions »  
Au moins l'un des bits indiqués doit exister parmi les permissions des objets recherchés.

Exemple : rechercher des objets de permission 755 :

```
% find . -perm 755 -print
```

### ◇ Options pour rechercher sur le propriétaire d'objets :

Plusieurs syntaxes possibles :

- Syntaxe : « `-user login` »  
Le propriétaire de l'objet doit être le login indiqué
- Syntaxe : « `-nouser` »  
Le fichier doit appartenir à un utilisateur non défini sur le système, c'est-à-dire l'UID n'a pas de login associé dans « `/etc/passwd` »

Exemple : rechercher les objets de l'utilisateur « `besancon` » :

```
% find . -user besancon -print
```

### ◇ Options pour rechercher sur types d'objets :

Syntaxe : « `-type X` »

avec X une lettre indiquant la nature de l'objet :

- « `d` » pour directory
- « `f` » pour file
- « `l` » pour lien symbolique
- « `c` » pour fichier character
- « `b` » pour fichier bloc
- « `s` » pour socket
- « `D` » pour door Solaris

Exemple : rechercher des répertoires :

```
% find . -type d -print
```



### ◇ Options pour rechercher sur des tailles d'objets :

Plusieurs syntaxes possibles :

- Syntaxe : « -size nombre »
- Syntaxe : « -size -nombre »
- Syntaxe : « -size +nombre »

Si le nombre indique « c », alors il exprime des octets sinon ce sont des blocs de 512 octets.

Exemple : rechercher des objets de plus de 1000000 caractères :

```
% find . -size +1000000c -print
```

### ◇ Options pour rechercher sur des dates :

Plusieurs syntaxes :

- Syntaxe : « -newer fichier »
- Syntaxe : « -atime nombre »
- Syntaxe : « -mtime nombre »
- Syntaxe : « -ctime nombre »

Sur SOLARIS, l'unité du nombre est en jours.

Sur LINUX, plusieurs unités sont possibles : « d » pour jour, « h » pour heure, « m » pour minute, « s » pour seconde

Exemple : rechercher des objets moins vieux de 3 jours

```
% find . -mtime -3 -print
```

### ◇ Composition d'options :

Plusieurs syntaxes possibles :

- Syntaxe : OU logique entre expressions :  
« condition1 -o condition2 »  
(attention : OU entre **expressions** ci-dessus)
- Syntaxe : ET logique entre expressions :  
« condition1 -a condition2 »  
(attention : ET entre **expressions** ci-dessus); le -a est facultatif
- Syntaxe : groupement d'expressions :  
« ( expression1 -[ao] expression2 ) »  
**ATTENTION : nécessiter de protéger du shell les parenthèses !**  
⇒ écriture en pratique :  
« \ ( expression1 -[ao] expression2 \ ) »

Exemples :

```
% find . -name fruits -type d -print
% find . -name ananas -o -name cerise -print
```

### ◇ Options pour affichage du nom des objets trouvés :

Plusieurs options possibles :

- Syntaxe : « -print »
- Syntaxe : « -ls »

Exemples :

```
% find . -type d -print
% find . -type f -ls
```

### ◇ Option d'exécution d'une commande :

Syntaxe : « `-exec commande {} ;` »

**ATTENTION : nécessiter de protéger du shell le point-virgule !**

⇒ écriture en pratique : « `-exec commande {} \;` »

Exemple : rechercher tous les objets s'appelant `a.out` ou s'appelant avec une extension `.o`, non utilisés depuis plus de 7 jours et on appliquera la commande d'effacement aux objets trouvés :

```
% find . \( -name 'a.out' -o -name '*.o' \) -atime +7 -exec rm {} \;
```

### ◇ Retour sur l'arborescence UNIX

Le répertoire courant est noté « `.` ».

Lancer une recherche à partir de l'endroit où l'on est :

```
% find . -name ananas.txt -print
```

## Ne pas confondre la commande find et la commande ls

Rechercher un fichier nommé « ananas.txt » dans une arborescence :

- OUI : « find . -name ananas.txt -print »
- NON : « ls -Rl | grep ananas.txt »

Pourquoi ?

→ La commande ls pourrait renvoyer des données parasites.

Dans le répertoire courant, chercher les fichier commençant par « banane » :

- NON : « find . -name 'banane\*' -print »
- OUI : « ls banane\* »

Pourquoi ?

→ La commande find va faire une recherche récursive et dépasser le niveau du répertoire courant en descendant plus bas.

## La directive -exec n'est pas très pratique.

Exemple où -exec n'est pas adapté :

Rechercher les objets commençant par banane et les renommer en 2003-banane...

→ La directive -exec symbolise le fichier à afficher par « {} » mais ne permet pas de construire des commandes non basiques :

```
% ls banane*
banane1  banane2

% find . -name 'banane*' -exec echo {} {} \;
./banane1 ./banane1
./banane2 ./banane2

% find . -name 'banane*' -exec echo {} 2003-{} \;
./banane1 2003-{}
./banane2 2003-{}

```

## Les objets trouvés partent du point de recherche et le mentionnent.

```
% ls banane*  
banane1  banane2  
  
% find . -name 'banane*' -print  
./banane1  
./banane2
```

Attention si vous devez donc retraiter les noms des objets trouvés (ici parasitage du « ./ »).

## Chapitre 12

# *Commandes UNIX réseau de base*

## Chapitre 12 • Commandes UNIX réseau de base

§12.1 • Nom de machine : hostname

Syntaxe : hostname

```
% hostname
serveur.formation.jussieu.fr
```

(la commande sert aussi à baptiser une machine. Syntaxe :  
hostname nom-de-machine)

## Chapitre 12 • Commandes UNIX réseau de base

§12.2 • Nom de système, de machine : uname

Syntaxe : uname [options]

```
% uname -n
serveur.formation.jussieu.fr
```

```
% uname -a
Linux serveur.formation.jussieu.fr 2.4.20-28.7smp #1 SMP Thu Dec
18 11:18:31 EST 2003 i686 unknown
```

```
% uname -s
Linux
```

```
% uname -m
i686
```

```
% uname -r
2.4.20-28.7smp
```

## Autre exemple :

```
% uname -n
solaris.example.com

% uname -a
SunOS solaris.example.com 5.10 Generic_118822-20 sun4u sparcsun4w,Sun-Blade-100

% uname -s
SunOS

% uname -m
sun4u

% uname -r
5.10
```

## Chapitre 12 • Commandes UNIX réseau de base

### §12.3 • Test de connectivité : `ping`

La commande « `ping` » permet de tester si une machine est joignable.

Principe : envoi d'un paquet réseau spécial à une machine (effet « ping ») qui répond par un autre paquet réseau spécial (effet « pong »)

Syntaxe : `ping [options] nom-de-machine`

```
% ping host1.example.com
sendto: Network is unreachable
% ping host2.example.com
host2.example.com is alive
```

### Humour :

```
% ping elvis
elvis is alive
```

## Autre type d'affichage :

```
% ping www.lemonde.fr
PING a245.g.akamai.net (81.52.207.14) from 134.157.46.137 :
56(84) bytes of data
.
64 bytes from 81.52.207.14: icmp_seq=0 ttl=53 time=11.834 msec
64 bytes from 81.52.207.14: icmp_seq=1 ttl=53 time=11.499 msec
64 bytes from 81.52.207.14: icmp_seq=2 ttl=53 time=11.103 msec
64 bytes from 81.52.207.14: icmp_seq=3 ttl=53 time=11.651 msec
64 bytes from 81.52.207.14: icmp_seq=4 ttl=53 time=10.817 msec
64 bytes from 81.52.207.14: icmp_seq=5 ttl=53 time=11.354 msec
^C
--- a245.g.akamai.net ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max/mdev = 10.817/11.376/11.834/0.349 ms
```

## Chapitre 12 • Commandes UNIX réseau de base

### §12.4 • (Windows : : Test de connectivité : ping.exe)

## Commande « ping.exe »



La commande `traceroute` permet de tester si une machine est joignable. Elle renvoie les intermédiaires réseau qui route notre acheminement vers la machine distante.

Syntaxe : `traceroute machine`

```
% traceroute ftp.lip6.fr
traceroute to nephtys.lip6.fr (195.83.118.1), 30 hops max, 40 byte
packets
 1 yacht (129.199.96.254)  0 ms  0 ms  0 ms
 2 renater (129.199.1.10)  2 ms  1 ms  1 ms
 3 195.221.127.61 (195.221.127.61)  3 ms  1 ms  1 ms
 4 195.221.126.1 (195.221.126.1)  2 ms  1 ms  1 ms
 5 195.221.126.78 (195.221.126.78)  2 ms  1 ms  1 ms
 6 jussieu.rap.prd.fr (195.221.126.33)  2 ms  2 ms  2 ms
 7 nephtys.lip6.fr (195.83.118.1)  2 ms  2 ms  2 ms
```

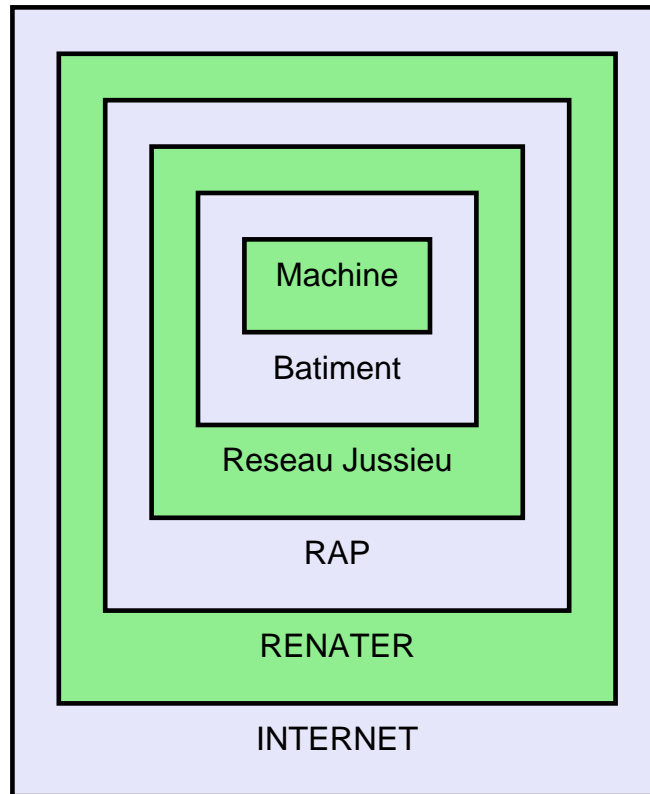
Le nombre d'intermédiaires n'est pas proportionnel à l'éloignement géographique de la machine destination.

En cas de soucis :

```
% traceroute serveur.formation.jussieu.fr
traceroute to serveur.formation.jussieu.fr (134.157.46.129), 30 hops
max, 40 byte packets
 1 yacht (129.199.96.254)  0 ms  0 ms  0 ms
 2 renater (129.199.1.10)  2 ms  1 ms  1 ms
 3 195.221.127.61 (195.221.127.61)  2 ms  2 ms  1 ms
 4 195.221.126.1 (195.221.126.1)  2 ms  1 ms  1 ms
 5 195.221.126.78 (195.221.126.78)  2 ms  1 ms  2 ms
 6 jussieu.rap.prd.fr (195.221.126.33)  3 ms  2 ms  2 ms
 7 134.157.254.123 (134.157.254.123)  3 ms  2 ms  2 ms
 8 * * *   <- symptôme de problème
^C
```

**Attention : ne pas oublier les filtrages réseau des firewalls qui peuvent se traduire au niveau de ping comme un problème (étoiles).**

Utile de connaître les FAI (*Fournisseur d'Accès à Internet*) que vous utilisez :



## Chapitre 12 • Commandes UNIX réseau de base

### §12.6 • (Windows : : Test de connectivité : tracert.exe

Commande « tracert.exe »

```

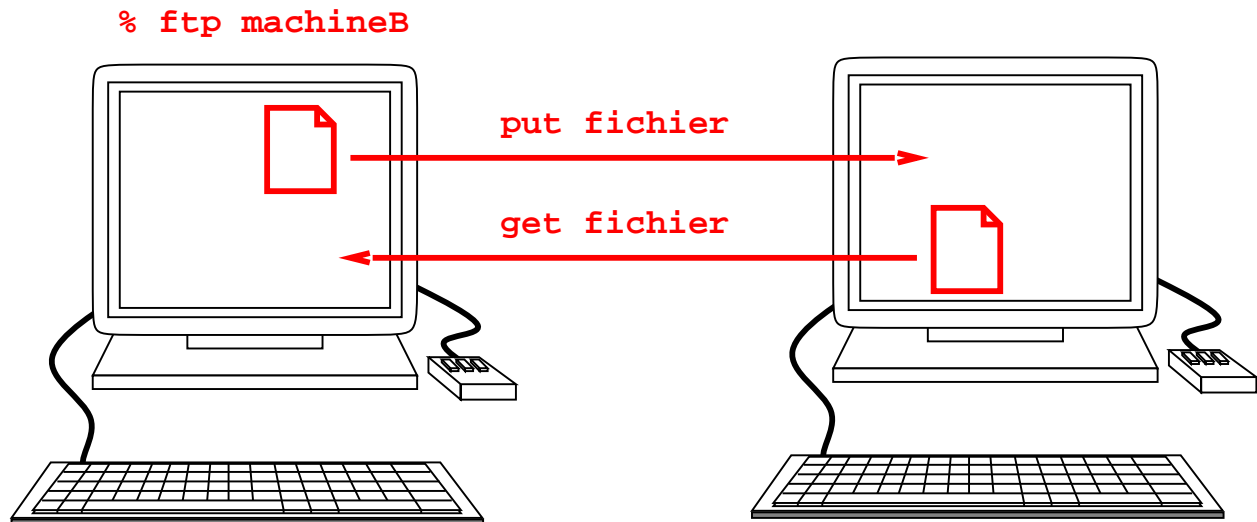
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\besancon>tracert serveur.formation.jussieu.fr
Tracing route to serveur.formation.jussieu.fr [134.157.46.129]
over a maximum of 30 hops:
  0  1 ms    1 ms    1 ms    192.168.1.254
  1  40 ms   40 ms   41 ms   192.168.1.1
  2  42 ms   *      135 ms  th2-6k-2-a5.routers.proxad.net [213.228.7.254]
  3  *      *      *      Request timed out.
  4  39 ms   40 ms   39 ms   p19-6k-2-po4.intf.routers.proxad.net [212.27.50.14]
  5  41 ms   42 ms   41 ms   aub-6k-1-v806.routers.proxad.net [212.27.50.162]
  6  43 ms   40 ms   41 ms   renater.sfinx.tm.fr [194.68.129.102]
  7  43 ms   40 ms   39 ms   nri-c-gi3-0-0-12.cssi.renater.fr [193.51.179.154]
  8  42 ms   40 ms   39 ms   jussieu-pos4-0.cssi.renater.fr [193.51.180.157]
  9  41 ms   45 ms   40 ms   gw-rap.rap.prd.fr [193.50.20.73]
 10  41 ms   43 ms   43 ms   jussieu-rap.rap.prd.fr [195.221.127.182]
 11  42 ms   49 ms   42 ms   r-intercon3.reseau.jussieu.fr [134.157.254.26]
 12  44 ms   41 ms   43 ms   serveur.formation.jussieu.fr [134.157.46.129]

Trace complete.
C:\Documents and Settings\besancon>_

```

(en anglais *file transfer protocol*)

Syntaxe : `ftp machine`



La commande `ftp` vous place dans une espèce de shell dans lequel vous disposez des commandes suivantes (ce sont les plus importantes à votre niveau) :

- commande « `binary` » : à utiliser si le fichier à transmettre contient des caractères non texte
- commande « `dir` » : pour lister les fichiers sur la machine distante
- commande « `!ls` » : pour lister les fichiers sur la machine locale
- commande « `cd directory` » : pour changer de répertoire sur la machine distante
- commande « `lcd directory` » : pour changer de répertoire sur la machine locale
- commande « `get fichier` » : pour récupérer sur la machine distante un fichier
- commande « `put fichier` » : pour déposer sur la machine distante un fichier
- commande « `quit` » : pour se déconnecter

```

% ftp unix.example.com
Connected to unix.example.com.
220 unix.example.com FTP server (SunOS 4.1) ready.
Name (unix:besancon):
331 Password required for besancon.
Password:
230 User besancon logged in.
ftp> dir
200 PORT command successful.
150 ASCII data connection for /bin/ls (192.168.0.1,3945) (0 bytes).
total 307
-rw-r--r--  1 besancon software      69 Mar 20  1995 .Xdefaults
...
ftp> get fichier [nouveau nom]
...
ftp> put fichier [nouveau nom]
200 PORT command successful.
150 ASCII data connection for fichier (192.168.0.1,3947).
226 ASCII Transfer complete.
local: fichier remote: fichier
802 bytes sent in 0.0042 seconds (1.9e+02 Kbytes/s)
ftp> quit
221 Goodbye.

```

## Chapitre 12 • Commandes UNIX réseau de base

### §12.8 • (Windows : : Transfert de fichiers : ftp.exe)

Multiples utilitaires pour faire du FTP sous WINDOWS.

Utilitaire « ftp.exe » fourni par WINDOWS :

```

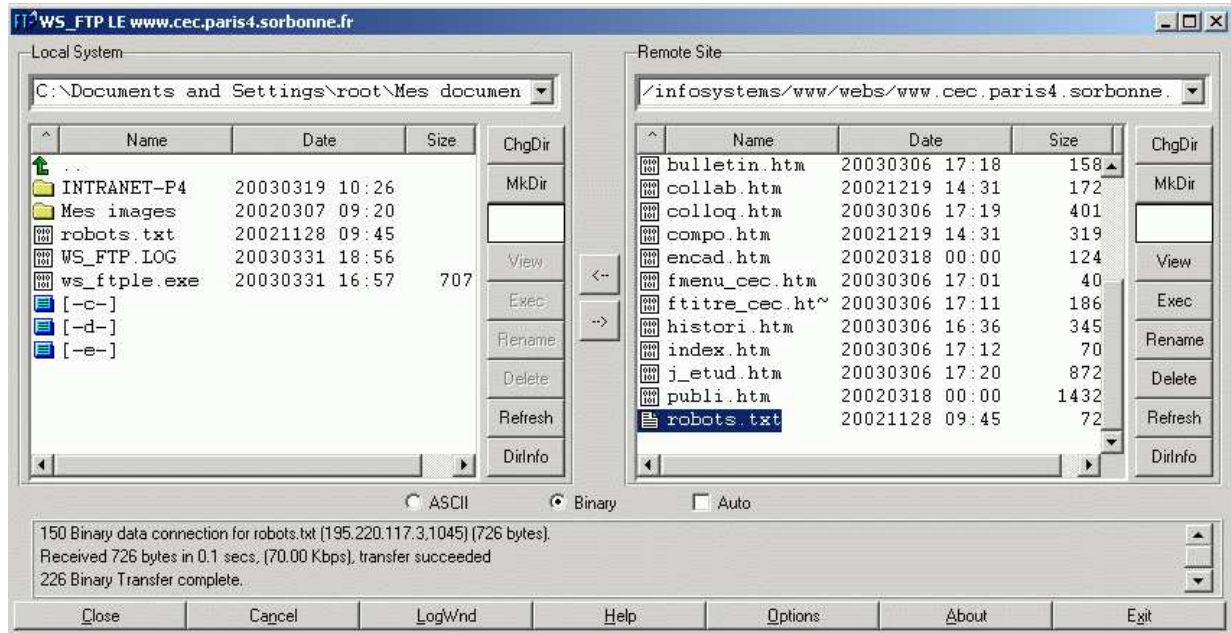
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\root>ftp [redacted]
Connecté à [redacted].org.
220 [redacted].org FTP server ready.
Utilisateur ([redacted].org:(none)) : besancon
331 Password required for besancon.
Mot de passe :
230 User besancon logged in.
ftp> quit
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 183 bytes in 0 transfers.
221-Thank you for using the FTP service on [redacted].org.
221 Goodbye.

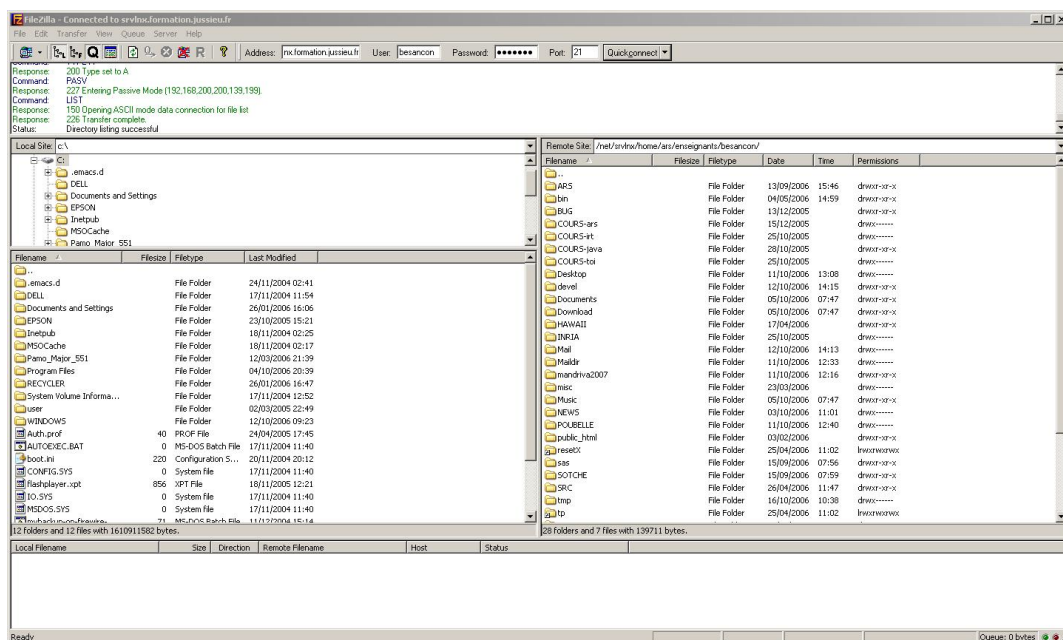
C:\Documents and Settings\root>

```

Exemple d'utilitaire graphique archi classique : WS\_FTP LE (site <http://www.ipswitch.com>)



Exemple d'utilitaire graphique : filezilla (site <http://filezilla.sourceforge.net>)



## Chapitre 12 • Commandes UNIX réseau de base

### §12.9 • Lancement de commande à distance : protocole SSH, ssh

(en anglais *secure shell*)

Le protocole SSH chiffre la communication avec la machine distante.  
 ⇒ protection contre les piratages du type mouchard réseau qui récupère les mots de passe

Plusieurs syntaxes de la commande « ssh » :

- ssh [-l utilisateur] nom-de-machine commande
- ssh utilisateur@nom-de-machine [commande]

Si l'on ne précise pas de commande, on lancera un shell en interactif.

Site pour récupérer un logiciel UNIX connaissant le protocole SSH :

<http://www.openssh.org>

#### ◇ Exemple 1 :

```
% ssh server.example.com
besancon@server.example.com's password: XXXXXXXXX
Last login: Sun Oct 12 15:20:16 2003 from ppp-3
Sun Microsystems Inc. SunOS 5.5 Generic November 1995
No mail.
server%
```

#### ◇ Exemple 2 :

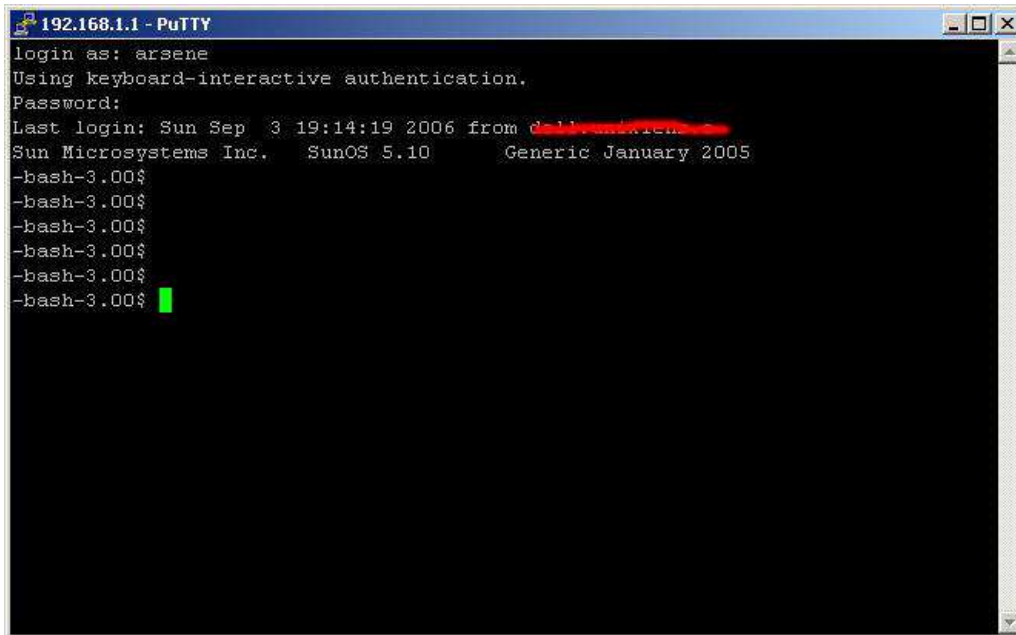
```
% ssh server.example.com date
Password: XXXXXXXXX
Fri Jun 9 21:27:34 MEST 2006
```

## Chapitre 12 • Commandes UNIX réseau de base

§12.10 • (Windows : : Connexion à distance interactive SSH : `putty.exe`)

L'utilitaire PUTTY fournit les fonctionnalités du protocole SSH pour Windows.

Site : <http://www.chiark.greenend.org.uk/~sgtatham/putty/>



## Chapitre 12 • Commandes UNIX réseau de base

§12.11 • Recopie de fichiers à distance : protocole SSH, `scp`

(en anglais *secure copy*)

Plusieurs syntaxes :

- machine distante à machine locale :  
`scp user@machine:chemin-fichier1 chemin-fichier2`
- machine locale à machine distante :  
`scp chemin-fichier1 user@machine:chemin-fichier2`

```
% scp ananas.txt besancon@server.example.com:/tmp/cerise.txt
besancon@server.example.com's password: XXXXXXXX
ananas.txt          100% |*****| 15          00:00

% ssh besancon@server.example.com ls -l /tmp/cerise.txt
besancon@server.example.com's password: XXXXXXXX
-rw-r--r--  1 besancon ars          15 Oct 12 15:24 /tmp/cerise.txt
```

Les logiciels implémentant SCP sont livrés avec ceux implémentant SSH

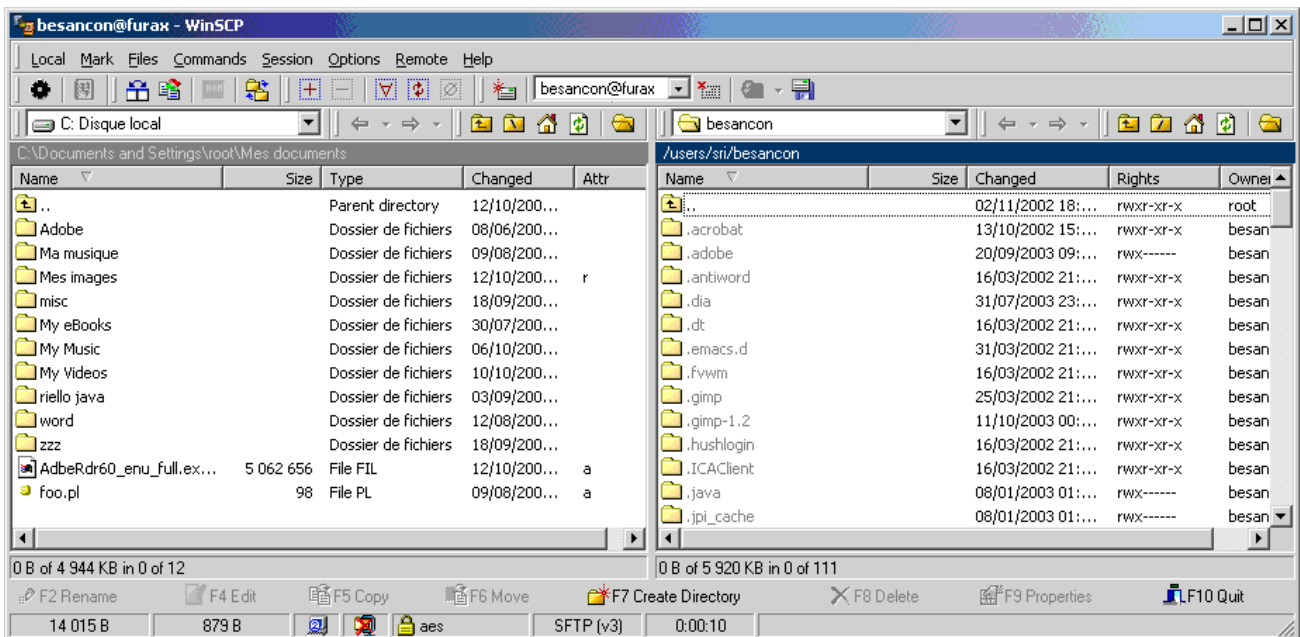
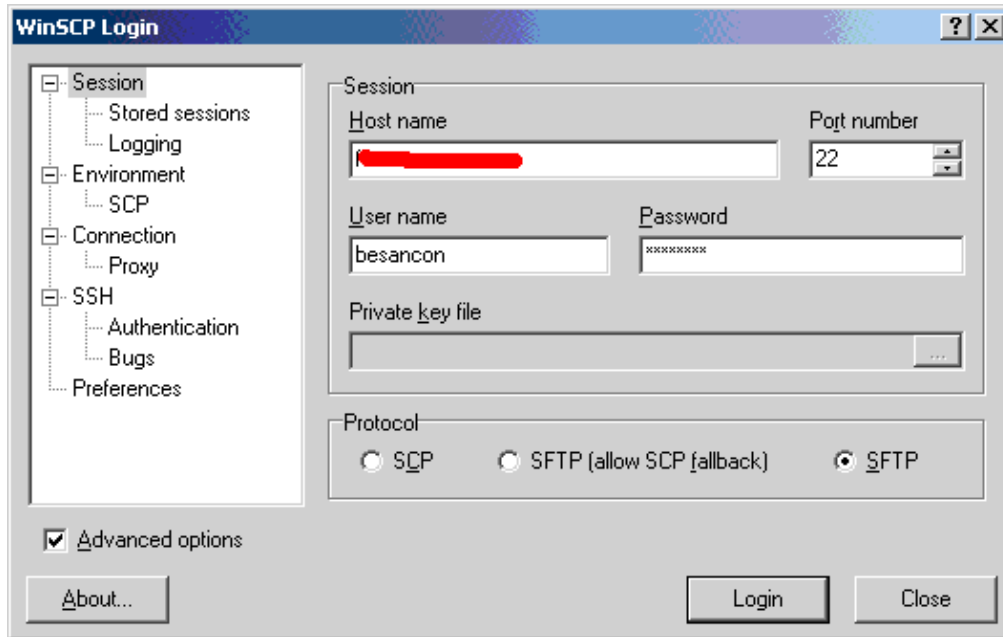
⇒ sur UNIX se reporter à <http://www.openssh.org>

## Chapitre 12 • Commandes UNIX réseau de base

§12.12 • (Windows : : Recopie de fichiers à distance : winscp.exe)

L'utilitaire WINSCP fournit les fonctionnalités du protocole SSH pour Windows.

Site : <http://winscp.sourceforge.net/eng/>



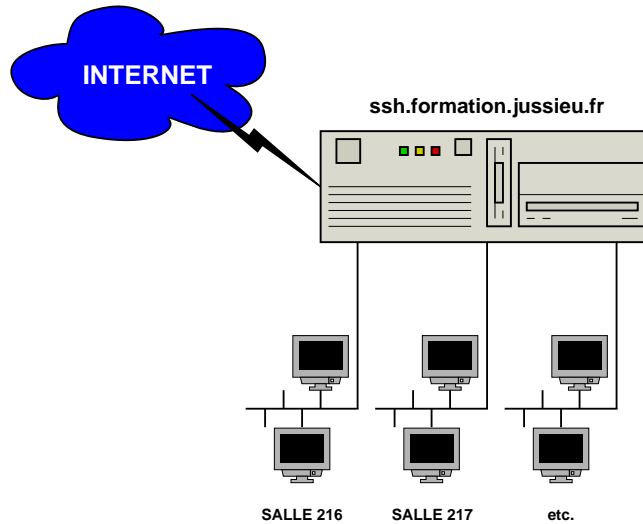


## Chapitre 12 • Commandes UNIX réseau de base

## §12.13 • Point d'entrée réseau de la Formation Permanente

Machine `ssh.formation.jussieu.fr` accessible via SSH/SCP  
**exclusivement**

**Point de passage obligatoire depuis internet vers les machines des salles**



## Chapitre 12 • Commandes UNIX réseau de base

## §12.14 • Liste des utilisateurs connectés : users

Syntaxe : `users`

```
% users
```

```
aidan besancon fouquet kahn vogel
```

## Chapitre 12 • Commandes UNIX réseau de base

## §12.15 • Liste des utilisateurs connectés : who, who am i, whoami

Syntaxe : who [options]

```
% who
kahn          tty0      Jul 11 22:48 (1.2.3.4)
vogel        tty1      Jul 10 20:33 (1.2.3.5)
aidan        tty2      Jul 12 00:35 (1.2.3.6)
besancon     tty3      Jul 12 00:46 (1.2.3.7)
fouquet      tty7      Jul 10 14:30 (1.2.3.8)
```

Forme déclinée sur la commande précédente :

```
% who am i
besancon     /dev/pts/5 Jul 14 23:47
```

Autre commande ressemblante :

```
% whoami
besancon
```

## Chapitre 12 • Commandes UNIX réseau de base

## §12.16 • Liste des utilisateurs connectés : w

Syntaxe : w [options]

```
% w
USER          TTY      FROM          LOGIN@  IDLE WHAT
kahn          p0       x.example.com Sun10PM  1:00 -bash (bash)
vogel        p1       y.example.com Sat08PM  2:00 mutt -f mai.04
aidan        p2       z.example.com 12:35AM   9 pine
besancon     p3       t.example.com 12:46AM   - w
fouquet      p7       q.example.com Sat02PM  1:06 -bash (bash)
```

## Chapitre 12 • Commandes UNIX réseau de base

## §12.17 • Liste des utilisateurs connectés : finger

Syntaxe : `finger [options]`

```
% finger
```

Login	Name	TTY	Idle	Login	Time	Office	Phone
aidan	Jonathan Aidan	*p2	10	Mon	00:35	UMR-THE	
besancon	Thierry Besancon	p3		Mon	00:46	UMR-ADM	7D10
fouquet	Mireille Fouquet	p7	1:07	Sat	14:30	UMR-PER	
kahn	Bruno Kahn	p0	1:00	Sun	22:48	UMR-CNR	
vogel	Pierre Vogel	p1	2:00	Sat	20:33	UMR-PER	

## Chapitre 12 • Commandes UNIX réseau de base

## §12.18 • Navigation Web : URL

URL = Universal Resource Locator

*URL : Dénomination unique à caractère universel qui permet de localiser une ressource, un document, sur l'Internet, et qui indique la méthode pour y accéder, le nom du serveur et le chemin à l'intérieur du serveur.*

Typiquement :

```
protocole://serveur/chemin
```

Les types les plus répandus :

- URL de page web : « `http://www.lemonde.fr/` »  
ouvrable par Mozilla, Firefox, Opera
- URL de page ftp : « `ftp://ftp.jussieu.fr/` »  
ouvrable par Mozilla, Firefox, Opera
- URL d'adresse email :  
« `mailto:Thierry.Besancon@formation.jussieu.fr` »  
ouvrable par Mozilla, Thunderbird

## Chapitre 12 • Commandes UNIX réseau de base

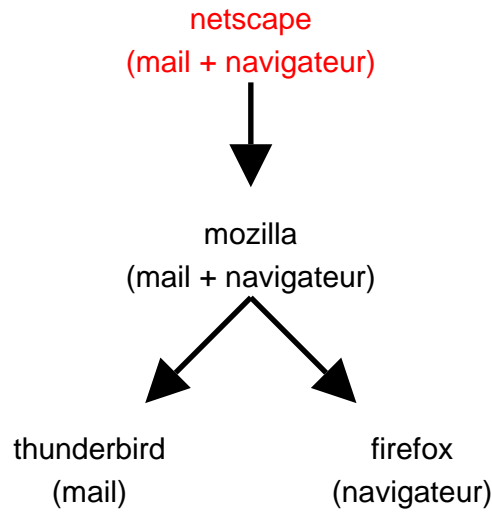
### §12.19 • Navigateur Web : lynx

`http://www.lynx.org/`

Syntaxe : `lynx URL`



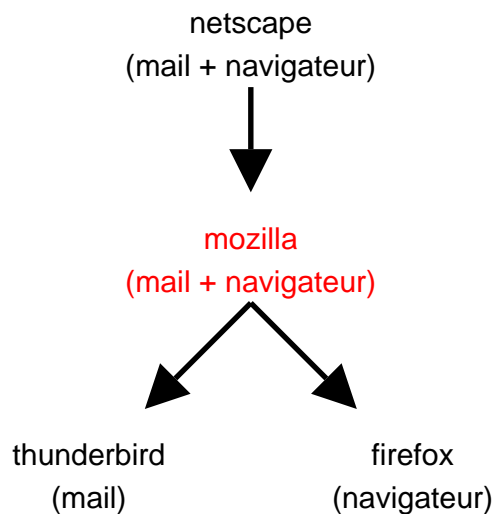
<http://www.netscape.com/>



Obsolète

<http://www.mozilla.org/>

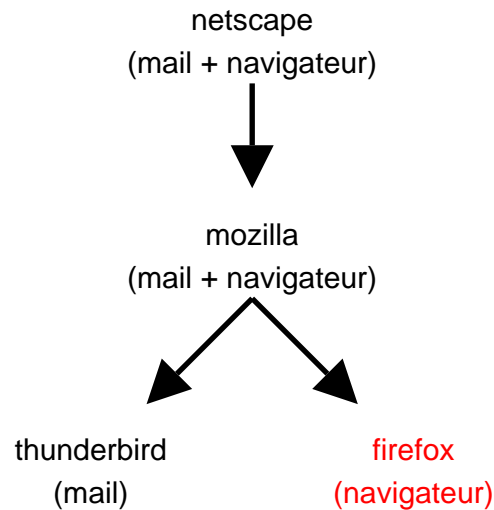
Syntaxe : mozilla URL



Obsolète

`http://www.firefox.org/`

Syntaxe : `firefox URL`



Parfois des problèmes liés à une mauvaise sortie antérieure de « firefox » :



Solution :

```
% rm -f $HOME/.mozilla/firefox/*/lock
% rm -f $HOME/.mozilla/firefox/*/.parentlock
```

Lorsque « firefox » quitte, les fichiers précédents doivent disparaître :

```

Shell - Konsole <2>
Session Edit View Bookmarks Settings Help

[besancon@216-pc01 ~/.mozilla/firefox/e0m0lyzj.default]$ ls -la
total 1532
drwx----- 6 besancon ars      4096 Oct 11 14:10 .
drwx----- 3 besancon ars        84 Oct 11 14:10 ..
-rw-r--r--  1 besancon ars         0 Oct 11 14:10 .parentlock
drwxr-xr-x  2 besancon ars       132 Oct 10 14:21 Cache
-rw-r--r--  1 besancon ars    1075519 Oct 10 14:14 XUL.mfasl
drwx----- 2 besancon ars        38 Oct 10 14:14 bookmarkbackups
-rw-r--r--  1 besancon ars     9804 Oct 10 14:14 bookmarks.html
-rw-----  1 besancon ars    65536 Oct 10 14:14 cert8.db
drwxr-xr-x  2 besancon ars        65 Oct 10 14:14 chrome
-rw-r--r--  1 besancon ars   124461 Oct 10 15:15 compreg.dat
-rw-----  1 besancon ars        511 Oct 10 14:14 cookies.txt
drwxr-xr-x  2 besancon ars         6 Oct 10 14:14 extensions
-rw-r--r--  1 besancon ars        854 Oct 10 14:14 extensions.cache
-rw-r--r--  1 besancon ars        835 Oct 10 14:14 extensions.ini
-rw-r--r--  1 besancon ars       8423 Oct 10 14:14 extensions.rdf
-rw-r--r--  1 besancon ars       2360 Oct 10 14:22 history.dat
-rw-----  1 besancon ars    131072 Oct 10 14:14 key3.db
-rw-r--r--  1 besancon ars        153 Oct 10 14:14 localstore.rdf
lrwxrwxrwx  1 besancon ars         19 Oct 10 14:14 lock -> 192.168.216.1:+5222
-rw-r--r--  1 besancon ars        287 Oct 10 14:14 mimeTypees.rdf
-rw-r--r--  1 besancon ars        483 Oct 10 14:14 prefs.js
-rw-r--r--  1 besancon ars        752 Oct 10 14:14 search.rdf
-rw-----  1 besancon ars    131072 Oct 10 14:14 secmod.db
  
```

©T.Besançon (version 11.0) Administration Unix ARS 2008-2009 Tôme 1 415 / 666

## Chapitre 12 • Commandes UNIX réseau de base

§12.23 • Navigateur Web : opera

<http://www.opera.com/>



Utilitaire de récupération de page web via ligne de commande.

Intérêt : permet d'automatiser des opérations de récupération de fichiers mis à disposition sur des serveurs web.

Syntaxe : `wget URL`

Site <http://www.gnu.org/software/wget>

Exemple : récupération d'un fichier PDF sur un site web :

```
% wget http://www.example.com/document.pdf
--01:46:59-- http://www.example.com/document.pdf
           => `document.pdf'
Resolving www.example.com... 192.168.0.1
Connecting to www.example.com[192.168.0.1]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4,241 [application/pdf]

100%[=====>] 4,241      --.--K/s

01:46:59 (28.28 MB/s) - `document.pdf' saved [4241/4241]
```



Utilitaire de récupération de page web via ligne de commande.

Intérêt : permet d'automatiser des opérations de récupération de fichiers mis à disposition sur des serveurs web.

Plus puissant que « wget » (supporte les connexions HTTPS en particulier, les formulaires, etc.)

Syntaxe : `curl URL`

Site <http://curl.haxx.se>

Exemple : récupération d'un fichier de backup sur un site sécurisé avec présentation d'un certificat d'authentification :

```
% curl \
  --show-error \
  --cacert ${HOME}/certificats/ca.crt \
  --key-type PEM \
  --key ${HOME}/certificats/thierry.besancon@example.com.key \
  --cert-type PEM \
  --cert ${HOME}/certificats/thierry.besancon@example.com.crt \
  --url https://www.example.com/backup/20060804.sql.gz \
  --output backup-20060804.sql.gz
% Total      % Received % Xferd  Average Speed   Time    Time       Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100  296  100  296    0    0    361      0  --:--:--  --:--:--  --:--:-- 12869
  2 backup-20060804.sql.gz
```

## Chapitre 12 • Commandes UNIX réseau de base

### §12.26 • Courrier électronique à la Formation Permanente

Votre adresse de courrier électronique est du type :

```
login@formation.jussieu.fr
```

Même mot de passe que le login UNIX !

Informations pour utiliser la messagerie de la Formation Permanente :

- protocole IMAP + sécurisation par SSL
- serveur depuis lequel lire le courrier :  
« mailhost.formation.jussieu.fr »  
ATTENTION : valable depuis tout internet
- serveur vers lequel envoyer le courrier :  
« mailhost.formation.jussieu.fr »  
ATTENTION : uniquement valable depuis les salles de TP de la Formation Permanente  
Utiliser le serveur SMTP de votre FAI.
- nom de connexion : login UNIX
- mot de passe : celui du login UNIX

Le fichier `$HOME/.forward` sert au renvoi du courrier électronique vers une autre adresse. On y précise les adresses vers lesquelles renvoyer.

◇ Exemple 1 :

Pour rediriger par exemple vers `besancon@example.com` :

```
besancon@example.com
```

◇ Exemple 2 :

Pour garder une copie locale (par exemple je suis `besancon@formation.jussieu.fr`) et quand même renvoyer vers une autre adresse (par exemple `besancon@example.com`) :

```
\besancon
besancon@example.com
```

Commandes de base historiques : `mail` ou `mailx` ou `Mail`

```
% Mail Thierry.Besancon@formation.jussieu.fr
Subject: test
Cc:
test
.
EOT
```

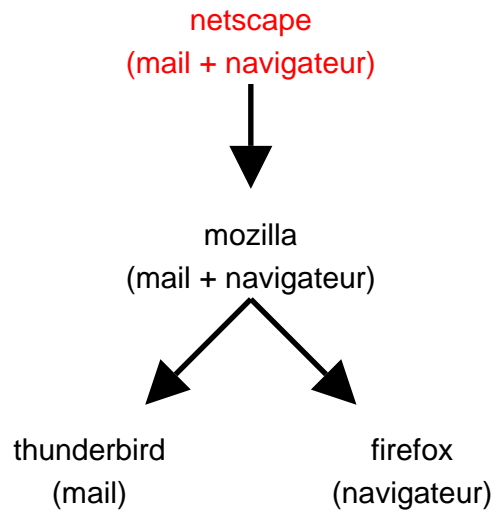
Utilisation pratique : envoi d'un fichier `texte` à quelqu'un :

```
mail -s "sujet du mail" adresses < contenu-mail.txt
```

(voir page 489 pour la redirection)

## Chapitre 12 • Commandes UNIX réseau de base

§12.29 • Courrier électronique : netscape

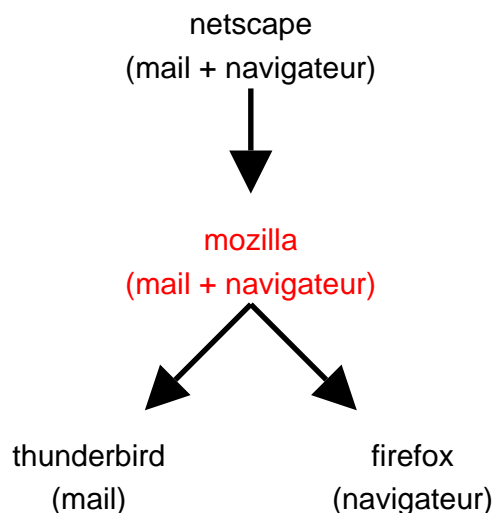
<http://www.netscape.com/>

Affichage graphique de vos mails. Consommateur de ressources.

Obsolète

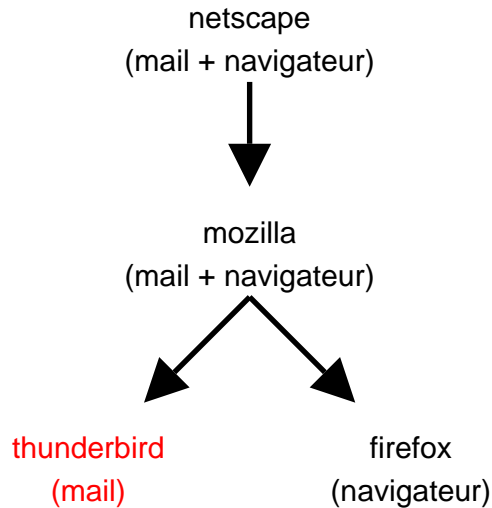
## Chapitre 12 • Commandes UNIX réseau de base

§12.30 • Courrier électronique : mozilla

<http://www.mozilla.org/>

Affichage graphique de vos mails. Consommateur de ressources.

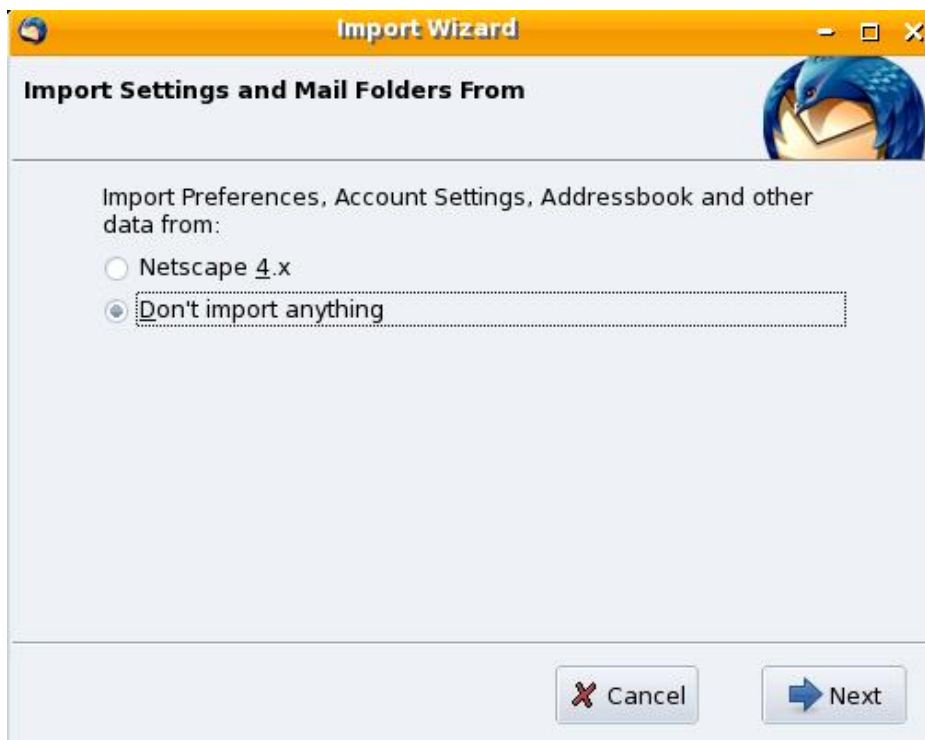
Obsolète



Affichage graphique de vos mails. Consommateur de ressources.

Préférer « thunderbird » à « mozilla » à « netscape ».

### ◇ Configuration de thunderbird



**Account Wizard** [X] [Close]

### Identity

Each account has an identity, which is the information that identifies you to others when they receive your messages.

Enter the name you would like to appear in the "From" field of your outgoing messages (for example, "John Smith").

Your Name:

Enter your email address. This is the address others will use to send email to you (for example, "user@example.net").

Email Address:

**Account Wizard** [X] [Close]

### Server Information

Select the type of incoming server you are using.

POP  IMAP

Enter the name of your incoming server (for example, "mail.example.net").

Incoming Server:

Enter the name of your outgoing server (SMTP) (for example, "smtp.example.net").

Outgoing Server:

**Account Wizard** □ ×

### User Names

Enter the incoming user name given to you by your email provider (for example, "jsmith").

Incoming User Name:

Enter the outgoing user name given to you by your email provider (this is typically the same as your incoming user name).

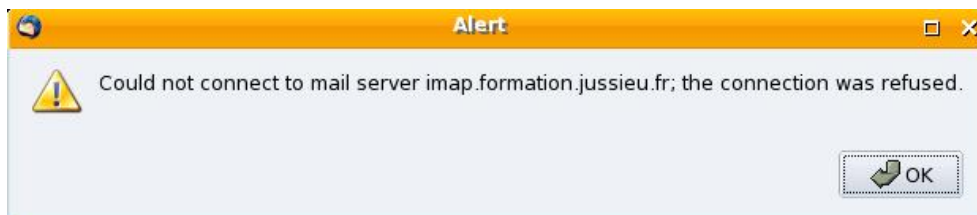
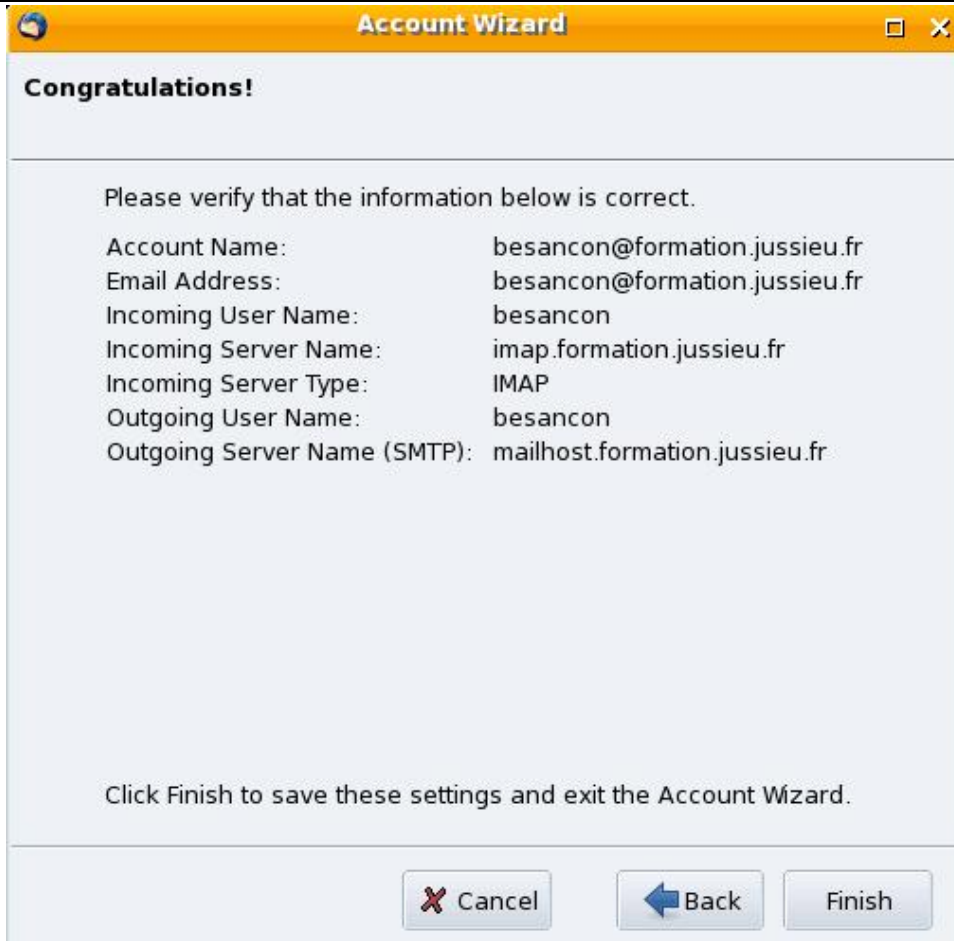
Outgoing User Name:

**Account Wizard** □ ×

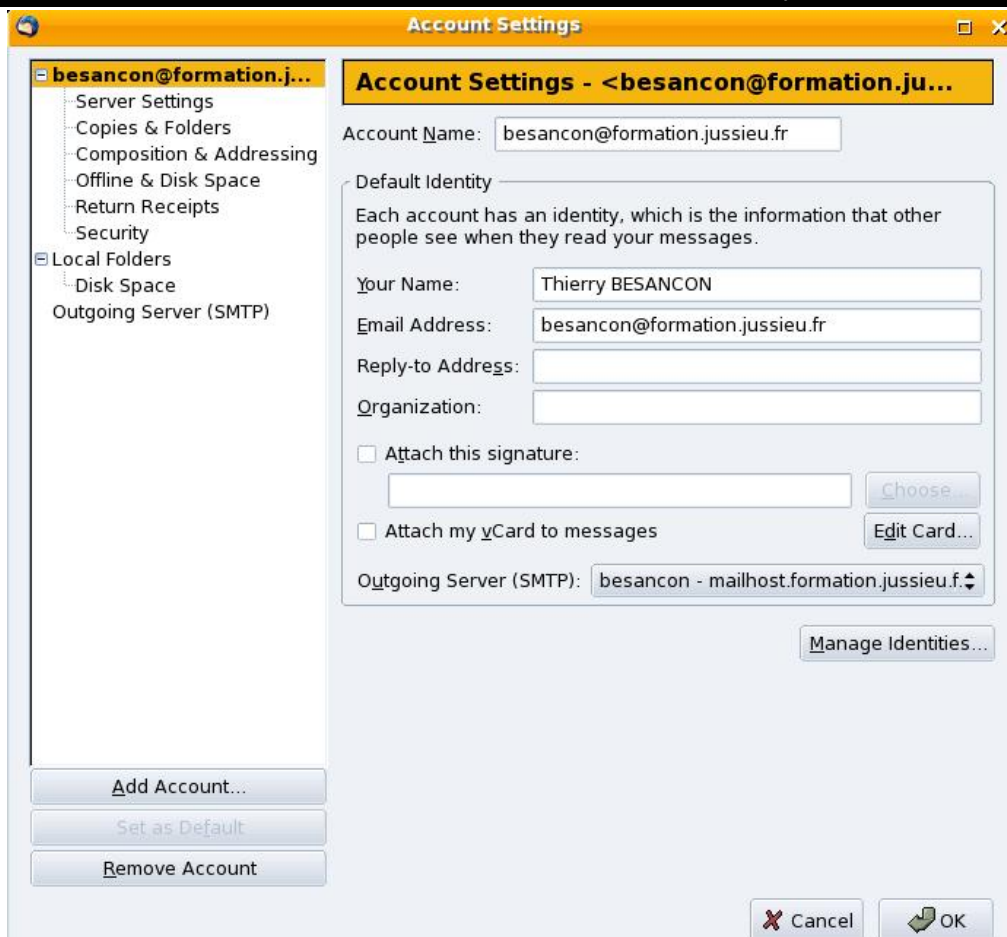
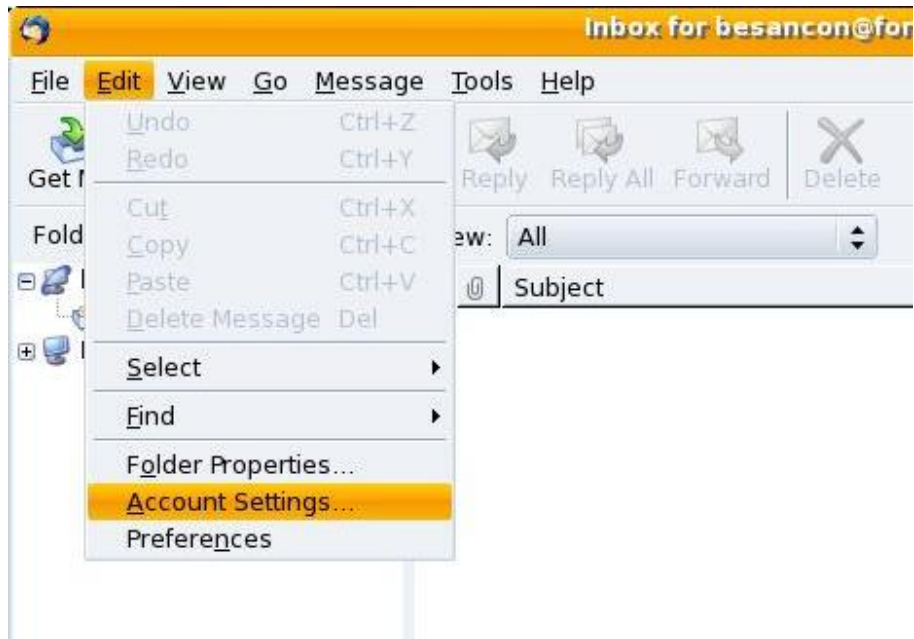
### Account Name

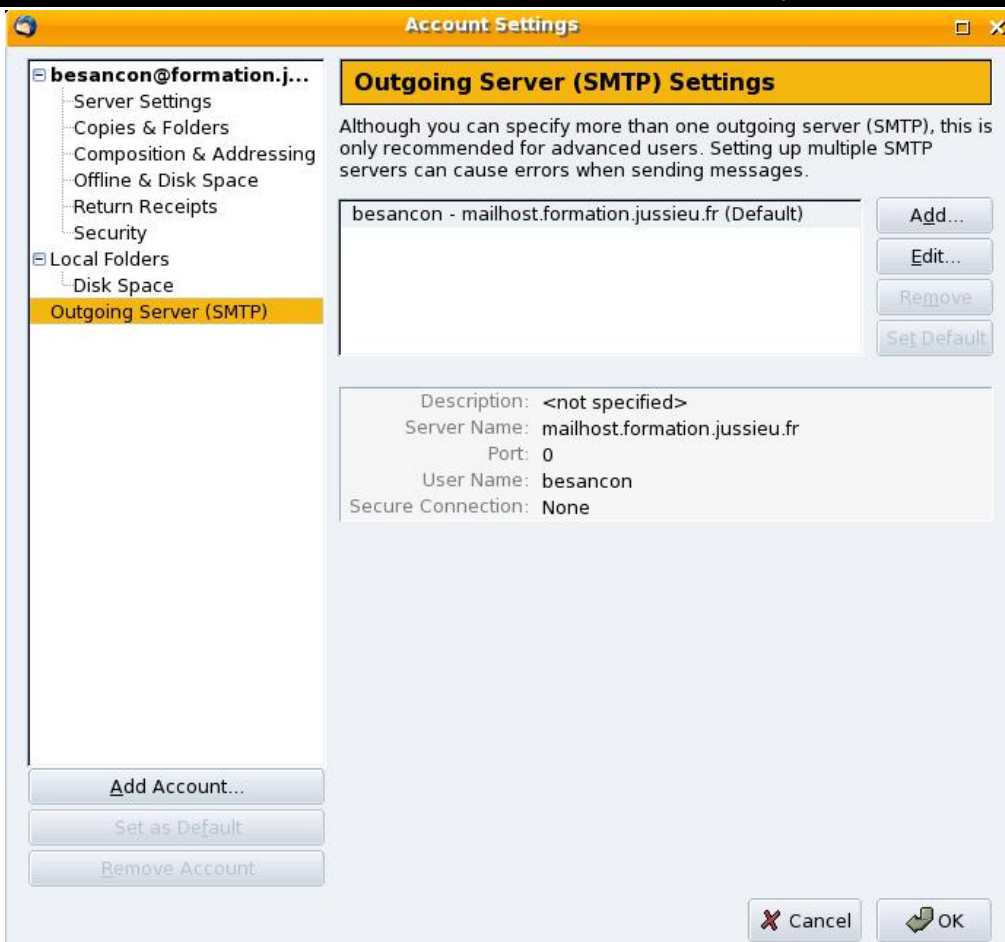
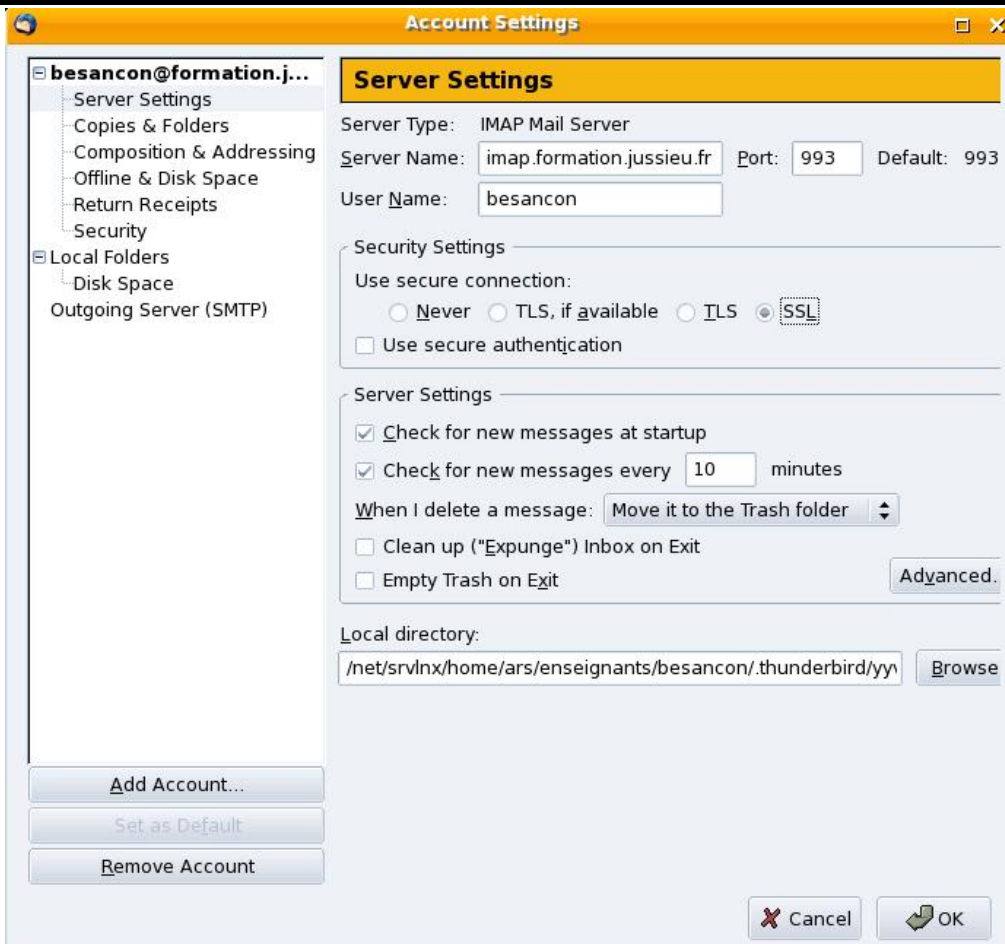
Enter the name by which you would like to refer to this account (for example, "Work Account", "Home Account" or "News Account").

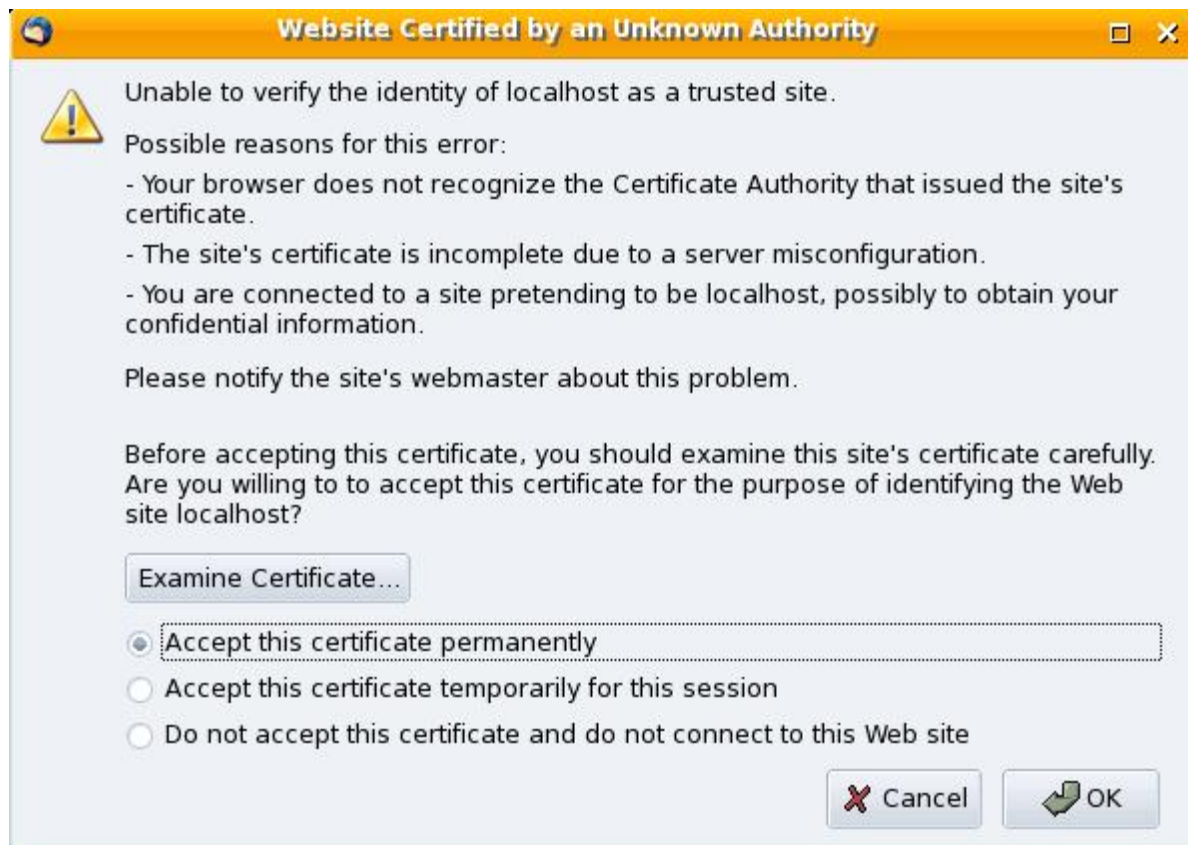
Account Name:

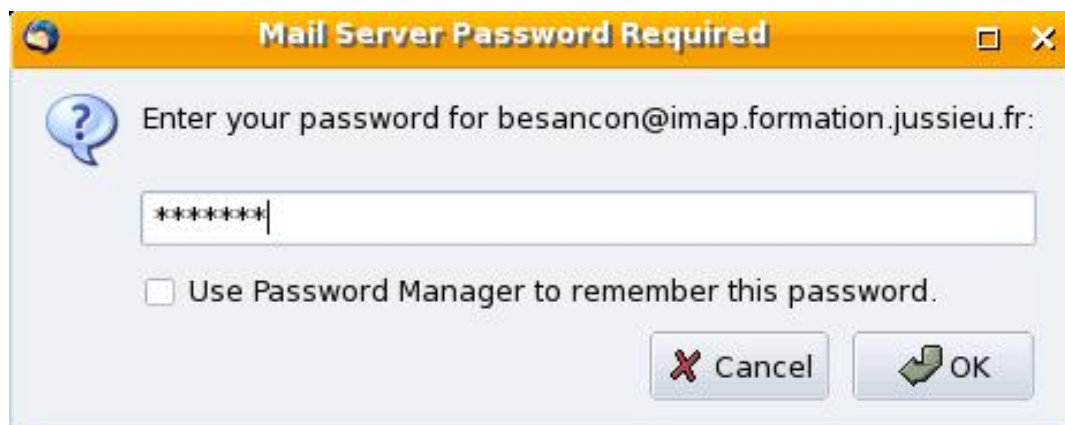


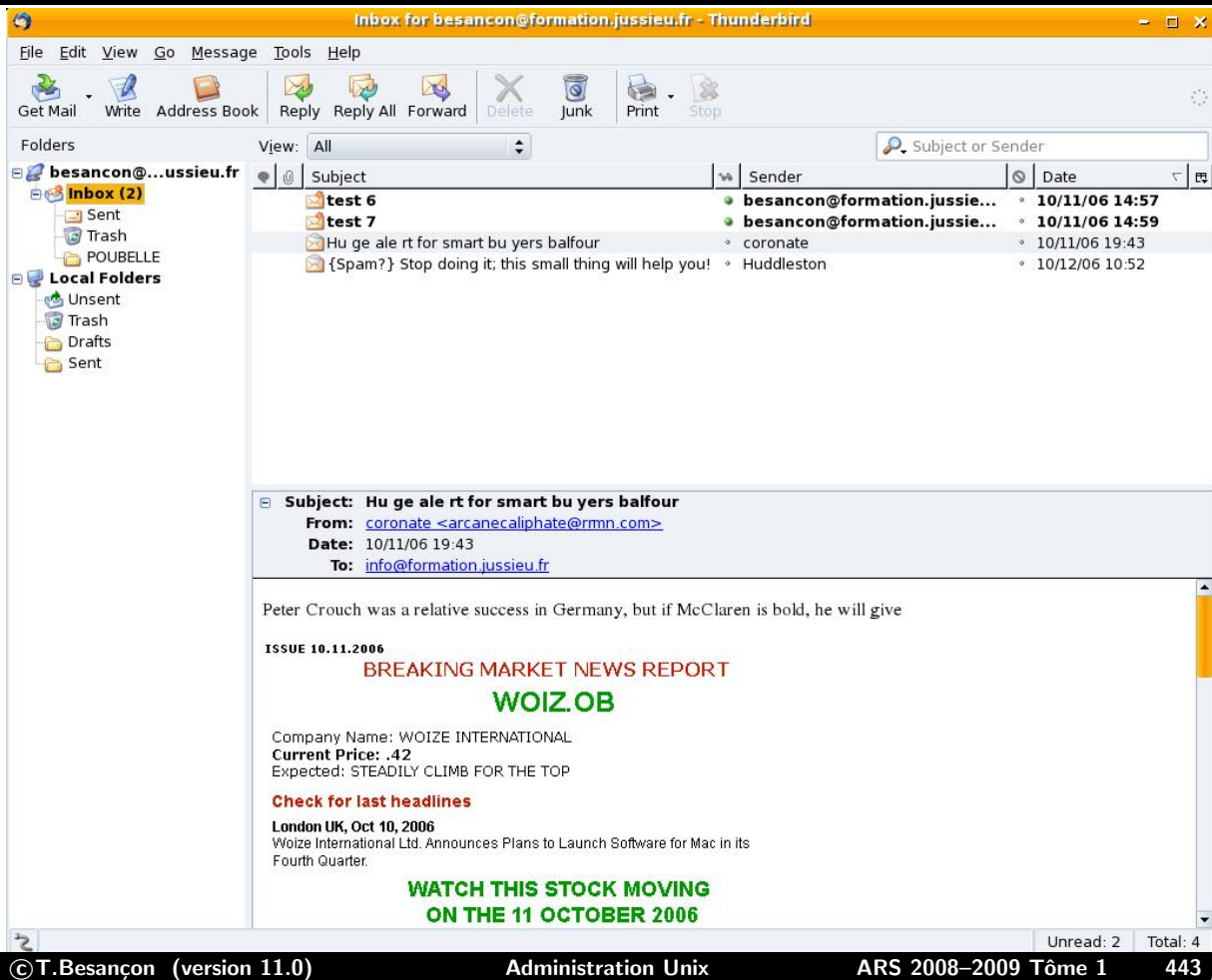












## Chapitre 12 • Commandes UNIX réseau de base

### §12.32 • Courrier électronique : webmail de la Formation Permanente

Le site web de la formation permanente offre un service de webmail.

Se reporter à « <http://www.formation.jussieu.fr/> » ou à « <https://webmail.formation.jussieu.fr/> »



## Chapitre 13

***Pratique du Bourne shell***

## Chapitre 13 • Pratique du Bourne shell

## §13.1 • Affichage d'une chaîne de caractères : echo

(en anglais *echo*)

Syntaxe : « echo caractères »

◇ Exemples 1

```
% echo quelquechose
quelquechose
```

```
% echo "quelquechose"
quelquechose
```

```
% echo "un deux"
un deux
```

```
% echo "un deux" "trois"
un deux trois
```

## ◇ Exemples 2

```
% echo ""  
<- ligne vide
```

```
% echo  
<- ligne vide
```

## Chapitre 13 • Pratique du Bourne shell

### §13.2 • Principe d'exécution par le shell d'une commande UNIX

- 1 attente d'une entrée de commande ;
- 2 traitement des caractères spéciaux de la commande ;
- 3 recherche de l'exécutable ;  
s'il n'est pas trouvé, on affiche un message d'erreur et le shell reprend à l'étape 1 ;
- 4 `fork()` + `exec()` de la commande à lancer ;
- 5 le shell fait un `wait()` de la commande ;
- 6 une fois la commande terminée, le shell reprend à l'étape 1.

Voir page 480 et page 481 aussi.

## Chapitre 13 • Pratique du Bourne shell

## §13.3 • Caractères spéciaux du shell : métacaractères

Caractères	Signification
tabulation, espace	appelés <i>white characters</i> ; délimiteurs des mots ; un tel caractère au minimum
retour charriot	fin de la commande à exécuter
;	séparateur de commandes sur une seule ligne
( )	exécution des commandes dans un sous shell
{ }	exécution des commandes en série
&	lance une commande en tâche de fond
' " \	appelés <i>quote characters</i> ; changent la façon dont le shell interprète les caractères spéciaux
< > << >> `	caractères de redirection d'entrées/sorties
* ? [ ] [ ^ ]	caractères de substitution de noms de fichiers
\$	valeur d'une variable

## Chapitre 13 • Pratique du Bourne shell

## §13.4 • Métacaractères tabulation, espace

Délimiteurs des mots de la ligne de commande : espace, tabulation

Un caractère délimiteur au minimum entre chaque mot de la ligne de commande.



## Chapitre 13 • Pratique du Bourne shell

## §13.5 • Métacaractère retour charriot

Le retour charriot valide la ligne entrée.

La présence du « \ » neutralise (voir page 475) le sens spécial du métacaractère retour charriot.

Exemple :

```
% echo ananas \  
>>banane \  
>>cerise  
ananas banane cerise
```

## Chapitre 13 • Pratique du Bourne shell

## §13.6 • Métacaractère point-virgule

Le point-virgule sert à lancer plusieurs commandes UNIX sur la même ligne de commande.

Exemple :

La commande

```
% date ; ls -l exemple.txt  
Thu Jul  8 19:49:19 MEST 2004  
-rw-r--r--  1 besancon ars          87 Jul  6 19:25 exemple.txt
```

est équivalente à

```
% date  
Thu Jul  8 19:49:19 MEST 2004  
% ls -l exemple.txt  
-rw-r--r--  1 besancon ars          87 Jul  6 19:25 exemple.txt
```

### ◇ Retour sur find :

Suite au sens de métacaractères de « ; » dans le shell, on écrit :

```
% find . \( -name 'a.out' -o -name '*.o' \) -atime +7 -exec rm {} \;
```

La présence du « \ » neutralise (voir page 475) le sens spécial du métacaractère « ; ».

## Chapitre 13 • Pratique du Bourne shell

### §13.7 • Métacaractères parenthèses ( )

Les parenthèses délimitent un bloc de commandes exécutées dans un second shell.

Exemple 1 :

```
% pwd ; ( cd /tmp ; ls -l ; pwd ) ; pwd
/home/besancon
total 2560
-rw-r--r--  1 besancon adm    7824 Oct  9 08:53 20060476.pdf
-rw-----  1 besancon adm     419 Sep 12 00:55 Acro000F5aivb
-rw-r--r--  1 besancon adm  10799 Oct  5 00:04 fvwmrcICqrb
drwxr-xr-x  2 besancon adm    117 Sep 12 00:46 hsperfddata_besancon
-rw-----  1 root      root   3533 Sep 11 21:13 pg_hba.conf
-rw-r--r--  1 root      root   1914 Sep 14 00:09 php.errors
-rw-----  1 root      root  13666 Sep 11 21:13 postgresql.conf
drwx-----  2 besancon adm    183 Jul 27 19:43 ssh-yXbXQ635
/tmp
/home/besancon
```

Le changement de directory se fait dans le second shell.

## Exemple 2 :

```
% a=3 ; echo $a ; ( a=5 ; echo $a ) ; echo $a
3
5
3
```

La valeur 5 est affectée à la variable « a » du second shell et disparaît avec lui.

◇ Retour sur find :

Suite au sens de métacaractères de « ( ) » dans le shell, on écrit :

```
% find . \( -name 'a.out' -o -name '*.o' \) -atime +7 -exec rm {} \;
```

La présence du « \ » neutralise (voir page 475) le sens spécial des métacaractères « ( ) ».

## Chapitre 13 • Pratique du Bourne shell

### §13.8 • Contrôle des commandes lancées : &, fg, bg, kill, ^C, ^Z

#### ◇ Avant plan

Lorsqu'une commande est en train de s'exécuter, le shell ne rend pas la main et attend que la commande se termine (correctement ou incorrectement).

Pour interrompre prématurément une commande : taper sur la touche **Ctrl** **et aussi** sur la touche **C** du clavier. Cela tue la commande qui tournait.

On notera l'appui sur ces 2 touches par « `Ctrl-C` » ou par « `^C` ».

#### ◇ Arrière plan

Si l'on veut une lancer une commande et récupérer la main tout de suite, avant même que la commande ait fini de s'exécuter, il faut lancer la commande par :

% commande &

Le signe « & » signifie de lancer en **tâche de fond**, en **background** la commande. Sans ce signe, la commande est lancée en **premier plan**, en **foreground**.

### ◇ Passage d'avant plan en arrière plan

Pour passer en background une commande lancée en foreground :

#### 1 Figer la commande en cours

Taper sur la touche « Control » **et aussi** sur la touche « Z », soit « Ctrl-Z » ou « ^Z » :

```
% commande
...
^Z
[1]+  Stopped                  commande
```

#### 2 Indiquer de l'exécuter dorénavant en background

Taper la commande « bg » :

```
% bg
[1]+  commande &
```

**D'une façon générale, un débutant UNIX doit proscrire l'utilisation de « Ctrl-Z ». Dans 9 cas sur 10, c'est « Ctrl-C » qu'il doit employer. On évitera ainsi une saturation de la machine avec des commandes suspendues et en attente d'être tuées ou relancées.**

### ◇ Passage d'arrière plan en avant plan

Pour passer en foreground une commande lancée en background :

#### 1 La commande est lancée

On a la main :

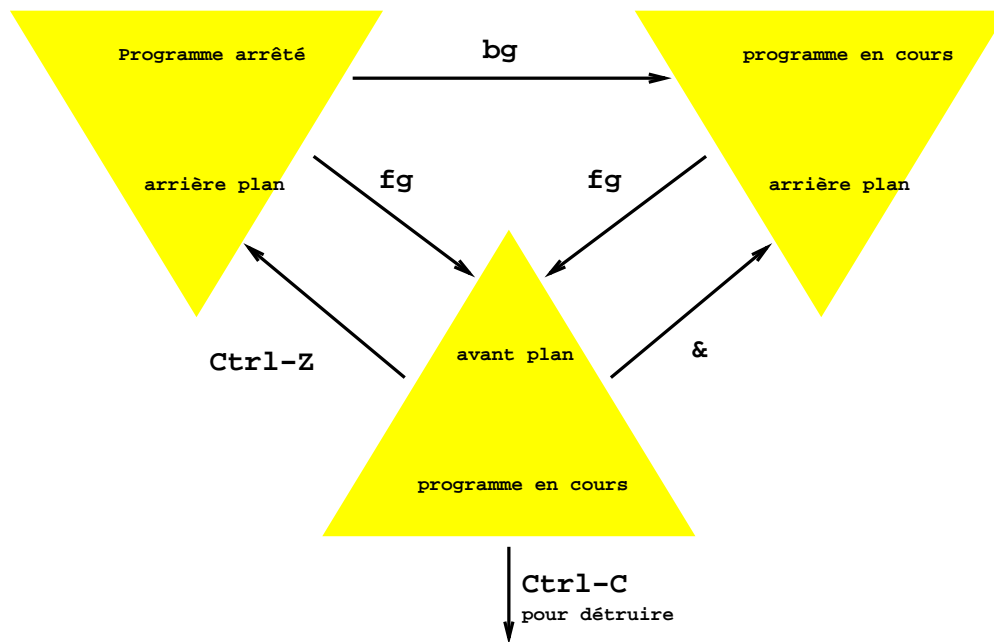
```
% commande &
...
```

#### 2 Indiquer de l'exécuter dorénavant en foreground

Taper la commande « fg » :

```
% fg
```

## ◇ En résumé



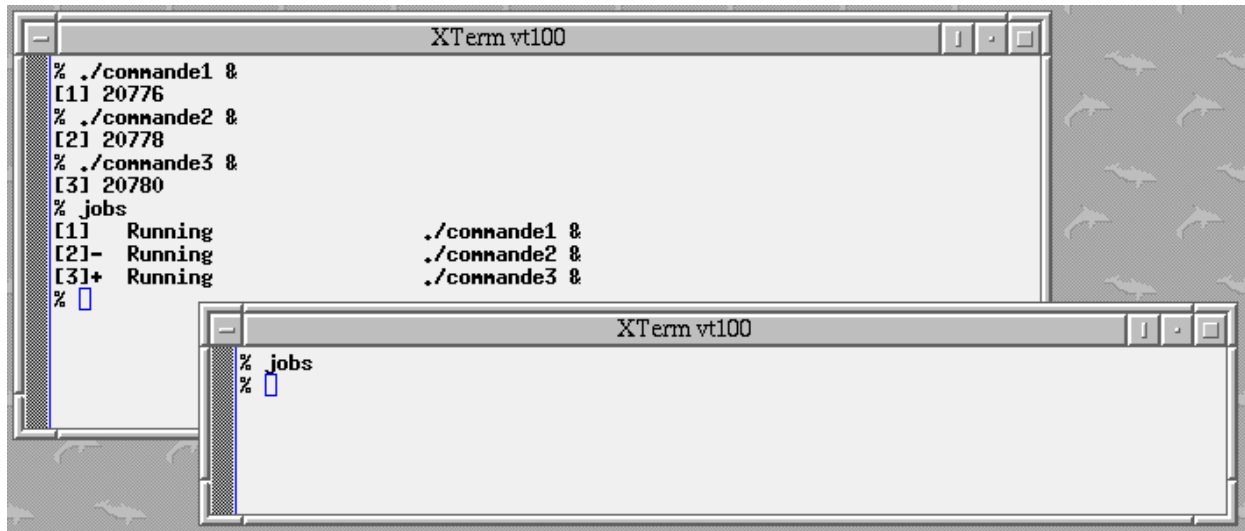
## ◇ Liste des processus en arrière plan

Pour connaître la liste des commandes en background :

```

% jobs
[1]  Running                commande1 &
[2]  Running                commande2 &
[3]  Running                commande3 &
[4]- Running                commande4 &
[5]+ Running                commande5 &
  
```

On peut avoir plusieurs commandes en background. D'où une numérotation des commandes qui sont affichées. Ce numéro peut être repris dans les commandes « fg » et « bg » ainsi que dans la commande suivante, « kill ».



### ◇ Tuer un processus en arrière plan

Pour tuer une commande en background :

```

% jobs
[1]  Running                commande1 &
[2]  Running                commande2 &
[3]  Running                commande3 &
[4]- Running                commande4 &
[5]+ Running                commande5 &
% kill %3
[3]  Terminated           commande3 &
% jobs
[1]  Running                commande1 &
[2]  Running                commande2 &
[4]- Running                commande4 &
[5]+ Running                commande5 &

```

(en anglais *processus*)

Les commandes « fg », « bg », « jobs » ne fonctionnent que sur les processus lancés par le shell courant. Les commandes vues précédemment peuvent donc être inutilisables si vous avez quitté votre shell.

La commande « ps » plus générale permet d'avoir des informations sur tous les processus de la machine.

2 syntaxes selon l'UNIX de la machine :

- syntaxe de la famille BSD ; FreeBSD, OpenBSD, NetBSD, etc.
- syntaxe de la famille System-V ; SUN, HP, IBM, LINUX, etc.

### ◇ Syntaxe de la famille d'UNIX BSD

- les processus associés à son terminal : « ps »
- tous ses processus : « ps -x »
- tous les processus de la machine : « ps -ax »
- tous les processus de la machine avec les noms de login associés :  
« ps -aux »

Exemple (partiel) de « ps -aux » :

```
% ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  1130    52 ?        S    Oct23   0:06  init
root         2  0.0  0.0     0     0 ?        SW   Oct23   0:00  [kflushd]
root         3  0.0  0.0     0     0 ?        SW   Oct23   0:01  [kupdate]
...
nobody     476  0.0  0.1  1300    44 ?        S    Oct23   0:01  [identd]
daemon     490  0.0  0.0  1144     0 ?        SW   Oct23   0:00  [atd]
xfs        636  0.0  0.3  2820   120 ?        S    Oct23   0:18  xfs -droppriv -da
root     14703  0.0  0.0   2256     0 tty1    SW   Oct25   0:00  [login]
root     9813  0.0  0.0   6912     0 ?        SW   Oct31   0:09  [kdm]
idiri    20810  0.0  0.0   6552     0 ?        SW   15:13   0:01  [kwm]
idiri    20863  0.0  0.0   1996     0 pts/0   SW   15:13   0:00  [tcsh]
besancon 21785  0.0  1.3   1732   416 pts/1   S    15:25   0:00  -bash
idiri    23660  0.2  1.5   1860   472 tty2    S    16:39   0:01  vi probleme6.c
...
```



## ◇ Syntaxe de la famille d'UNIX System-V

- les processus associés à son terminal : « ps »
- tous les processus de la machine avec les noms de login associés :  
« ps -edf »

Exemple (partiel) de « ps -edf » :

```
% ps -edf
  UID    PID  PPID  C    STIME TTY      TIME CMD
  root     0     0  0  09:09:47 ?        0:01 sched
  root     1     0  0  09:09:47 ?        0:02 /etc/init -
  root     2     0  0  09:09:47 ?        0:00 pageout
  root     3     0  0  09:09:47 ?        0:52 fsflush
  root    181     1  0  09:12:07 ?        0:06 /usr/lib/autofs/automountd
...
  daemon  283     1  0  09:12:12 ?        0:11 /usr/sbin/lpd
  root    291     1  0  09:12:13 ?        0:00 /usr/local/apache/bin/httpd
  root    296     1  0  09:12:14 ?        0:00 /usr/local/admin/lib/idled
  nobody 15130   291  0  23:30:56 ?        0:00 /usr/local/apache/bin/httpd
besancon 16463 16461  0  00:12:26 pts/0    0:00 -csh
...
```

## Chapitre 13 • Pratique du Bourne shell

### §13.10 • Contrôle des processus : kill

(en anglais *kill*)

La commande « kill » sert à communiquer avec des processus :

- arrêt de processus
- demande au processus de se reconfigurer
- passage en mode verbeux du processus
- etc.

La commande « kill » existe sur tous les UNIX et il n'y a pas de différence de fonctionnement selon les UNIX.

2 syntaxes possibles :

- syntaxe numérique : « kill -9 2878 »
- syntaxe symbolique : « kill -KILL 2878 »

Nom en langage C	Nom pour la commande « kill »	Valeur	Comportement	Sens
SIGHUP	HUP	1	Exit	Hangup
SIGINT	INT	2	Exit	Interrupt
SIGQUIT	QUIT	3	Core	Quit
SIGILL	ILL	4	Core	Illegal Instruction
SIGTRAP	TRAP	5	Core	Trace or Breakpoint Trap
SIGABRT	ABRT	6	Core	Abort
SIGEMT	EMT	7	Core	Emulation Trap
SIGFPE	FPE	8	Core	Arithmetic Exception
SIGKILL	KILL	9	Exit	Killed
SIGBUS	BUS	10	Core	Bus Error
SIGSEGV	SEGV	11	Core	Segmentation Fault
SIGSYS	SYS	12	Core	Bad System Call
SIGPIPE	PIPE	13	Exit	Broken Pipe
SIGALRM	ALRM	14	Exit	Alarm Clock
SIGTERM	TERM	15	Exit	Terminated
SIGUSR1	USR1	16	Exit	User Signal 1

Nom en langage C	Nom pour la commande kill	Valeur	Comportement	Sens
SIGUSR2	USR2	17	Exit	User Signal 2
SIGCHLD	CHLD	18	Ignore	Child Status Changed
SIGPWR	PWR	19	Ignore	Power Fail or Restart
SIGWINCH	WINCH	20	Ignore	Window Size Change
SIGURG	URG	21	Ignore	Urgent Socket Condition
SIGPOLL	POLL	22	Exit	Pollable Event
SIGSTOP	STOP	23	Stop	Stopped (signal)
SIGTSTP	TSTP	24	Stop	Stopped (user)
SIGCONT	CONT	25	Ignore	Continued
SIGTTIN	TTIN	26	Stop	Stopped (tty input)
SIGTTOU	TTOU	27	Stop	Stopped (tty output)
SIGVTALRM	VTALRM	28	Exit	Virtual Timer Expired
SIGPROF	PROF	29	Exit	Profiling Timer Expired
SIGXCPU	XCPU	30	Core	CPU time limit exceeded
SIGXFSZ	XFSZ	31	Core	File size limit exceeded
SIGWAITING	WAITING	32	Ignore	Concurrency signal reserved by threads library

Les signaux les plus utiles sont :

- SIGHUP

Cela envoie l'équivalent du « Ctrl-C » du clavier.

- SIGKILL

Cela envoie un signal que le processus est obligé de suivre et qui se traduira inélectablement par la mort du processus.

## Chapitre 13 • Pratique du Bourne shell

### §13.11 • Contrôle des processus : top

Inconvénient de « ps » : c'est la liste des processus à un instant t.

On ne pourra jamais sous UNIX avoir la liste des processus en cours : le temps de chercher les processus et de faire le rapport, certains processus peuvent avoir disparu.

Amélioration de « ps » : la commande « top » (n'est cependant pas standard sur tous les UNIX)

Son intérêt : elle affiche une liste des processus toutes les *n* secondes

URL : <ftp://ftp.groupsys.com/pub/top/>

```

XTerm vt100
last pid: 21509; load averages: 0.02, 0.03, 0.04 14:52:22
71 processes: 69 sleeping, 1 running, 1 on cpu
CPU states: 98.2% idle, 0.6% user, 0.6% kernel, 0.6% iowait, 0.0% swap
Memory: 128M real, 141M swap in use, 445M swap free

  PID USERNAME THR PRI NICE  SIZE  RES STATE  TIME  CPU COMMAND
 21507 thb      1  58   0 2544K 1592K cpu    0:00  0.73% top
   326 thb      1  48   0   28M   17M run    77:44  0.52% Xsun
 21486 thb      1  48   0 2576K 1904K sleep  0:00  0.24% bash
 21485 thb      1  58   0 4120K 3128K sleep  0:00  0.14% xterm
 21509 thb      1  58   0 2920K 1616K sleep  0:00  0.14% xwd
 18503 thb      1  58   0 4128K 2760K sleep  0:06  0.11% xterm
   327 thb      1  58   0 2704K 1432K sleep  1:29  0.05% fvwm
 18504 thb      1  48   0 2592K 1736K sleep  0:02  0.03% bash
   539 thb      1  59   0   48M   14M sleep  24:41  0.01% emacs-20.4
   362 thb      1  13  15 3360K 1560K sleep  2:04  0.00% xbuffy
20528 thb      1  59   0   47M   25M sleep  5:46  0.00% netscape
   215 root     10  51   0 3040K 1880K sleep  0:50  0.00% nscd
     1 root      1  58   0   776K  144K sleep  0:31  0.00% init
   275 root      1  58   0  1896K  552K sleep  0:22  0.00% sfsfd1
   133 root      1  58   0  1856K   696K sleep  0:10  0.00% in.ndpd

```

## Chapitre 13 • Pratique du Bourne shell

### §13.12 • (Windows : : Contrôle des processus : taskmgr.exe)

Le programme « taskmgr.exe » offre une fonctionnalité du genre de « top ».

Image Name	PID	User Name	Session ID	CPU	CPU Time	Mem Usage	USER Objects
HardCopy.exe	2620	besancon	0	00	0:00:00	6 584 K	149
DVDIdlePro.exe	2504	besancon	0	00	0:00:02	592 K	16
taskmgr.exe	2184	besancon	0	01	0:00:01	3 324 K	125
PowerDVD.exe	2092	besancon	0	24	0:02:09	798 476 K	53
BCMWLTRY.EXE	2036	SYSTEM	0	00	0:00:00	652 K	0
ctfmon.exe	1988	besancon	0	00	0:00:00	504 K	9
realsched.exe	1872	besancon	0	00	0:00:00	84 K	4
VPTray.exe	1856	besancon	0	00	0:00:00	768 K	11
vmware-authd.exe	1852	SYSTEM	0	00	0:00:00	724 K	0
quickset.exe	1772	besancon	0	00	0:00:00	452 K	28
Apoint.exe	1756	besancon	0	00	0:00:00	848 K	33
svchost.exe	1752	SYSTEM	0	00	0:00:00	1 644 K	0
BCMSMMMSG.exe	1688	besancon	0	00	0:00:00	400 K	2
svchost.exe	1660	SYSTEM	0	00	0:00:01	5 192 K	0
nvsvc32.exe	1620	SYSTEM	0	00	0:00:00	300 K	0
explorer.exe	1536	besancon	0	00	0:00:09	14 836 K	224
svchost.exe	1220	SYSTEM	0	00	0:00:00	1 168 K	0
lsass.exe	1044	SYSTEM	0	00	0:00:00	1 104 K	0
services.exe	1032	SYSTEM	0	00	0:00:01	684 K	0
winlogon.exe	988	SYSTEM	0	00	0:00:00	1 368 K	0
acrotray.exe	968	besancon	0	01	0:00:00	352 K	13
csrss.exe	952	SYSTEM	0	00	0:00:02	1 064 K	0
ApntEx.exe	928	besancon	0	00	0:00:00	236 K	5
WLTRY SVC.EXE	892	SYSTEM	0	00	0:00:00	92 K	0
smss.exe	868	SYSTEM	0	00	0:00:00	52 K	0

Processes: 33 CPU Usage: 26% Commit Charge: 973M / 2462M

Caractères	Nom	Description
'	single quote	le shell n'interprète aucun caractère spécial entre deux '
"	double quote	le shell n'interprète aucun caractère spécial à l'exception de \$ ` et \
\	backslash	le shell n'interprète pas le caractère spécial suivant le backslash

Exemples d'utilisation pour conserver les espaces dans les chaînes de caractères :

- 1 % echo un        deux  
un deux
- 2 % echo "un        deux"  
un        deux
- 3 % echo 'un        deux'  
un        deux
- 4 % echo un\ \ \ \ \ deux  
un        deux

Exemples d'utilisation pour afficher des apostrophes ou des guillemets :

- 1 % echo c'est aujourd'hui  
cest aujourd'hui <- **pas d'apostrophe**
- 2 % echo c\'est aujourd\'hui  
c'est aujourd'hui
- 3 % echo "c'est aujourd'hui"  
c'est aujourd'hui
- 4 % echo \"anas\"  
"anas"
- 5 % echo 'anas'  
"anas"
- 6 % echo ""anas""  
anas
- 7 % echo ' ''  
"
- 8 % echo '' ''  
'

Exemples de ce qui est interprété ou pas selon que l'on a des guillemets ou des apostrophes ou des backslashes :

- 1 % echo "\$HOME"  
/net/serveur/home/ars/enseignants/besancon
- 2 % echo '\$HOME'  
\$HOME
- 3 % echo "\\$HOME"  
\$HOME
- 4 % echo '\\$HOME'  
\\$HOME

### ◇ Retour sur grep :

Pour neutraliser tout caractère dans une regexp que le shell pourrait vouloir interpréter, on écrit :

```
% grep -E '^[a-e]' exemple.txt
```

### ◇ Retour sur sed :

Pour neutraliser tout caractère dans une regexp que le shell pourrait vouloir interpréter, on écrit :

```
% sed -e 's/^[a-e]//' exemple.txt
```

## Chapitre 13 • Pratique du Bourne shell

### §13.14 • Lancement d'une commande par le shell

En interne, le shell lance une commande par la fonction « `execl()` » du langage C (après un « `fork()` ») :

```
% ls -l -----> execl("/bin/ls",
                    "ls",          <--> argv[0] <--> $0
                    "-l",         <--> argv[1] <--> $1
                    NULL
                    ) ;
```

Se reporter à la page de manuel de « `exec(2)` ».

## Chapitre 13 • Pratique du Bourne shell

### §13.15 • Interprétation de la ligne de commande

La difficulté :

- certains caractères sont pour le shell
- certains caractères sont pour la commande invoquée
- le shell se sert toujours en premier

Conséquence :

Si un caractère spécial a un sens pour le shell et pour la commande, il faut l'écrire de façon non ambiguë. Le shell doit comprendre si le caractère spécial est pour lui ou pour la commande.

#### ◇ Exemple 1

```
% rm exemple.txt    --> execl("/bin/rm",  
% ...                "rm",  
                      "exemple.txt",  
                      NULL  
                      );
```



### ◇ Exemple 2

```
% ls
fichier1.doc  fichier2.doc
% rm *.doc    --> execl("/bin/rm",
% ...        "rm",
              "fichier1.doc",
              "fichier2.doc",
              NULL
              );
```

### ◇ Exemple 3

```
% ls
ananas  cerise
% expr 2 * 3  -----> execl("/usr/bin/expr",
expr: syntax error    "expr",
% ...                 "2",
                      "ananas",
                      "cerise",
                      "3",
                      NULL
                      );
```

## ◇ Exemple 4

```
% ls
ananas cerise
% expr 2 \* 3 -----> execl("/usr/bin/expr",
6                               "expr",
% ...                            "2",
                                "* ",
                                "3",
                                NULL
                                );
```

## Chapitre 13 • Pratique du Bourne shell

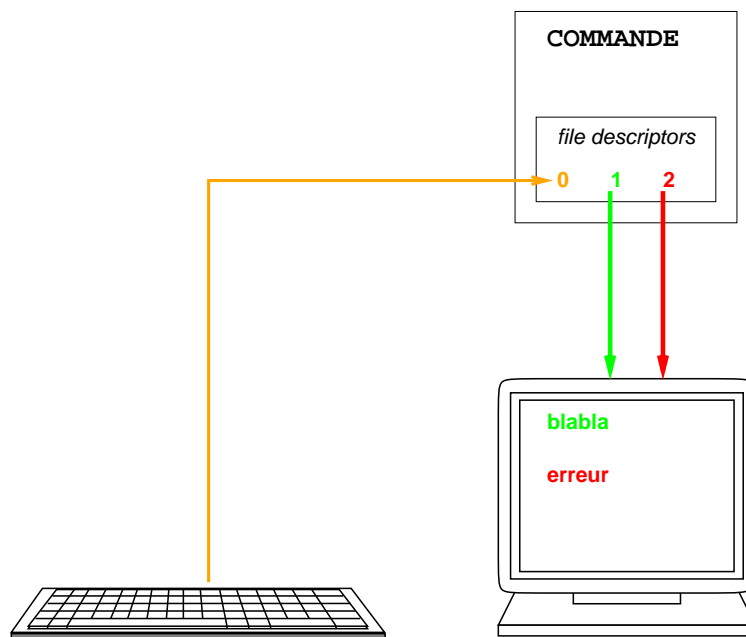
## §13.16 • File descriptors : stdin, stdout, stderr

Toutes les entrées/sorties d'UNIX sont réalisées au moyen de fichiers. Chaque processus ouvre donc un certain nombre de fichiers. Ces fichiers sont référencés en interne par une table d'entiers dits **file descriptors**.

Nom	File descriptor	Destination par défaut
standard input ( <b>stdin</b> )	0	clavier
standard output ( <b>stdout</b> )	1	écran
standard error ( <b>stderr</b> )	2	écran

Les file descriptors existent en langage C et sont profondément ancrés dans le fonctionnement interne d'UNIX.

Les file descriptors par défaut se présentent ainsi :



Le fichier système de programmation C « /usr/include/stdio.h » indique :

```
#define stdin    (&__sF[0])
#define stdout   (&__sF[1])
#define stderr   (&__sF[2])
```

Quelques petits exemples C de démonstration :

```
#include<stdio.h>
main()
{
    char line[1024];
    (void) fgets(line, 1024, stdin);

    fprintf(stdout, "Bonjour !\n");

    fprintf(stderr, "Enfer et damnation !\n");
}
```

## Chapitre 13 • Pratique du Bourne shell

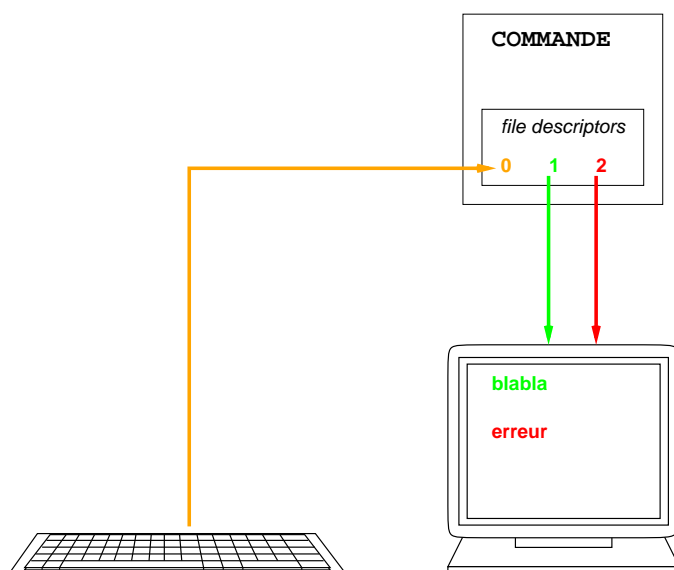
## §13.17 • Métacaractères de redirection : &lt;, &gt;, &gt;&gt;, &lt;&lt;, \, |, 2&gt;, &gt;&amp;

Canal	Format	Description
0	commande < fichier.txt	stdin de commande provient de fichier.txt
1	commande > fichier.txt	stdout de commande placé dans fichier.txt dont le contenu précédent est écrasé
1	commande >> fichier.txt	stdout de commande placé en fin de fichier.txt
1	variable='commande'	remplace 'commande' par le résultat de l'exécution de commande
1 + 0	commande1   commande2	passse le stdout de commande1 comme stdin de commande2
2	commande 2> fichier.txt	redirige stderr de commande dans fichier.txt
2	commande 2>> fichier.txt	stderr de commande placé en fin de fichier.txt

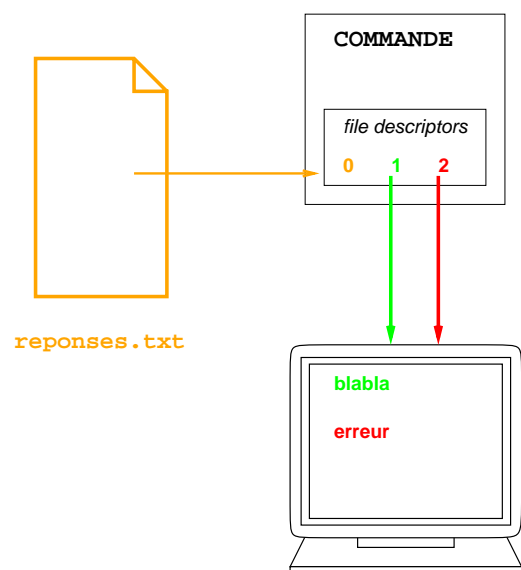
## ◇ Redirection du file descriptor 0

Comment s'organisent les file descriptors lors de « commande < reponses.txt » ?

```
% commande
```



```
% commande < reponses.txt
```



Exemples :

```
% application.exe < reponses.txt
```

Exemple : extrait d'un script qui partitionne un disque dur en 3 partitions de 128 MB, 128 MB et 255 MB respectivement.

```
% fdisk /dev/sdb <<EOF
```

```
n
```

```
p
```

```
1
```

```
+127
```

```
n
```

```
p
```

```
2
```

```
+127
```

```
n
```

```
p
```

```
3
```

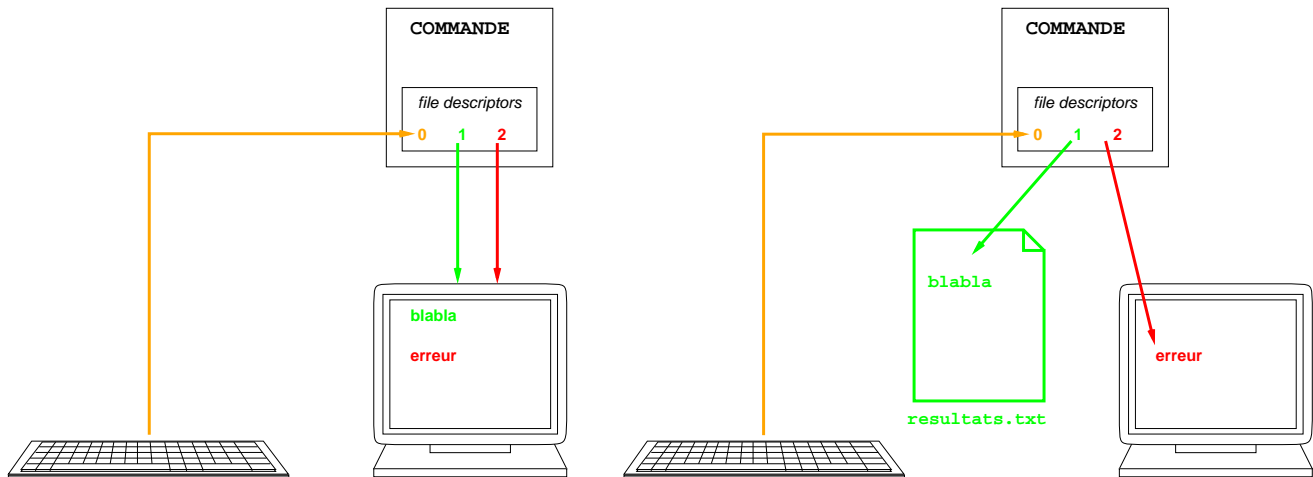
```
+255
```

## ◇ Redirection du file descriptor 1

Comment s'organisent les file descriptors lors de  
« commande > resultats.txt » ?

% commande

% commande > resultats.txt



Exemples :

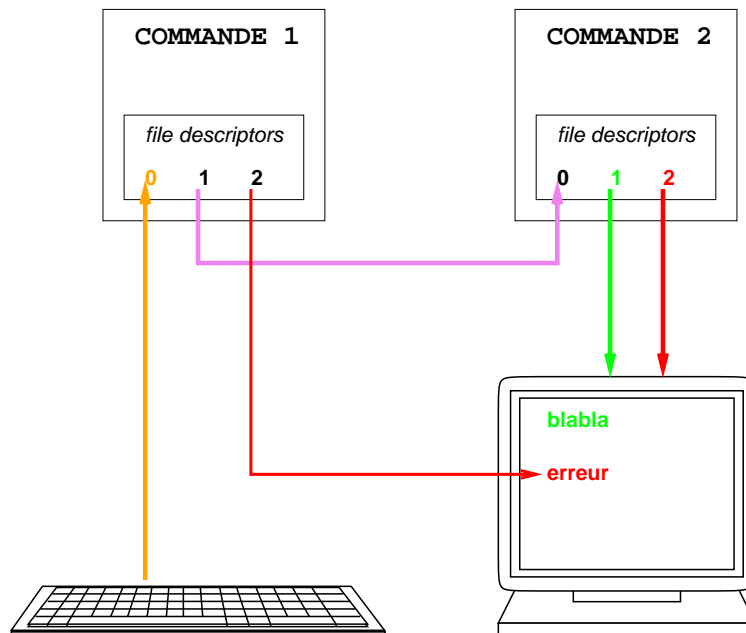
**1** % ls > /tmp/exemple.txt

**2** % ls /etc >> /tmp/exemple.txt

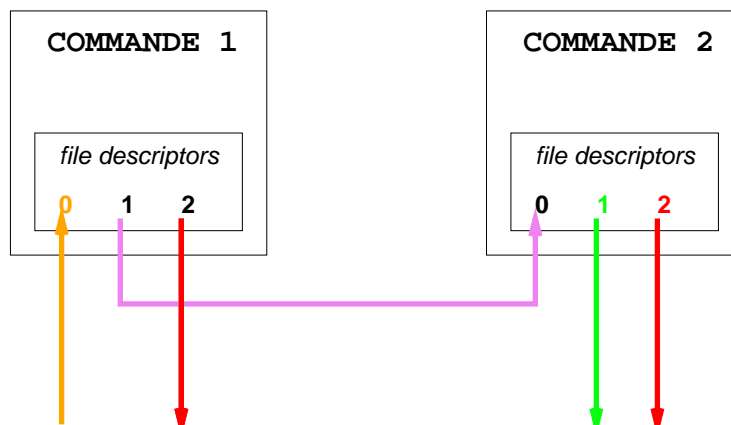
**3** % n=`wc -l /etc/passwd`  
% echo \$n  
170 /etc/passwd

### ◇ Redirection avec un pipe

Comment s'organisent les file descriptors lors de « `commande1 | commande2` » ?



Pour que « `commande1 | commande2` » fonctionne, il faut que la commande2 lise sur stdin (filedescriptor 0) !



Ce n'est pas le cas de toutes les commandes !  
Certaines commandes UNIX ne pourront jamais être utilisées dans un pipe !

## Exemples qui fonctionnent :

1 % ls -l | more

2 % cat /etc/group | more

qui équivaut à

more /etc/group

## Exemples qui ne fonctionnent pas :

1 % echo exemple.txt | ls  
ananas cerise

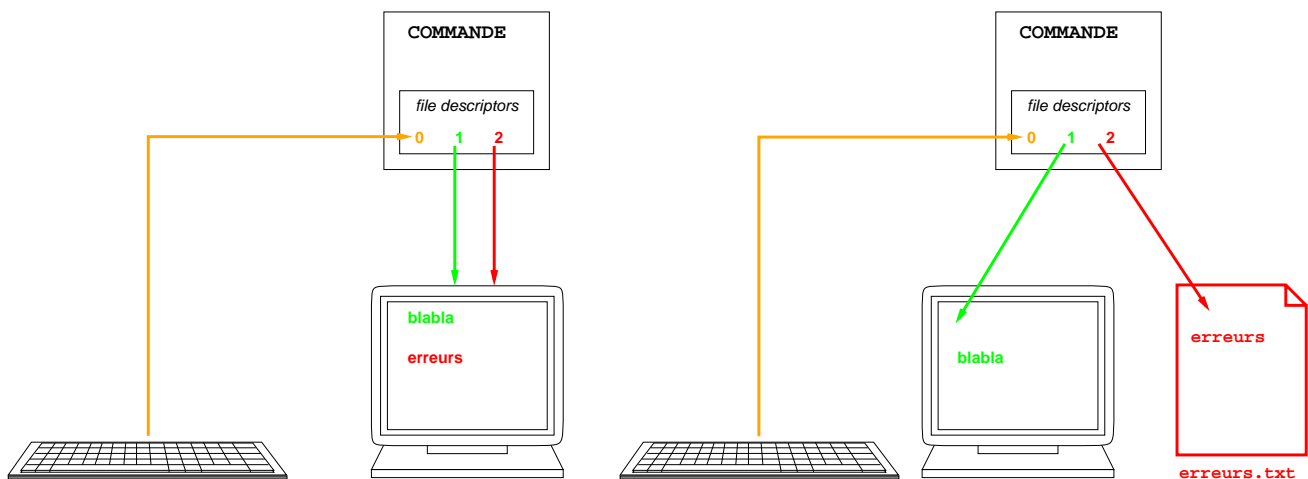
2 % echo exemple.txt | rm  
usage: rm [-fiRr] file ...

◇ Redirection du file descriptor 2

Comment s'organisent les file descriptors lors de  
« commande 2> erreurs.txt » ?

% commande

% commande 2> erreurs.txt





### Exemple :

```
% ls fichier-inexistant.txt
ls: fichier-inexistant.txt: No such file or directory

% ls fichier-inexistant.txt > erreurs.txt
ls: fichier-inexistant.txt: No such file or directory

% ls fichier-inexistant.txt 2> erreurs.txt
% cat erreurs.txt
ls: fichier-inexistant.txt: No such file or directory
```

Exemple compliqué qui montre que l'ordre des redirections est important :  
Où est le message d'erreur ? :

```
% ls exemple.txt inexistant.txt >&2 2> erreurs.txt
exemple.txt
```

Où est le message normal affiché ? :

```
% ls exemple.txt inexistant.txt 2> erreurs.txt >&2
% ...
```

Quelques explications :

- Le shell évalue la ligne de commande de gauche à droite.
- Dans la commande  
« `ls exemple.txt inexistant.txt >&2 2> erreurs.txt` »,  
stdout est redirigé sur l'écran puisque stderr équivaut à l'écran à cet instant, puis stderr est redirigé sur `erreurs.txt`. D'où le résultat.

### ◇ Moralité :

- Forme générale de lancement d'une commande en background :

```
% commande < reponses.txt > resultats.txt 2> erreurs.txt &
```

- Suivi de l'avancée du calcul : le processus tourne-t-il encore ?
  - si l'on n'a pas terminé le shell dans lequel on a lancé le calcul
    - ⇒ utiliser « jobs »
  - on a terminé le shell dans lequel on a lancé le calcul
    - ⇒ utiliser « ps » :
 

```
% ps -edf | grep commande
```

### ◇ Moralité (suite) :

- Suivi de l'avancée du calcul via l'inspection des fichiers

« resultats.txt » et « erreurs.txt » :

- consultation des fichiers par « cat » ou « more » ou autre commande de ce type :

```
% more resultats.txt
```

```
% more erreurs.txt
```

- consultation des fichiers par « tail -f » :

```
% tail -f resultats.txt
```

```
% tail -f erreurs.txt
```

(terminer un « tail -f » par Ctrl-C)

### ◇ Protection contre l'écrasement de fichiers lors de redirection

Sous le shell BASH de LINUX on peut se protéger contre l'écrasement intempestif de fichier lors d'une redirection via une variable spéciale interne de BASH :

```
% ls -l
-rw-r--r--  1 besancon ars      0 Aug  7 20:39 ananas.txt
-rw-r--r--  1 besancon ars      0 Aug  7 20:40 banane.txt
-rw-r--r--  1 besancon ars      0 Aug  7 20:40 cerise.txt
% set -o noclobber
% ls > cerise.txt
bash: cerise.txt: cannot overwrite existing file
```

## Chapitre 13 • Pratique du Bourne shell

### §13.18 • Trou noir pour redirection : /dev/null

On peut vouloir se débarrasser d'une partie de l'affichage.

Solution inefficace :

```
% application > /tmp/resultats
...
% rm /tmp/resultats
```

La solution est de rediriger vers « /dev/null » :

```
% application > /dev/null
...
```

**« /dev/null » est indispensable dans la vie de l'administrateur système.**

◇ Forme générale de lancement d'une commande en background :

- On garde tous les messages émis par le calcul :

```
% calcul < reponses.txt > resultats.txt 2> erreurs.txt &
```

- On garde les messages normaux émis par le calcul mais pas les messages d'erreur :

```
% calcul < reponses.txt > resultats.txt 2> /dev/null &
```

- On garde les messages d'erreur émis par le calcul mais pas les messages normaux :

```
% calcul < reponses.txt > /dev/null 2> erreurs.txt &
```

- On ne garde aucun message émis par le calcul :

```
% calcul < reponses.txt > /dev/null 2> /dev/null &
```

(le reste des remarques précédentes sur « jobs », « ps », « more », « cat », « tail -f » reste vrai)

## Chapitre 13 • Pratique du Bourne shell

## §13.19 • Métacaractères : \*, ?, [], [^]

**Ces metacharacters servent pour construire des noms de fichiers.**

Caractère	Description
*	0 ou plus caractères
?	1 caractère exactement
[ ]	1 caractère dans l'ensemble entre crochets
[ ^ ]	1 caractère non énuméré dans l'ensemble entre crochets

## Exemples :

```

1 % ls *
  fichier1.txt  fichier2.txt  fichier3.txt  fichier4.txt

2 % ls /etc/*.??
  /etc/locate.rc      /etc/pwd.db          /etc/spwd.db
  /etc/mail.rc        /etc/sendmail.cf

3 % ls /var/log/[lp]*
  /var/log/lastlog  /var/log/lpd-errs  /var/log/ppp.log

4 % ls /var/log/[^mw]*
  /var/log/dmesg                /var/log/ppp.log
  /var/log/dmesg.today          /var/log/sendmail.st
  /var/log/dmesg.yesterday      /var/log/setuid.today
  /var/log/lastlog              /var/log/setuid.yesterday
  /var/log/lpd-errs             /var/log/slip.log

```

## Chapitre 13 • Pratique du Bourne shell

## §13.20 • Métacaractère \$ et variables shell

A un shell sont associées des variables uniquement définies dans ce shell et uniquement accessibles dans ce shell.

◇ Liste des variables définies dans la session shell

```
set
```

### ◇ Assignment de variable

```
variable=valeur
```

Rappel : l'espace est un caractère spécial donc pas d'espace de part et d'autre du signe « = ».

Sinon grosses erreurs :

```
% a =3
a: not found
% a= 3
3: not found
% a = 3
a: not found
```

### ◇ Consultation de variable

Au choix :

- écriture « \$variable »
- écriture « \${variable} »

Préférer la seconde écriture. Pour la raison suivante :

```
% a=ananas
% echo $a33

% echo ${a}33
ananas33
```

### ◇ Suppression de variable

```
unset variable
```

### ◇ Variables de type numérique

Cela n'existe pas en Bourne shell. Les variables sont de type caractères.

⇒ Il est donc impossible de faire :

```
compteur=1
compteur=$compteur + 1
```

La bonne façon de faire est d'utiliser la commande UNIX « `expr` » :

```
compteur=1
compteur=`expr $compteur + 1`
```

Se reporter à la page de manuel de « `expr` » pour les autres opérations mathématiques réalisables.

### **ATTENTION : retour sur le métacaractère « \* » du shell :**

Il faut écrire :

```
% expr 2 \* 3
6
```

ou

```
% expr 2 "*" 3
6
```

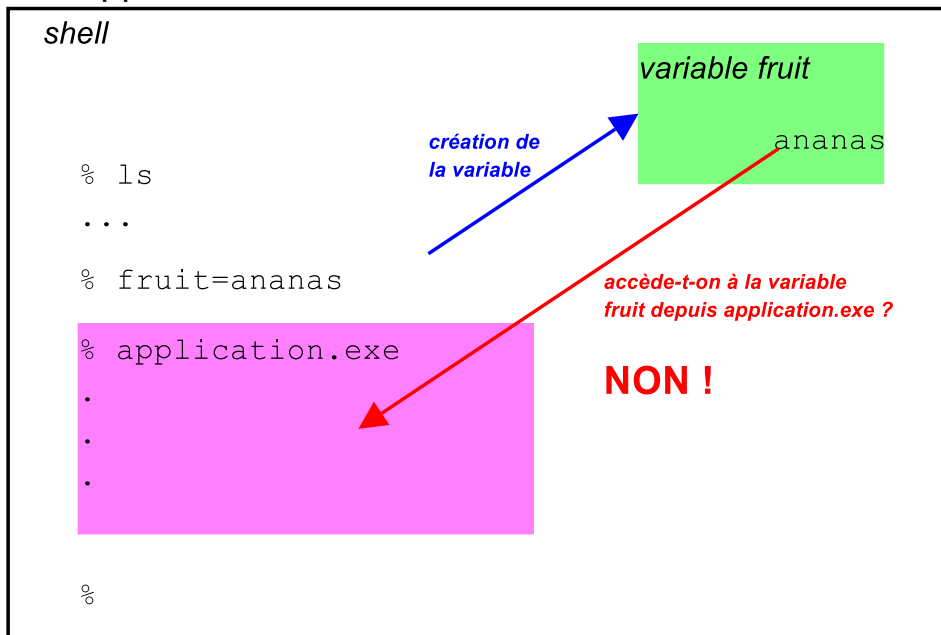
ou

```
% expr 2 '*' 3
6
```

**mais la forme ci-dessous est FAUSSE (voir page 484) :**

```
% expr 2 * 3
expr: syntax error
```

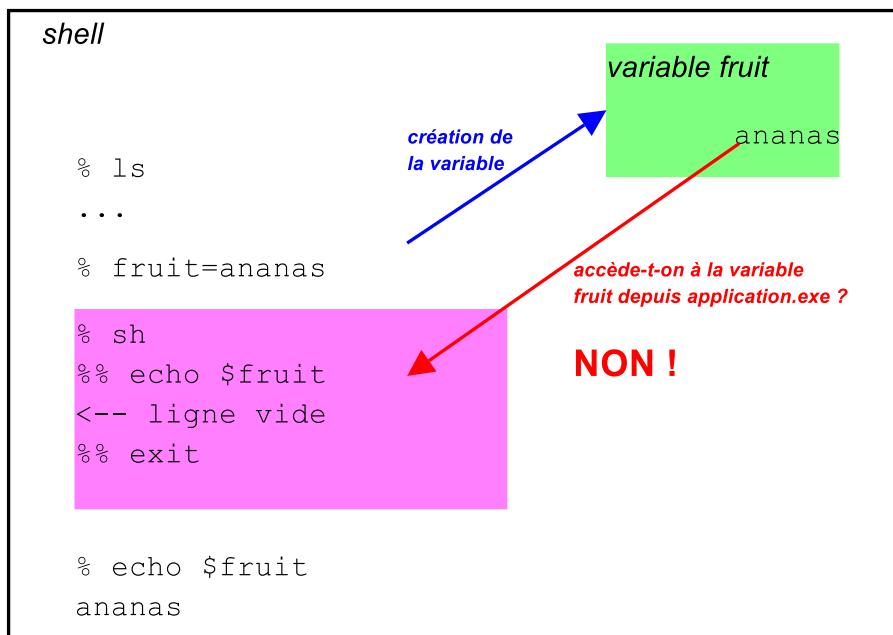
Comment les applications accèdent-elles aux variables du shell ?



⇒ Les variables ne sont pas héritées par défaut par les commandes lancées par le shell.

Vérification en lançant une application spéciale, un shell.

Intérêt du shell : le shell permet de faire des vérifications en mode interactif.



⇒ Les variables ne sont pas héritées par défaut par les commandes lancées par le shell.



La solution consiste à utiliser **export** (son contraire est **unset**) :

```
shell

% ls
...
% FRUIT=cerise
% export FRUIT

% application.exe
.
. utilisation de
. "cerise"
.

%
```

*création de la variable*

variable *FRUIT*  
cerise

*accède-t-on à la variable FRUIT depuis application.exe ?*

**OUI !**

NB : traditionnellement, les variables d'environnement sont écrites en lettres majuscules

Vérification avec un shell :

```
shell

% ls
...
% FRUIT=cerise
% export FRUIT

% sh
%% echo $FRUIT
cerise
%% exit

% echo $FRUIT
cerise
```

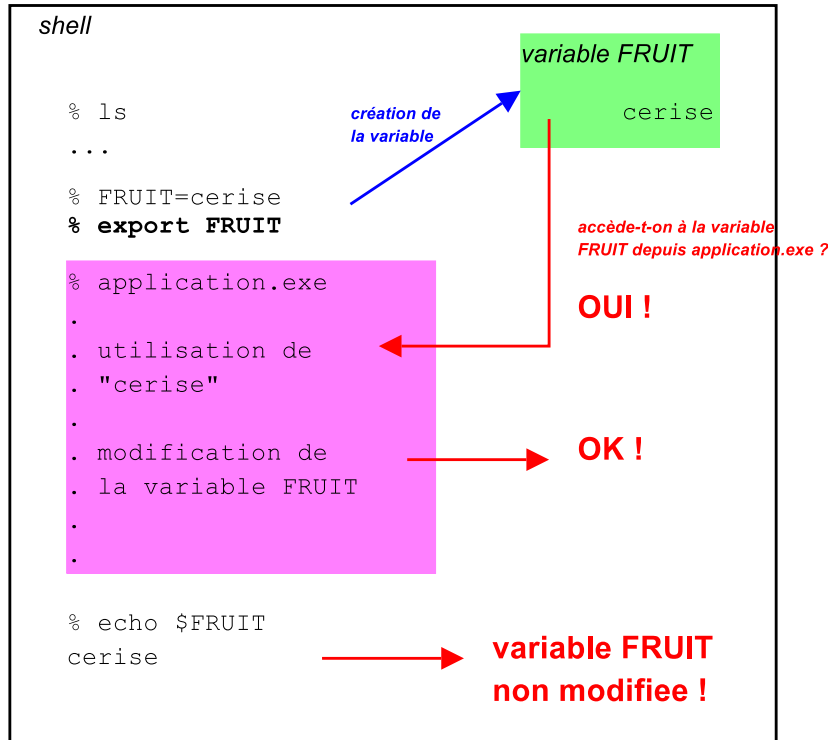
*création de la variable*

variable *FRUIT*  
cerise

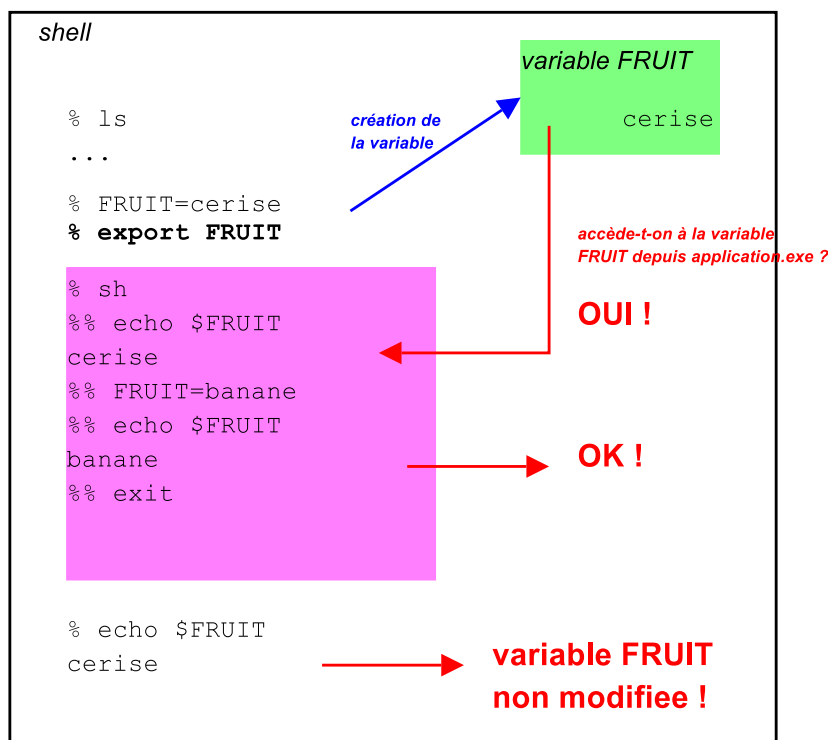
*accède-t-on à la variable FRUIT depuis application.exe ?*

**OUI !**

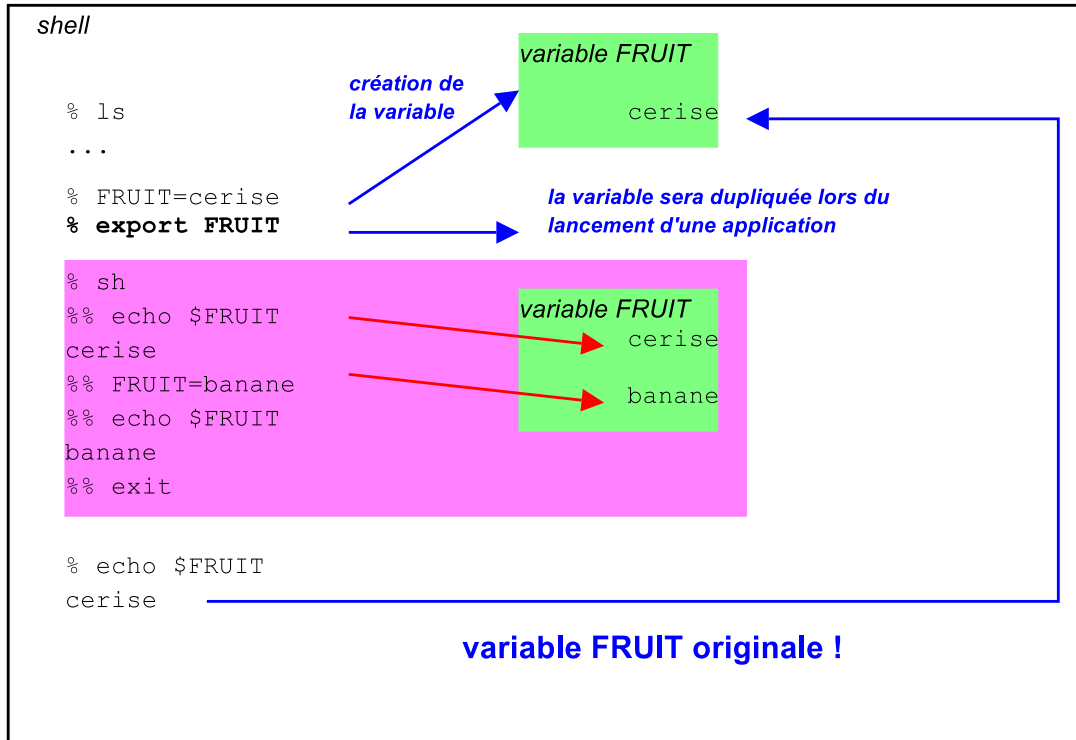
ATTENTION!!! : L'environnement est hérité en accès en lecture uniquement.



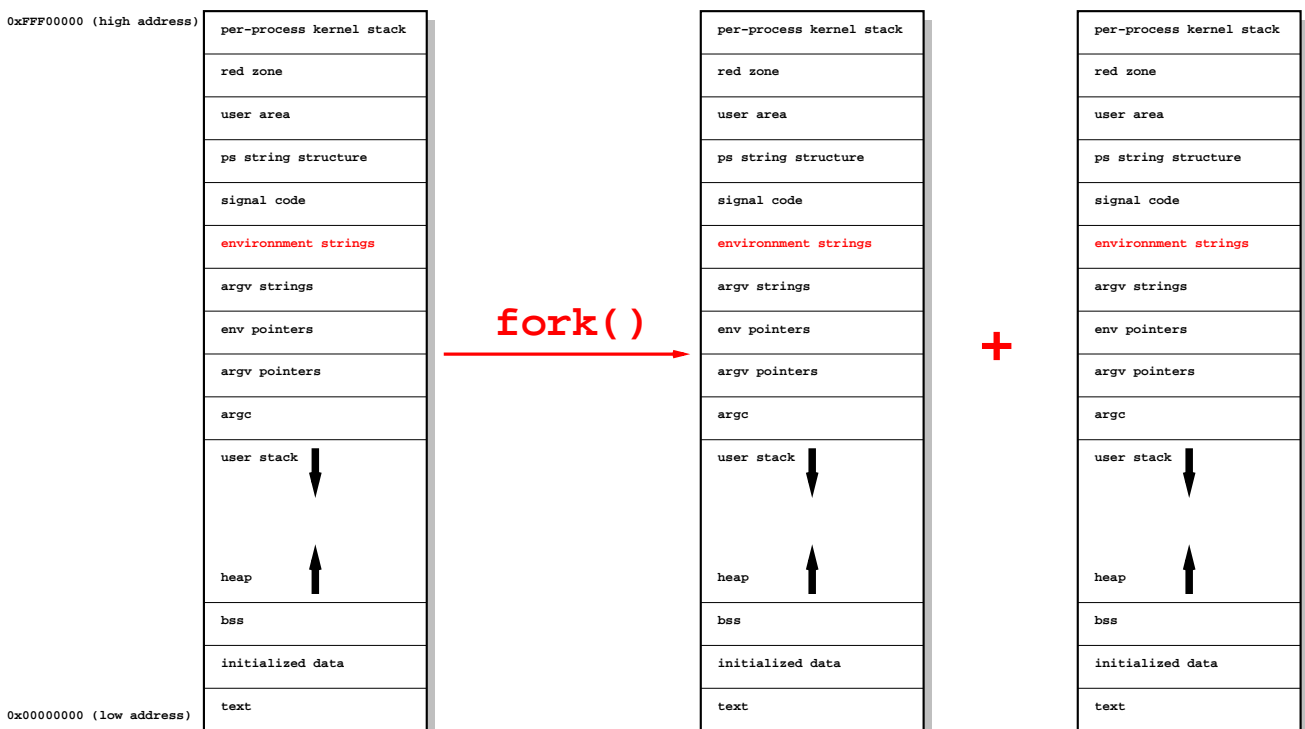
Vérification avec un shell :



Le fonctionnement réel est le suivant :



Retour sur le mécanisme du fork() utilisé lorsque le shell lance une commande :



Voici la liste des variables d'environnement standard :

Variable	Description
HOME	homedirectory
USER	username
SHELL	path du shell utilisé
PATH	liste des directories dans lesquels chercher des commandes
TERM	type du terminal utilisé

Les autres variables d'environnement ne sont pas standard.

## Chapitre 13 • Pratique du Bourne shell

### §13.22 • Variable d'environnement PATH

En Bourne Shell, la variable « PATH » stocke le chemin de commandes.

Chemin de commandes  $\equiv$  liste de répertoires séparés par des « : »

Parcours de tous les répertoires jusqu'à trouver la commande en question

- Hypothèse : le « PATH » vaut « /bin:/usr/bin:/usr/local/bin »
- Hypothèse 2 : l'utilisateur tape « ls »

- 1 Le shell cherche si l'exécutable « /bin/ls » existe. Si non étape suivante.
- 2 Le shell cherche si l'exécutable « /usr/bin/ls » existe. Si non étape suivante.
- 3 Le shell cherche si l'exécutable « /usr/local/bin/ls » existe. Si non étape suivante.
- 4 Si plus de répertoires de « PATH » à analyser, afficher l'erreur « command not found »

En mode interactif, pour ajouter le répertoire

« /chemin/vers/application/bin » à son « PATH » :

- en début de « PATH », faire :

```
% PATH=/chemin/vers/application/bin:$PATH
% export PATH
```

- en fin de « PATH », faire :

```
% PATH=$PATH:/chemin/vers/application/bin
% export PATH
```

Pour retirer un répertoire de son « PATH » :

- pas de méthode à part retaper tout :

```
% PATH=/repertoire1:/repertoire2:/repertoire3:...:/repertoireN
% export PATH
```

## On ne met jamais « . » dans son PATH ! DANGER!!!

« . » ≡ le répertoire courant

Dangers :

- Que contient le répertoire courant ?
- L'environnement peut être hostile.
- On peut faire des fautes de frappe, des coquilles.

Si besoin d'une commande dans le répertoire courant, l'appeler explicitement par « ./ » :

```
% ./commande-dans-répertoire-courant options paramètres...
```

## Chapitre 13 • Pratique du Bourne shell

### §13.23 • Régler son PATH de façon permanente

Hypothèse : on est sous Bourne Shell ou sous BASH.

Hypothèse 2 : on se rappelle le passage « Variable d'environnement PATH » vu dans le chapitre « Pratique du Bourne Shell ».

Objectif : on veut modifier son « PATH » de façon permanente.

Solution :

On utilise le fichier « `$HOME/.profile` » avec un lien symbolique « `$HOME/.bashrc` » dessus.

On règle ainsi (sh ou bash) :

- cas du shell interactif de login
- cas du shell interactif non de login
- cas du shell non interactif

On ajoute dans le fichier « `$HOME/.profile` »

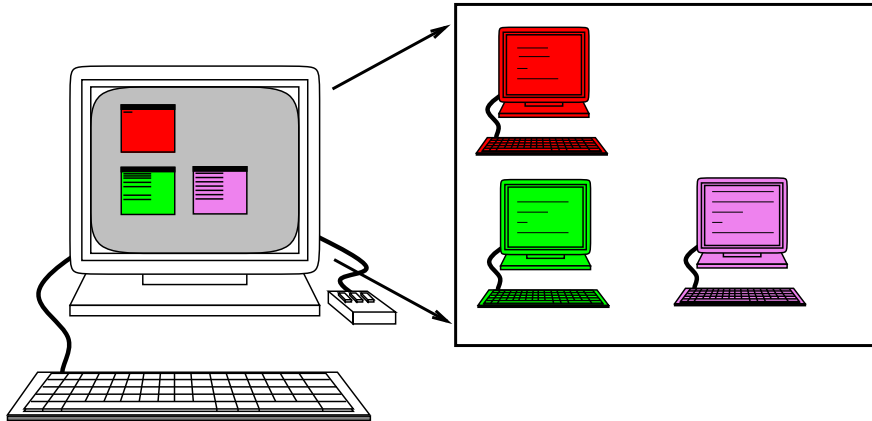
```
PATH=/repertoire1:/repertoire2:/repertoire3:...:/repertoireN
export PATH
```

ou

```
PATH=$PATH:/chemin/vers/application/bin
export PATH
```

## Chapitre 13 • Pratique du Bourne shell

Rappel :



Tous les programmes texte utilisent des « **escape sequences** » pour déplacer le curseur à l'écran.

⇒ nécessité de savoir comment faire

⇒ base de données des escape sequences ; l'entrée dans la base de données est donnée par la variable « TERM »

La variable d'environnement « TERM » définit le modèle de terminal texte utilisé.

Dans le passé, il y avait pléthore de console de terminaux texte.

⇒ base de données des modèles de terminaux :

- Implémentation système BSD : base de données au format TERMCAP stockée dans le fichier « /etc/termcap »
- Implémentation système System-V : base de données au format TERMINFO stockée dans l'arborescence `/usr/share/lib/terminfo/`  
Par exemple pour le minitel :


« /usr/share/lib/terminfo/m/minitel »

Base à jour disponible à <http://www.tuxedo.org/~esr/terminfo>

Utilitaires « tic » (terminfo compiler), « captainfo » pour convertir les fichiers de description de terminaux de termcap à terminfo

**Ne jamais changer la valeur de « TERM » mise automatiquement par le système sinon problèmes dans éditeurs, etc.**

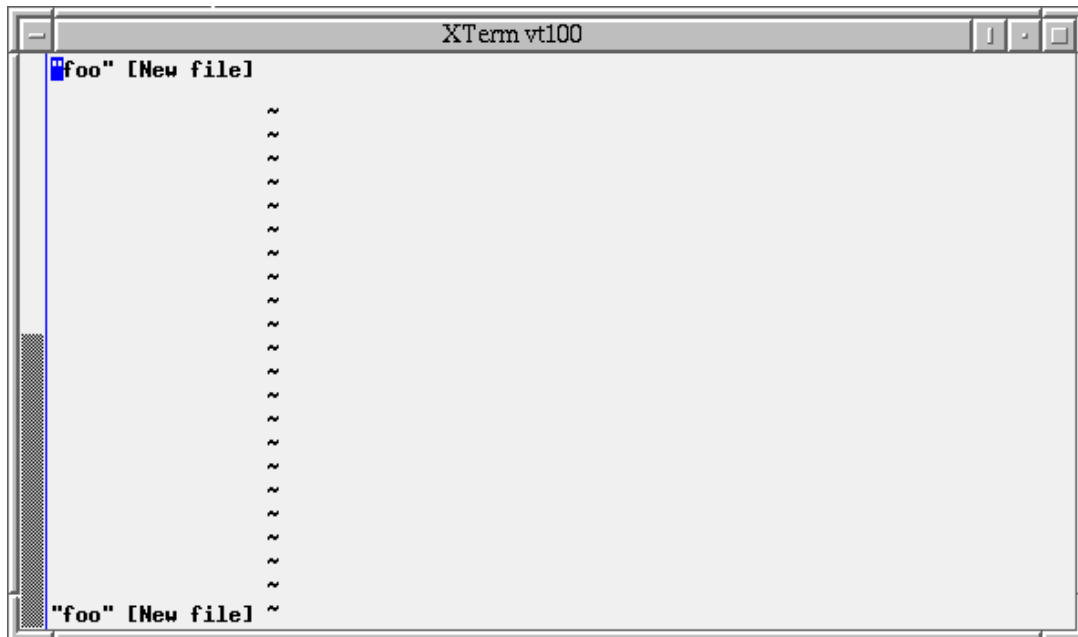
Jouons avec le feu en changeant la valeur mise par le système pour votre fenêtre :



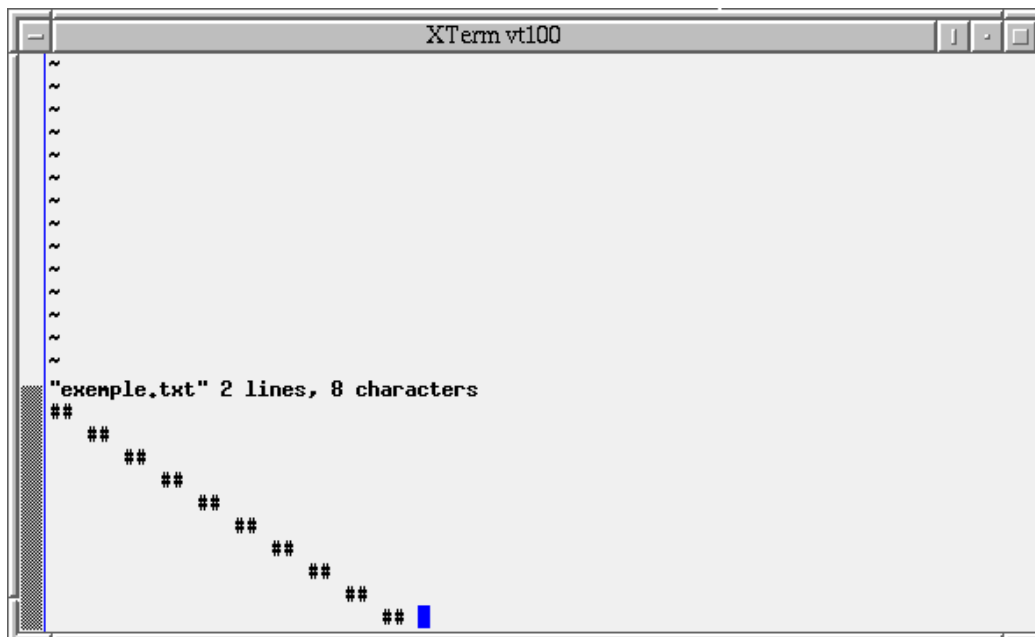
```
XTerm vt100
## echo $TERM
vt100
## TERM=sun
## export TERM
##
```



Sous l'éditeur « vi », les ennuis sont là :



Sous le shell, les ennuis sont là :



Solution : commande « reset » ; retour aux réglages par défaut du terminal si des escape séquences ont fichu la pagaille

La commande « `stty` » donne les caractéristiques bas niveau du terminal.

Dans les résultats renvoyés on trouve les configurations de certaines séquences de touches utilisées interactivement.

Par exemple l'effacement du caractère précédent de la ligne via la touche `Del` ou `Backspace`.

Il y a 2 versions de la commande selon la famille d'UNIX. L'affichage des données est différent mais on manipule la même chose au final.

Attention à ne pas confondre ce que permet la gestion du terminal texte et ce que permet le shell (par exemple « `bash` » offre des possibilités de déplacement du curseur au sein de la ligne de commande).

Principales noms de séquences utiles :

- séquence « `erase` » : effacement du caractère précédent
- séquence « `werase` » : effacement du mot précédent
- séquence « `kill` » : effacement de la ligne complète
- séquence « `intr` » : envoi du signal SIGINT
- séquence « `quit` » : envoi du signal SIGABRT
- séquence « `susp` » : envoi du signal SIGTSTP
- séquence « `eof` » : End Of File
- séquence « `start` » : relance le flux de l'affichage texte
- séquence « `stop` » : arrête le flux de l'affichage texte
- séquence « `lnext` » : permet la saisie de la séquence suivante sans l'interpréter

### ◇ Exemple de « stty » sur Solaris

```
% /usr/bin/stty -a
speed 9600 baud;
rows = 55; columns = 80; ypixels = 719; xpixels = 579;
csdata ?
eucw 1:0:0:0, scrw 1:0:0:0
intr = ^c; quit = ^\; erase = ^?; kill = ^u;
eof = ^d; eol = <undef>; eol2 = <undef>; swtch = <undef>;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
-parenb -parodd cs8 -cstopb hupcl cread -clocal -loblk -crtsets -crtsoff
-ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
ixon -ixany -ixoff imaxbel
isig icanon -xcase echo echoe echok -echonl -noflsh
-tostop echoctl -echoprt echoke -defecho -flusho -pendin iexten
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
```

### ◇ Modification de séquence via « stty »

Reprenons les séquences précédentes :

```
intr = ^c; quit = ^\; erase = ^?; kill = ^u;
eof = ^d; eol = <undef>; eol2 = <undef>; swtch = <undef>;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
```

Le shell standard sous Linux est « bash » qui permet de revenir en début de ligne tapée par « ^e ».

Pour changer « intr » de « ^c » en « ^e », on tapera donc en pratique :

```
% stty intr ^v^e
```

Le « ^v » neutralise l'effet de « ^e » (sous bash retour en début de ligne) lorsqu'on le tape.

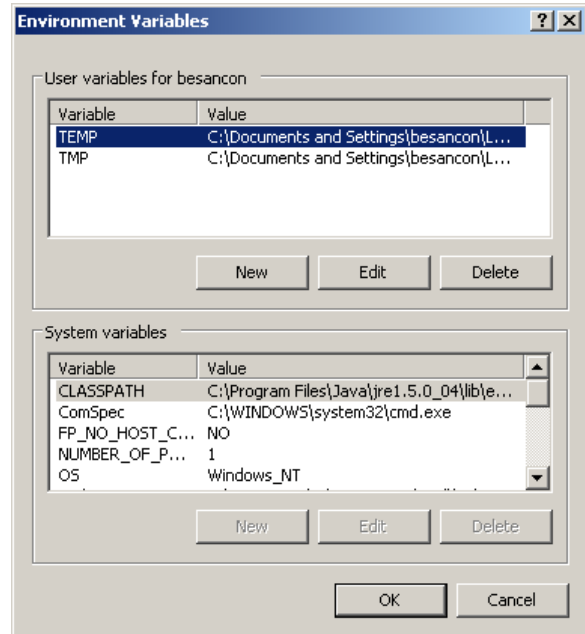
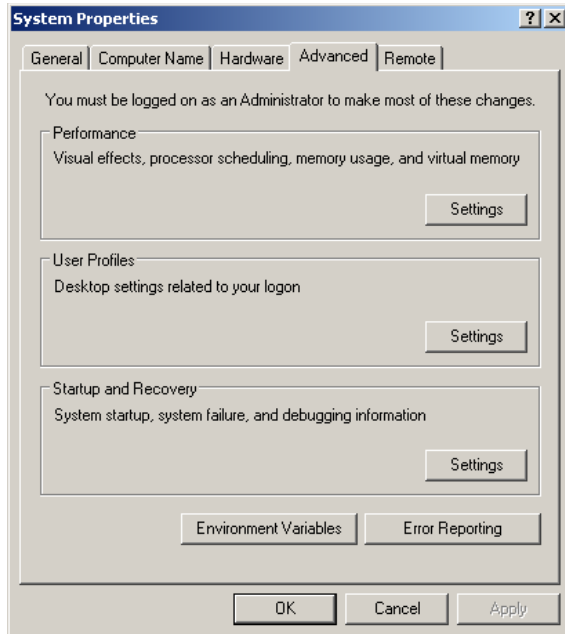
Le « ^v » est indiqué par la séquence « lnext » de « stty ».

## Chapitre 13 • Pratique du Bourne shell

§13.26 • (Windows : : Variables d'environnement)

Interface de base fournie par Windows :

Start &gt; Control Panel &gt; System &gt; Advanced &gt; Environment Variables



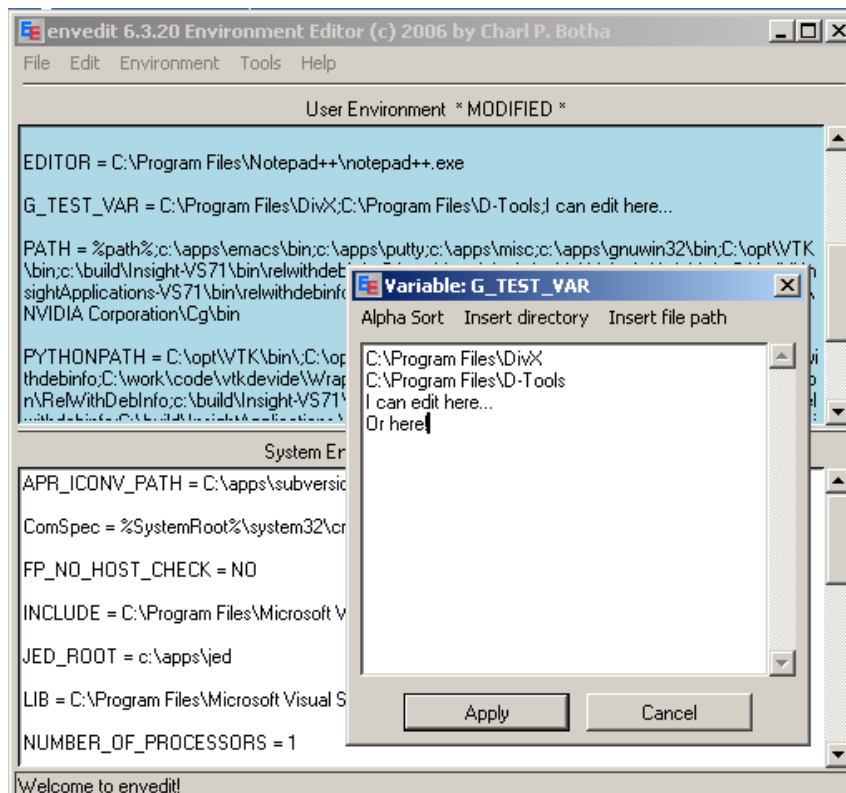
Interface peu pratique

©T.Besançon (version 11.0)

Administration Unix

ARS 2008–2009 Tôme 1

537 / 666

Programme plus ergonomique : **envedit** sur<http://cpbotha.net/Software/envedit>

©T.Besançon (version 11.0)

Administration Unix

ARS 2008–2009 Tôme 1

538 / 666

- 1 Redirection des entrées/sorties
- 2 Substitution des variables
- 3 Substitution des noms de fichiers

### ◇ Exemple 1 :

```
% pipe=\\  
% echo $pipe  
|
```

#### Explications :

- 1 pas de caractères de redirection des entrées/sorties ;  
la commande est « echo \$pipe »
- 2 remplacement de la variable par son contenu ;  
la commande est « echo | »
- 3 pas de caractères de substitution de noms de fichiers ;  
la commande est « echo | »

## ◇ Exemple 2 :

```
% star=\*
% echo $star
ananas banane cerise
```

## Explications :

- 1 pas de caractères de redirection des entrées/sorties ;  
la commande est « echo \$star »
- 2 remplacement de la variable par son contenu ;  
la commande est « echo \* »
- 3 remplacement du caractère « \* » par la liste des fichiers ;  
la commande est « echo ananas banane cerise »

## Chapitre 13 • Pratique du Bourne shell

## §13.28 • Se déconnecter du shell : exit, Ctrl-D

Pour se déconnecter : taper « exit »

Parfois, sous le shell BASH de LINUX on peut se déconnecter via « Ctrl-D » :

```
% ^D
<-- on est déconnecté
```

« Ctrl-D » est appelé **EOF** (*End Of File*)

Possibilité de désactiver la séquence « Ctrl-D » sous BASH via une variable spéciale interne de BASH :

```
% set -o ignoreeof
% ^DUse "exit" to leave the shell.
```

La commande « `login` » lance un **shell de login**.  
Le nom du shell est alors précédé d'un signe moins :

```
% ps -edf | grep bash
besancon 1773      1  0   Aug 24 console  0:00 -bash
besancon 1959    1949  0   Aug 24 pts/3     0:09 bash
besancon 8300    8299  0   Aug 26 pts/5     0:00 bash
```

Le shell sait alors qu'il est un shell de login (via « `argv[0]` »)

⇒ le shell exécute les fichiers de configuration de login

Un shell non de login n'exécutera pas les fichiers de configuration de login et sera initialisé plus simplement.

On peut forcer le mode shell de login :

- avec `bash` : « `bash --login` »
- avec `tcsh` : « `tcsh -l` »
- avec `xterm` : « `xterm -ls` » ou  
« `xterm -xrm '*loginShell: true'` »

Un shell est interactif si les commandes sont saisies sur un terminal.

Un shell est non interactif si les commandes sont lues dans un fichier.

Si le shell est non interactif, on n'exécutera pas les fichiers de commandes interactives et le shell sera initialisé plus simplement.

La commande « `tty -s` » permet de tester si l'on est en mode interactif ou pas :

- code de retour 0 : mode interactif
- code de retour 1 : mode non interactif

On peut forcer le mode shell interactif :

- avec `bash` : « `bash -i` »
- avec `sh` : « `sh -i` »
- avec `tcsh` : « `tcsh -i` »
- avec `csh` : « `csh -i` »

◇ Mode shell de login interactif

(sauf option contraire « `--noprofile` ») :

- 1 on exécute « `/etc/profile` »
- 2 on exécute le premier et seulement le premier fichier existant parmi  
« `$HOME/.bash_profile` », « `$HOME/.bash_login` »,  
« `$HOME/.profile` »

◇ Mode shell non interactif mais avec l'option « `-login` »

(sauf option contraire « `--noprofile` ») :

- 1 on exécute « `/etc/profile` »
- 2 on exécute le premier et seulement le premier fichier existant parmi  
« `$HOME/.bash_profile` », « `$HOME/.bash_login` »,  
« `$HOME/.profile` »



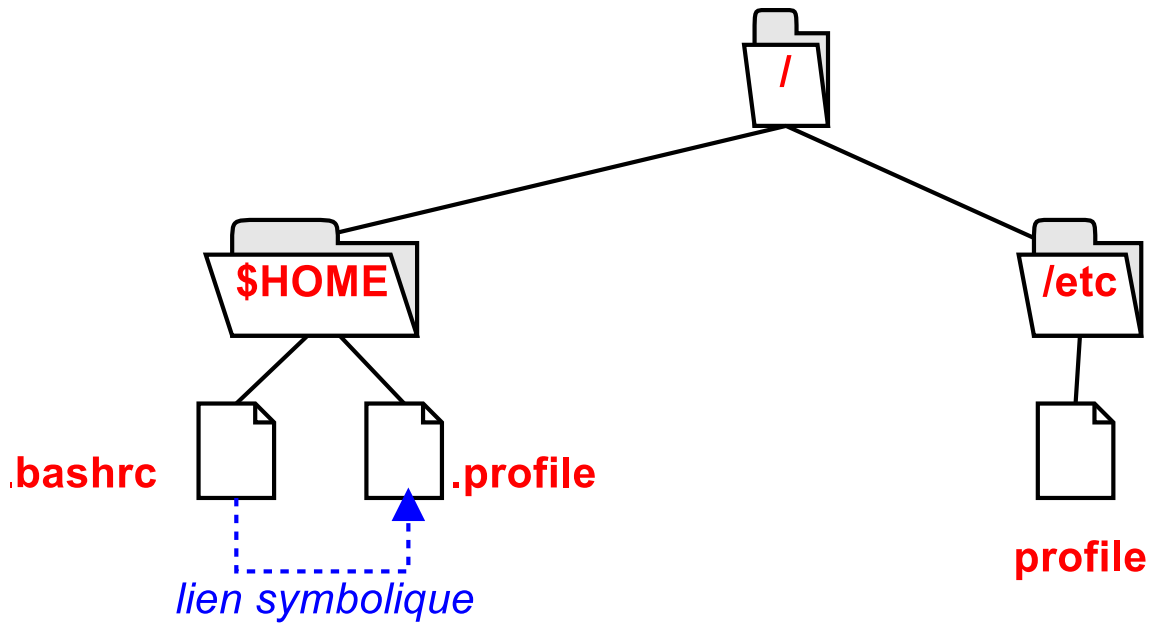
◇ Mode shell non de login interactif :

**1** on exécute « `$HOME/.bashrc` » (sauf option contraire « `--norc` »)

◇ Mode shell non de login non interactif :

**1** on exécute le fichier mentionné par « `$BASH_ENV` »

Conseil pour traiter simplement les trois premiers cas :



◇ Quand un shell de login se termine :

**1** on exécute « \$HOME/.bash\_logout »

◇ Travailler sous bash avec des caractères accentués (éèà...) :  
 ajouter au fichier « \$HOME/.inputrc » les lignes suivantes :

```
set meta-flag on
set convert-meta off
set output-meta on
```

## Chapitre 13 • Pratique du Bourne shell

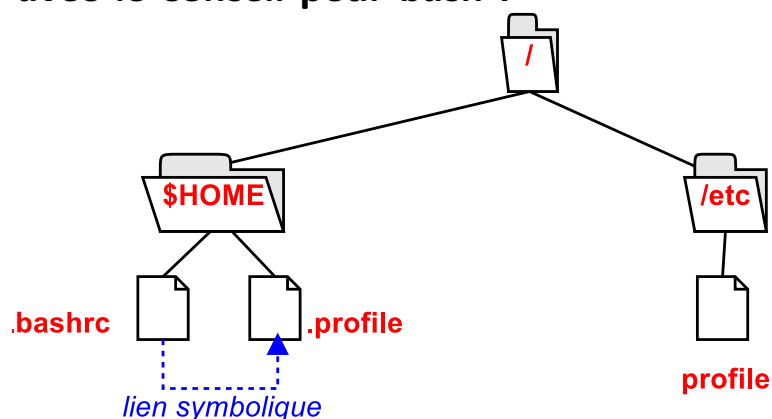
### §13.32 • Fichiers d'initialisation pour sh

En mode shell de login :

- 1 on exécute « /etc/profile »
- 2 on exécute « \$HOME/.profile »

En mode autre : on n'exécute aucun fichier !

**Compatibilité avec le conseil pour bash :**



En mode shell de login :

- 1 on exécute « /etc/csh.cshrc »
- 2 on exécute « /etc/csh.login »
- 3 on exécute « \$HOME/.tcshrc » (ou « \$HOME/.cshrc » à la place si non existant)
- 4 on exécute « \$HOME/.history »
- 5 on exécute « \$HOME/.login »
- 6 on exécute « \$HOME/.cshdirs »

En mode shell non de login ou en mode non interactif :

- 1 on exécute « /etc/csh.cshrc »
- 2 on exécute « \$HOME/.tcshrc » (ou « \$HOME/.cshrc » à la place si non existant)

Quand un shell de login se termine :

- 1 on exécute « /etc/csh.logout »
- 2 on exécute « \$HOME/.logout »

## Chapitre 13 • Pratique du Bourne shell

## §13.34 • Fichiers d'initialisation pour csh

En mode shell de login :

- 1 on exécute « /etc/.login »
- 2 on exécute « \$HOME/.cshrc »
- 3 on exécute « \$HOME/.login »
- 4 on exécute « \$HOME/.cshdirs »

Quand un shell de login se termine :

- 1 on exécute « \$HOME/.logout »

## Chapitre 13 • Pratique du Bourne shell

## §13.35 • Complétion interactive

Sur LINUX, le shell s'appelle BASH = version améliorée du Bourne Shell

Une des fonctionnalités interactives les plus intéressantes : l'expansion des noms de fichier via la touche TAB : on parle de **complétion** :

```
% ls
abricot.txt      asperge.txt      choux.txt        poire.txt
ananas.txt       banane.txt       fraise.txt       poireau.txt
artichaut.txt   cerise.txt       patate.txt       pomme.txt
% ls pTAB
patate.txt      poire.txt        poireau.txt     pomme.txt
% ls poTAB
poire.txt       poireau.txt     pomme.txt
% ls pomTAB
% ls pomme.txt
```

## Chapitre 13 • Pratique du Bourne shell

§13.36 • (Windows : : Complétion interactive)

La complétion est aussi disponible sous WINDOWS dans un « cmd.exe ».

Pour Windows NT et Windows 2000 : réglage dans la base de registres : affecter la valeur 9 à la clef

```
HKEY_CURRENT_USER/Software/Microsoft/Command Processor/  
CompletionChar
```

Pour Windows XP : fonctionnalité installée de base

## Chapitre 14

# *Programmation en Bourne shell*

Le shell propose un langage de programmation interprété.

Son utilité :

- automatisation d'actions
- utilisation de structures plus avancées :
  - boucles
  - tests
  - ...
- scripts d'installation de logiciels à adapter

On appelle « shell script » un programme écrit dans la syntaxe d'un shell et s'appuyant sur les commandes UNIX.

Caractéristiques d'un shell script :

- C'est un programme écrit en langage shell.
- Il est écrit pour un shell particulier, à la syntaxe bien particulière. Un shell script ne peut pas être exécuté par un autre shell en général.
- Il est exécutable.  
⇒ faire « `chmod a+x exemple.sh` »

Structure d'un shell script :

### ■ Désignation du shell utilisé

La première ligne du shell script commence par « #! » suivi du path du shell utilisé et de ses arguments éventuels.

### ■ Commentaires

Un commentaire est introduit par le caractère « # » et se poursuit jusqu'à la fin de la ligne.

Un commentaire peut être placé n'importe où.

La première ligne du script est un commentaire très particulier.

### ■ Code

Traditionnelles lignes de code respectant la syntaxe du shell utilisé.

Exemple :

```
#!/bin/sh
#
# Script d'exemple. Il affiche simplement la date.
#
date
```

## ATTENTION!!!

Ceci est faux (pas uniquement à cause de l'orthographe) :

```
#####
#
# ARS 1999/2000
# AUTEUR : BERGOUGNOUX YVES
# DATE DE CREATION : 07/01/2000
# DATE DE MODIFICATION : 07/01/2000
# THEME : enrgitre l'expiration d'un compte de stagiaire ayant les memes
# droit que sont parain pour géré la fin du compte dans le fichier cpt_sta#
# NOM DE LA COMMANDE : hda5/home/yves/projet/set-deadline (disque UNIX)
#
#####

#!/bin/sh

...
```

Pourquoi ?

A cause de la position de la ligne « #!/bin/sh »



**En l'absence d'indication de l'interpréteur de commandes en première ligne du script, le script est exécuté par le shell courant de la session de l'utilisateur.**

**Attention aux systèmes comme Linux où le shell par défaut est compatible avec le Bourne shell, masquant ainsi l'erreur!!!**

Preuve :

Soit le script « erreur.sh » suivant :

```
# Je suis un commentaire qui n'a rien a faire ici
#!/bin/sh
for i in *
do
    echo $i
done
```

Si l'on exécute le script précédent, on obtient selon le shell de la session :

```
% echo $SHELL
/bin/csh
```

```
% ./erreur.sh
for: Command not found.
do: Command not found.
i: Undefined variable.
```

```
% echo $SHELL
/bin/bash
```

```
% ./erreur.sh
a
erreur.sh
repertoire1
```

Moralité :

**LA PREMIERE LIGNE DU SCRIPT DOIT ETRE  
CELLE EN « #!/bin/sh »**

## Chapitre 14 • Programmation en Bourne shell

### §14.4 • Code de retour d'un shell script : `exit`

La commande « **exit** » renvoie une valeur de retour pour le shell script et provoque l'arrêt de l'exécution du script.

La valeur de retour est un entier compris entre 0 et 255.

Le code de retour d'un shell script suit la même convention que pour les commandes UNIX :

- code de retour nul  
le script s'est exécuté correctement
- code de retour non nul  
le script a rencontré une condition logique d'erreur

Exemple de script :

```
#!/bin/sh
exit 0
```

## Chapitre 14 • Programmation en Bourne shell

### §14.5 • Passage de paramètres à un shell script : `$1` à `$9`

Comme tout programme, on peut passer des paramètres à un shell script.

Variable	Description
<code>\$0</code>	Nom du shell script
<code>\$1</code> à <code>\$9</code>	Les 9 premiers paramètres
<code>\$#</code>	Le nombre de paramètres
<code>\$*</code>	Tous les paramètres passés au shell script sous la forme de mots individuels séparés
<code>\$@</code>	Tous les paramètres passés au shell script

Exemple : soit le script `exemple.sh` suivant :

```
#!/bin/sh
echo "Parametre 1 : $1"
echo "Parametre 2 : $2"
```

Son exécution donne :

```
% ./exemple.sh AAAAA BBBB
Parametre 1 : AAAAA
Parametre 2 : BBBB
```

Comment accéder à tous les paramètres ?

Soit le script `erreur.sh` suivant :

```
#!/bin/sh
echo $0 $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12
```

Son exécution donne :

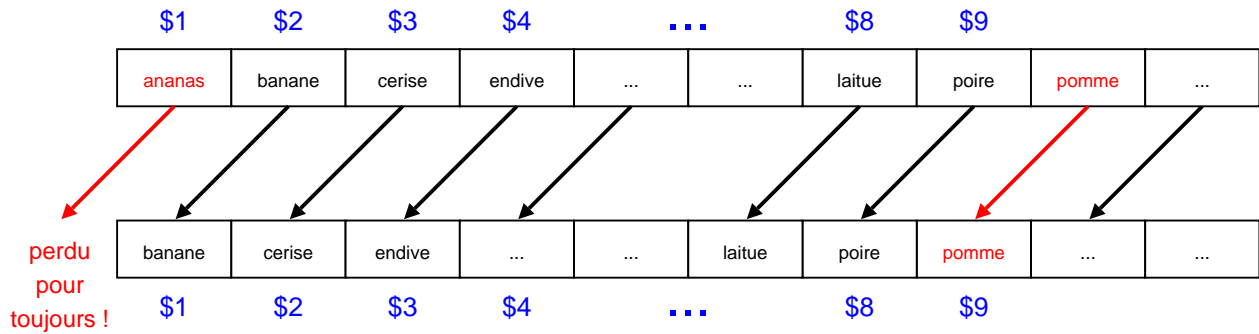
```
% ./erreur.sh a b c d e f g h i j k l
./erreur.sh a b c d e f g h i a0 a1 a2
```

Pourquoi ces résultats ?

Conclusion ⇒ comment faire ?

La solution consiste à utiliser la commande « **shift** ».

(en anglais *shift*, décalage)



Soit le script `exemple.sh` suivant : Son exécution donne :

```
#!/bin/sh
echo $1 $2 $3
shift
echo $1 $2 $3
```

```
% ./exemple.sh a b c d
a b c
b c d
```

Attention !

A chaque emploi de `shift`, le paramètre `$1` précédent est perdu. Du même coup, ce paramètre est supprimé de `$*` et `$@`, `$#` est décrémenté de 1.

```
#!/bin/sh
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
shift
echo "$# ; $1 $2 $3 ; $@"
```

```
% ./exemple.sh a b c d
4 ; a b c ; a b c d
3 ; b c d ; b c d
2 ; c d ; c d
1 ; d ; d
0 ; ;
```

## Attention !

L'emploi de `shift` nécessite que le shell script ait au moins un paramètre :

```
#!/bin/sh
echo $1 $2 $3
shift
echo $1 $2 $3

% ./exemple.sh <-- pas de paramètres

shift: can't shift that many
```

## Chapitre 14 • Programmation en Bourne shell

### §14.6 • Liste des paramètres d'un shell script : \$\*, @\$

Deux syntaxes pour la liste des paramètres :

- « \$\* » : Tous les paramètres passés au shell script sous la forme de mots individuels séparés
- « @\$ » : Tous les paramètres passés au shell script

Soit un shell script que l'on appelle ainsi :

```
% ./exemple.sh "ananas" "deux mots" "cerise"
```

Alors :

« \$\* » est une liste de **4** éléments :

- 1 "ananas"
- 2 "deux"
- 3 "mots"
- 4 "cerise"

« @\$ » est une liste de **3** éléments :

- 1 "ananas"
- 2 "deux mots"
- 3 "cerise"

La variable « \$? » contient le code de retour de la dernière commande exécutée.

On ne peut que consulter cette variable.

Soit le script « `exemple.sh` » suivant avec ses commentaires explicatifs :

```
#!/bin/sh

# Hypothèse : le fichier fichier1 existe
ls -l fichier1
echo $?

# Hypothèse : le fichier fichier2 n'existe pas
ls -l fichier2
echo $?
```

Son exécution donne :

```
% ./exemple.sh
-rw-r--r--  1 besancon ars          0 Jul  9 00:27 fichier1
0
fichier2: No such file or directory
2
```

La variable « \$\$ » contient le PID du shell script qui est en train de s'exécuter.

On ne peut que consulter cette variable.

Soit le script « `exemple.sh` » suivant :

```
#!/bin/sh
sleep 30
echo Je suis le processus $$
```

Son exécution donne :

```
% ./exemple.sh
Je suis le processus 1406
```

On peut en parallèle (grâce aux 30 secondes du `sleep`) chercher par la commande « `ps` » le numéro de processus :

```
% ps -edf | grep exemple.sh
besancon 1406 28602 0 00:31:02 pts/1 0:00 /bin/sh ./exemple.sh
```

Utilisation classique : création de fichiers temporaires uniques associés au script

```
#!/bin/sh
temporaire=/tmp/exemple.$$
touch $temporaire
ls -l $temporaire
```

Son exécution donne

```
% ./exemple.sh
-rw-r--r--  1 besancon ars      0 Jul  9 00:34 /tmp/exemple.1416
```

Une autre exécution donne

```
% ./exemple.sh
-rw-r--r--  1 besancon ars      0 Jul  9 00:35 /tmp/exemple.1419
```

## Chapitre 14 • Programmation en Bourne shell

### §14.9 • Commandes internes du shell : builtins

(en anglais *built in*)

Le shell dispose de commandes internes (*builtins*) : « cd », « set », etc.

Si une commande est builtin, elle est programmée dans le code C du shell.

Si une commande n'est pas builtin, elle correspond à un exécutable dans l'arborescence du système.



**Attention, passage difficile :**

La commande « `cd` » est un builtin dans tous les shells.

Soit le script « `exemple.sh` » suivant :

```
#!/bin/sh
cd /
```

On se trouve initialement dans « `/tmp` ».

On lance le shell script.

Où se retrouve-t-on ?

**Chapitre 14 • Programmation en Bourne shell****§14.10 • Commandes internes du shell : `type`**

Une méthode pour identifier les builtins est d'utiliser la commande du Bourne Shell « **type** » :

```
% type cd
cd is a shell builtin
```

```
% type echo
echo is a shell builtin
```

```
% type ls
ls is /bin/ls
```

## Chapitre 14 • Programmation en Bourne shell

### §14.11 • Commande d'affichage : builtin echo, /bin/echo

La commande d'affichage de caractères est « **echo** ».

Attention !

La commande « `echo` » peut ne pas être un builtin du shell.  
En Bourne Shell, c'est toujours un builtin.

Il existe une commande UNIX `/bin/echo` :

```
% ls -l /bin/echo
-r-xr-xr-x 1 bin bin 32768 May 20 12:30 /bin/echo
```

**Le comportement du builtin peut être différent de celui de la commande UNIX.**

#### ◇ Première syntaxe possible pour `echo`

La commande `echo` comprend des séquences semblables à celles de « `printf()` » du langage C (commande `echo` d'inspiration System-V) :

Séquence	Description
<code>\b</code>	Backspace
<code>\c</code>	Pas de newline envoyé
<code>\n</code>	Newline
<code>\r</code>	Carriage return
<code>\t</code>	Tabulation horizontale
<code>\v</code>	Tabulation verticale
<code>\\</code>	Backslash
<code>\nnn</code>	Caractère dont le code octal ASCII est donné

C'est le cas du « `echo` » du shell « `sh` » sur SUN SOLARIS.

Exemple de cette syntaxe :

Soit le script « `exemple.sh` » suivant :

```
#!/bin/sh
a=2
b=3
# Mauvais affichage
echo "a*b="
expr $a \* $b
# Affichage correct
echo "a*b=\c"
expr $a \* $b
```

Son exécution donne :

```
% ./exemple.sh
a*b=
6
a*b=6
```

### ◇ Seconde syntaxe possible pour echo

La commande « `echo` » comprend des options (commande `echo` d'inspiration BSD) :

- option « `-n` » : Pas de newline envoyé

C'est le cas de « `echo` » du shell « `bash` » sur LINUX.

Exemple de cette syntaxe :

```
#!/bin/sh
a=2
b=3
# Mauvais affichage
echo "a*b="
expr $a \* $b
# Affichage correct
echo -n "a*b="
expr $a \* $b
```

Son exécution donne :

```
% ./exemple.sh
a*b=
6
a*b=6
```

Exemple : qu'arrive-t-il si l'on se trompe de syntaxe ?  
(ici syntaxe BSD avec un echo de syntaxe System-V)

```
#!/bin/sh
a=2
b=3
# Mauvais affichage
/bin/echo "a*b="
expr $a \* $b
# Affichage correct
/bin/echo -n "a*b="
expr $a \* $b
```

Son exécution donne :

```
% ./exemple.sh
a*b=
6
-n a*b=
6
```

La commande « **read** » permet de lire au clavier et de placer les mots lus dans une liste de variables.

Syntaxe :

```
read variable-list
```

Le premier mot va dans la première variable, le deuxième mot va dans la deuxième variable... Tous les mots en trop sont stockés dans la dernière variable mentionnée.

### ◇ Exemple 1

Soit le script « `exemple.sh` » suivant :

```
#!/bin/sh
echo "Entrez quelque chose : \c"
read reponse
echo "Vous avez entré : $reponse"
```

Son exécution donne :

```
% ./exemple.sh
Entrez quelque chose : il etait une fois
Vous avez entré : il etait une fois
```

On notera au passage l'utilisation de « `\c` » dans la première ligne « `echo` » pour coller la réponse entrée au texte de la question.

## ◇ Exemple 2

Soit le script « `exemple.sh` » suivant :

```
#!/bin/sh
read variable1 variable2
echo "Premiere variable : $variable1"
echo "Seconde variable : $variable2"
```

Son exécution donne :

```
% ./exemple.sh
Unix MS-DOS Windows 95 Windows NT MacOS
Premiere variable : Unix
Seconde variable : MS-DOS Windows 95 Windows NT MacOS
```

## Chapitre 14 • Programmation en Bourne shell

### §14.13 • Structure if - then - else

Syntaxes :

```
1 if condition-est-vraie
   then
       bloc-de-commandes-unix
   fi
```

```
2 if condition-est-vraie
   then
       bloc-1-de-commandes-unix
   else
       bloc-2-de-commandes-unix
   fi
```

La condition (booléenne) est en général le code de retour d'une commande UNIX. Le code de retour de la commande détermine le test « `if` » :

- Code de retour valant zéro :  
Le test « `if` » est vrai.
- Code de retour non nul :  
Le test « `if` » est faux.

## Conseils :

- Ne pas utiliser une autre forme possible :

```
if condition-1-est-vraie
then
    bloc-1-de-commandes-unix
elif condition-2-est-vraie
    bloc-2-de-commandes-unix
fi
```

car rapidement illisible à mon goût

- Indenter les blocs pour être lisible

## Exemple :

```
#!/bin/sh
if ls > /dev/null
then
    echo Il y a des fichiers
fi
```

Forme générique d'un « if » dans un shell script :

```
#!/bin/sh
if commande [options] parametres > resultats.txt 2> erreurs.txt
then
    # code de retour (exit) valant 0
    bloc-1-de-commandes-unix
else
    # code de retour (exit) différent de 0
    bloc-2-de-commandes-unix
fi
```

## Chapitre 14 • Programmation en Bourne shell

### §14.14 • Structure case

La commande « case » permet de tester une chaîne de caractères par rapport à un certain nombre d'autres chaînes prédéfinies :

```
case chaine-a-tester in
    possibilite1) bloc-1-de-commandes-unix
                ;;
    possibilite2) bloc-2-de-commandes-unix
                ;;
    ...
    possibiliteN) bloc-N-de-commandes-unix
                ;;
esac
```



### ◇ Exemple 1 : forme simple

Les possibilités sont de simples chaînes de caractères statiques :

```
#!/bin/sh
echo -n "Donnez un chiffre entre 1 et 3 -->"
read reponse
case "$reponse" in
  "1") echo "Vous avez entré le chiffre 1"
        ;;
  "2") echo "Vous avez entré le chiffre 2"
        ;;
  "3") echo "Vous avez entré le chiffre 3"
        ;;
  * ) echo "Erreur"
        exit 1
        ;;
esac
exit 0
```

### ◇ Exemple 2 : forme sophistiquée

Les possibilités peuvent être construites à partir de méta-caractères :

```
#!/bin/sh
echo -n "Donnez un chiffre entre 1 et 5 -->"
read reponse
case "$reponse" in
  [1-5]) echo "Le chiffre est bien entre 1 et 5. Merci."
          ;;
  *) echo "Erreur"
        exit 1
        ;;
esac
exit 0
```

### ◇ Exemple 3 : forme sophistiquée

Les possibilités peuvent être construites à partir de méta-caractères :

```
#!/bin/sh
echo -n "Entrer le mot unix ou windows ou macintosh --> "
read reponse
case "$reponse" in
    unix | windows) echo "Gagné !"
                    ;;
    macintosh) echo "Perdu !"
               ;;
    *) echo "Réponse non autorisée !"
       exit 1
       ;;
esac
exit 0
```

### ◇ Exemple 4 : forme sophistiquée

Les possibilités peuvent être construites à partir de méta-caractères :

```
#!/bin/sh
echo -n "Entrer un mot français -->"
read reponse
case "$reponse" in
    [aeiouy]*) echo "Le mot commence par une voyelle."
               ;;
    [0-9]*) echo "Le mot commence par un chiffre"
            ;;
    *) echo "Le mot commence par autre chose."
       ;;
esac
exit 0
```

Dans de nombreuses structures shell, on teste une condition. La commande « **test** » permet de réaliser divers tests.

Liste partielle (lisez la page de manuel de `test` pour compléter) :

Format	Description
« -d objet »	Vrai si l'objet existe et est un répertoire.
« -f objet »	Vrai si l'objet existe et est un fichier.
« -s objet »	Vrai si l'objet existe, est un fichier et a une taille supérieure à zéro.
« -w objet »	Vrai si l'objet existe et que l'on peut écrire dans l'objet.
« -x objet »	Vrai si l'objet existe et que l'on peut l'exécuter

Format	Description
« -n string »	Vrai si la chaîne est non vide.
« s1 = s2 »	Vrai si les chaînes s1 et s2 sont identiques.
« s1 != s2 »	Vrai si les chaînes s1 et s2 ne sont pas identiques.
« n1 -eq n2 »	Vrai si les chaînes n1 et n2 sont mathématiquement égales (anglais <i>equal</i> ).
« n1 -ne n2 »	Vrais si les chaînes n1 et n2 ne sont pas mathématiquement égales (anglais <i>not equal</i> ).
« n1 -gt n2 »	Vrai si la chaîne n1 est mathématiquement strictement supérieure à n2 (anglais <i>greater than</i> ).
« n1 -ge n2 »	Vrai si la chaîne n1 est mathématiquement supérieure ou égale à n2 (anglais <i>greater or equal</i> ).
« n1 -lt n2 »	Vrai si la chaîne n1 est mathématiquement strictement inférieure à n2 (anglais <i>less than</i> ).
« n1 -le n2 »	Vrai si la chaîne n1 est mathématiquement inférieure ou égale à n2 (anglais <i>less or equal</i> ).

Format	Description
« ! expression »	Vrai si l'expression est fausse.
« expression1 -a expression2 »	Vrai si expression1 et expression2 sont vraies.
« expression1 -o expression2 »	Vrai si expression1 ou expression2 est vraie.

**Attention à ne pas baptiser du nom « test » un de vos scripts à vous. La commande « test » du système pourrait avoir priorité sur votre script selon le PATH si bien que votre script ne tournerait jamais.**

⇒ **Baptisez vos scripts de test (et aussi exécutables) de noms comme « essai », etc.**

Double forme de la commande test :

**1** forme normale :

```
#!/bin/sh
if test "$1" = hello
then
    echo hello world
fi
```

**2** forme crochet :

```
#!/bin/sh
if [ "$1" = hello ]
then
    echo hello world
fi
```

Vérification :

```
% ls -li /bin/[ /bin/test
15422 -r-xr-xr-x  2 bin  bin  45056 May 20 12:31 /bin/[
15422 -r-xr-xr-x  2 bin  bin  45056 May 20 12:31 /bin/test
```

Par contre pas de commande UNIX « ] », bien sûr.

Exemple 1 :

```
#!/bin/sh
if test "$1" = hello
then
    echo hello world
fi
```

Exemple 2 :

```
if [ $1 -gt $2 -o $1 -eq $2 ]
then
    echo $1 is greater than or equal to $2
fi
```

Exemple 3 :

```
#!/bin/sh
[ $# -eq 0 ] && echo You entered no parameters
```

## Chapitre 14 • Programmation en Bourne shell

## §14.16 • Structure de boucles : while, for, until

Syntaxes :

<pre>while condition do   bloc-cmdes-unix done</pre>	<pre>for variable in list do   bloc-cmdes-unix done</pre>	<pre>until condition do   bloc-cmdes-unix done</pre>
--	---	--

Ces 3 formes sont équivalentes.

**En pratique, on choisira la forme pour laquelle la condition s'exprime le plus facilement ou le plus naturellement.**

### ◇ Exemple 1a

<pre>#!/bin/sh while [ "\$1" ] do   echo \$1   shift done</pre>	<pre>% ./exemple.sh a b "deux mots" d e a b deux mots d e</pre>
---	---

### ◇ Exemple 1b

```
#!/bin/sh                                     % ./exemple.sh
compteur=5                                     5
while [ $compteur -ge 0 ]                     4
do                                              3
    echo $compteur                             2
    compteur=`expr $compteur - 1`             1
done                                           0
```

### ◇ Exemple 2a

```
#!/bin/sh                                     % ./exemple.sh a b "deux mots" d e
echo $#                                       5
for i in "$@"                                a
do                                            b
    echo $i                                   deux mots
done                                          d
                                           e
```

◇ Exemple 2b

```
#!/bin/sh
compteur=0
for i in "$@"
do
    compteur=`expr $compteur + 1`
    echo "argv[$compteur]=$i"
done
```

```
% ./exemple.sh a b "deux mots" d e
argv[1]=a
argv[2]=b
argv[3]=deux mots
argv[4]=d
argv[5]=e
```

◇ Exemple 3a

```
#!/bin/sh
until [ "$1" = "" ]
do
    echo $1
    shift
done
```

```
% ./exemple.sh a b "deux mots" d e
a
b
deux mots
d
e
```



### ◇ Exemple 3b

```
#!/bin/sh                                     % ./exemple.sh
compteur=5                                    5
until [ $compteur -lt 0 ]                    4
do                                            3
    echo $compteur                            2
    compteur=`expr $compteur - 1`            1
done                                          0
```

Technique classique de boucle for : utilisation d'un fichier manifeste  
Habituellement :

```
#!/bin/sh
for img in *.jpg
do
    djpeg -scale 1/4 $img | cjpeg -progressive > small/$img
done
```

Si la liste des images est dans le fichier `liste` alors :

```
#!/bin/sh
for img in `cat liste`
do
    djpeg -scale 1/4 $img | cjpeg -progressive > small/$img
done
```

## Chapitre 14 • Programmation en Bourne shell

## §14.17 • Contrôle du flux d'exécution : break, continue

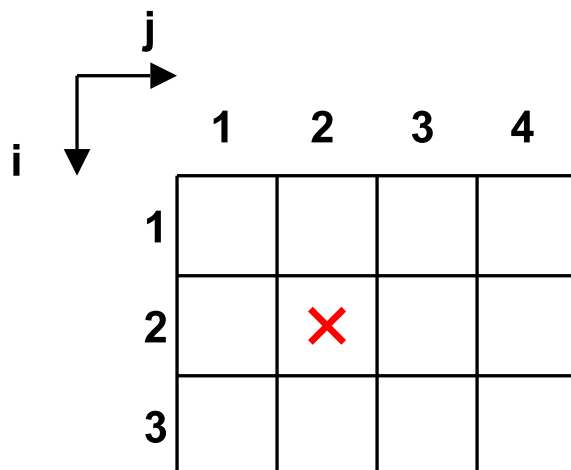
Deux commandes du shell permettent de contrôler le flux d'exécution du code :

- « **break** » ou « **break n** »

Cela permet de terminer prématurément la boucle courante ou N niveaux de boucles imbriquées.

- « **continue** » ou « **continue n** »

Cela permet de finir prématurément le tour de la boucle courante ou ce tour dans N boucles imbriquées.



```
#!/bin/sh
for i in 1 2 3
do
  for j in 1 2 3 4
  do
    if [ $i -eq 2 -a $j -eq 2 ]
    then
      echo ""
      echo "Bingo : $i,$j"
      break 2
    else
      echo "[$i,$j]\c"
    fi
  done
done
echo ""
```

```
% ./exemple
[1,1][1,2][1,3][1,4][2,1]
Bingo : 2,2
%
```

	j			
i	1	2	3	4
1		X		
2		X		
3		X		
4		X		

```
#!/bin/sh
for i in 1 2 3 4
do
  for j in 1 2 3 4
  do
    if [ $j -eq 2 ]
    then
      echo ""
      echo "Bingo : $i,$j"
      continue 2
    else
      echo "[ $i,$j ]\c"
    fi
  done
done
echo ""

% ./exemple
[1,1]
Bingo : 1,2
[2,1]
Bingo : 2,2
[3,1]
Bingo : 3,2
[4,1]
Bingo : 4,2

%
```

## Chapitre 14 • Programmation en Bourne shell

### §14.18 • Debugging d'un shell script : set -x

2 méthodes :

- 1 *The most effective debugging tool is still careful thought, coupled with judiciously placed print statements.* (Brian Kernighan [1978])
- 2 Placer en début de script la ligne

```
set -x
```

Exemple :

```
#!/bin/sh
set -x
echo $USER
```

ce qui donne à l'exécution :

```
% ./exemple.sh
+ echo besancon
besancon
```

La commande « `script` » sert à enregistrer le texte d'une session shell.

Syntaxe : `script [fichier]`

A noter :

- S'il n'y a pas de paramètre, alors le texte est enregistré dans le fichier appelé « `typescript` ».
- On trouve le caractère « `\r` » (c'est-à-dire Ctrl-M) en fin de chaque ligne de la session.
- En première ligne se trouve la date du début d'enregistrement du texte.
- En dernière ligne se trouve la date de la fin d'enregistrement du texte.

### ◇ Exemple 1 :

```
% script
Script started, file is typescript
sh-2.05$ ls
exemple.txt  typescript
sh-2.05$ Script done, file is typescript

% cat typescript
Script started on Sun Oct 03 22:42:59 2004
sh-2.05$ ls
exemple.txt  typescript
sh-2.05$
script done on Sun Oct 03 22:43:03 2004
```

## ◇ Exemple 2 :

```
% script session.txt
Script started, file is session.txt
sh-2.05$ date
Sun Oct  3 22:50:57 MEST 2004
sh-2.05$ Script done, file is session.txt

% od -c session.txt
0000000  S   c   r   i   p   t           s   t   a   r   t   e   d
0000020  n           S   u   n           O   c   t           0   3           2
0000040  5   0   :   5   3           2   0   0   4  \n  s   h   -
0000060  0   5   $           d   a   t   e  \r  \n  S   u   n
0000100  t           3           2   2   :   5   0   :   5   7
0000120  S   T           2   0   0   4  \r  \n  s   h   -   2   .
0000140  $           \n  s   c   r   i   p   t           d   o   n   e
0000160  n           S   u   n           O   c   t           0   3           2
0000200  5   1   :   0   0           2   0   0   4  \n
0000213
```

## Chapitre 15

***Programmation en langage AWK***

Nom « `awk` » déduit des noms des auteurs (*Aho, Weinberger, Kernighan*)

C'est un utilitaire recherchant des motifs dans un fichier et réalisant des opérations sur les lignes répondant aux critères.

C'est plus généralement un mini langage de programmation à la syntaxe proche du langage C.

Il est très souvent utilisé pour réaliser des filtres sur des fichiers.

Se reporter à un manuel disponible en français :

`ftp://ftp.imag.fr/pub/DOC.UNIX/AWK/awk.pdf`

`ftp://ftp.imag.fr/pub/DOC.UNIX/AWK/awk.ps.gz`

Syntaxe : `awk [options] [fichiers]`

Options intéressantes :

- option « `-FC` » où C est le caractère séparateur de champs
- option « `-f exemple.awk` » pour indiquer le fichier « `exemple.awk` » contenant le programme `awk` à exécuter

## Chapitre 15 • Programmation en langage AWK

## §15.3 • Structure d'un programme AWK

Un programme AWK peut être composé de :

- définition de fonctions
- instructions

◇ Définition d'une fonction

```
function nom(paramètres) { instructions }
```

◇ Format des instructions :

```
[masque] { instructions }
```

◇ Variables utilisables :

- champs : « \$0 », « \$1 », « \$2 », ...
- variables prédéfinies
- variables utilisateur

## Chapitre 15 • Programmation en langage AWK

## §15.4 • Variables prédéfinies de AWK

« FS »	caractère séparateur de champs
« NF »	nombre de champs sur la ligne courante
« NR »	numéro d'ordre de la ligne courante
« RS »	caractère séparateur de lignes
« OFS »	caractère séparateur de champs, en sortie
« ORS »	caractère séparateur de lignes, en sortie
« FILENAME »	nom du fichier en cours de traitement
« ENVIRON[ ] »	variables d'environnement

Les masques conditionnent les instructions à exécuter sur chaque ligne.

Les masques sont évalués lors de la lecture de chaque ligne.

Seuls les instructions associées aux masques actifs sont exécutées pour chaque ligne.

Principaux masques :

- « BEGIN »
- « END »
- « /expression rationnelle/ »

++	incréméntation
--	décréméntation
!	négation logique
* / % + -	multiplication, division, reste, addition, soustraction
< > <= >=	infériorité, supériorité
== =	égalité, différence
&&	ET logique
	OU logique



### ◇ Test

```
if (condition) instruction [ else instruction]
```

### ◇ Boucles

```
while (condition) instruction  
do instruction  
while(condition)  
for (expr1; expr2; expr3) instruction
```

### ◇ Contrôle de boucles

```
break  
continue
```

### ◇ Terminaison

```
exit [code_de_retour]
```

### ◇ Passage à la ligne suivante

```
next
```

### ◇ Affichage de données

```
print [expressions] [>fichier]  
printf format, expressions [>fichier]
```

◇ Longueur d'une chaîne

```
length (chaîne)
```

◇ Valeur entière d'une expression

```
int (expression)
```

◇ Recherche d'une chaîne dans une autre

```
index(chaîne1, chaîne2)
```

◇ Extraction d'une sous-chaîne

```
substr(chaîne, position, longueur)
```

## Exemples :

## ■ Affichage de tous les noms d'utilisateurs

```
awk -F: '{ print $1 }' < /etc/passwd
```

## ■ Inversion des champs 3 et 4

```
awk '{ print $1, $2, $4, $3 }'
```

## ■ Affichage du contenu d'un fichier avec numérotation des lignes

```
awk '{ print NR, $0 }'
```

## ■ Cumul des sommes présentes en troisième colonne

```
awk '{ s += $3 } END { print s }'
```

## ■ Suppression de toutes les lignes dont le premier champ est égal à celui de la ligne précédente

```
awk '{if ( $1 != prev ) { print; prev = $1; } }'
```

- Vérification que toutes les lignes ont le même nombre de champs que la première

```
BEGIN { nberr = 0 }
{ if (NR == 1)
    nb = NF;
  else
    if ( nb != NF )
      nberr++;
}
END {
  if ( nberr != 0 )
    print nberr , " enregistrements incorrects ";
}
```

- Tuer tous les processus UNIX appartenant à l'utilisateur « thb » :

```
% ps aux | awk '$1=="thb" {print "kill " $2}' | sh
```

## Chapitre 16

# *Langage perl*

PERL  $\equiv$  *Practical Extraction and Report Language*

Programme et modules téléchargeables sur <http://www.cpan.org>

Cf <http://www.activestate.com> pour une version de PERL pour Windows.

Langage de script :

```
#!/usr/bin/perl -wall
# exemple de "hello world!"
print " hello world! ";
```

Comme pour le langage C, une instruction se termine toujours par un point-virgule « ; ».

Pas de distinction entre nombres entiers et nombres flottants.

Tous les calculs internes sont faits en nombres flottants.

2 types de chaînes de caractères :

- les simple-quote ; aucun traitement sur le contenu
- les double-quote ; remplacement des caractères de formatage ainsi que des variables.

Les caractères de formatage (proches du langage C) :

Caractère	Action
\n	Nouvelle ligne
\r	Retour
\t	Tabulation
\b	Backspace
\v	Tabulation verticale
\e	Escape
\\	Backslash
\"	Double quote
\a	Bruit d'une cloche

Caractère dollar « \$ » pour spécifier une variable.

Caractères autorisés : lettres, nombres, underscore « \_ »

Le premier caractère ne peut pas être un chiffre.

Exemples :

```
$i = 10;
$i2 = 20;
$string = "hello world!";
$string_2 = "je dis : ${string}";
```

- Opérations sur les variables scalaires :

- Assignment d'une valeur :

```
$a = 10;
```

- Opérations sur les variables :

```
$a = $a + 1;
```

```
$a += 10; # équivalent à $a=$a+10;
```

```
$a /= 20;
```

```
$c = $a * $b;
```

- Incrément automatique

```
$a++; # incrément de 1
```

```
$a = $b++;
```

```
$b--;
```

- Opérations sur les chaînes :

- Assignment d'une chaîne :

```
$string = "Hello world!";
```

- Concaténation de 2 chaînes

```
$string = "${string_a}${string_b}
```

- Suppression du dernier caractère d'une chaîne

```
chop($string);
```

- Suppression du dernier newline d'une chaîne

```
chomp($string);
```

## Chapitre 16 • Langage perl

### §16.5 • Les listes simples

Le caractère « @ » sert à désigner une liste simple.

```
@foo = ("un", "deux", "trois");
```

```
@foo = (@list, 6, 7);
```

```
$elt = @test[4];
```

Liste prédéfinie « @ARGV » (« \$ARGV[0] », « \$ARGV[1] », etc.)

## Opérations sur les listes simples :

## ■ Assignement d'une liste

```
@foo = ("un", "deux", "trois");
@list = (1, 2, 3, 4, 5);
```

## ■ Taille d'une liste

```
$taille = @list;
```

## ■ Accès en absolu sur un élément de la liste

```
@test = (1,2,3,4,5,6,7,8,9);
$elt = @test[4]; # ->valeur 5
@test[4] = 0; # on remplace 5 par 0
```

## ■ Ajout d'un ou plusieurs éléments dans une liste

```
@foo = (@list, "6", "7");
@list = ("0", @list);
```

## ■ Ajouter un ou plusieurs éléments en fin de liste

```
@test = (1, 2, 3);
push(@test, 4,5);
```

## Opérations sur les listes simples :

## ■ Supprimer le dernier élément de la liste

```
$elt = pop(@test);
```

## ■ Ajouter un ou plusieurs éléments en début de liste

```
@test = (1, 2, 3);
unshift(@test, 4,5);
```

## ■ Supprimer le premier élément de la liste

```
$elt = shift(@test);
```

## ■ Duplication d'une liste

```
@liste2 = @liste;
```

## ■ Inverser l'ordre des éléments d'une liste

```
@new_liste = reverse(@liste);
```

## ■ Trier une liste

```
@new_liste = sort(@liste);
```

Les listes simples utilisent des valeurs numériques pour référencer les données (on peut adresser le quatrième élément d'une liste avec « @test[4] »).

Les listes associatives utilisent une notion de clef arbitraire pour indexer les données.

C'est le caractère « % » qui va désigner une liste associative.

Un élément d'une liste associative sera représenté comme :

« \$nom\_liste{\$clef} »

Par exemple :

```
$test{2001} = "hello world";
$linux{linus} = "torwalds";
$linux{tux} = "pingouin";
$data = $linux{linus};
```

Si on utilise une liste associative comme une liste simple avec des clefs d'index numériques, par exemple :

```
$data{1} = "un";
$data{2} = "deux";
...
$data{9} = "neuf";
```

alors on va pouvoir convertir une liste associative en liste simple

(clef, valeur) :

```
@liste_simple = %data ;
```

ce qui nous donne :

```
@liste_simple = (1, "un", 2, "deux", ..., 9, "neuf");
```

ou inversement :

```
@liste_simple = { "un", "deux", ... "neuf" };
%data = @liste_simple;
```



## Opérations sur les listes associatives :

## ■ Assignement d'un élément

```
$test{1} = "un";
$test{un} = 1;
```

## ■ Suppression d'un élément dans une liste

```
delete $test{1};
```

## ■ Taille d'une liste

```
$taille = %list;
```

## ■ Accès en absolu à un élément de la liste

```
print $test{un};
```

## ■ Duplication d'une liste

```
%list_dup = %list;
```

## ■ Pour récupérer les clefs d'une liste

```
@liste_clefs = keys(%test);
```

## ■ Pour récupérer les valeurs d'une liste

```
@liste_valeurs = values(%test);
```

## ■ Pour parcourir une liste

```
while ( ($clef, $valeur) = each(%test) )
    { print "$clef ; $valeur\n"; }
```

## Chapitre 16 • Langage perl

## §16.7 • Structure de contrôle if / else

La structure `if / else` est identique à celles des langages évolués :

```
if (expression)
{
    lignes;
}
```

```
if (expression)
{
    lignes;
}
else
{
    lignes;
}
```

```
if (expression)
{
    lignes;
}
elseif (expression)
{
    lignes;
}
else
{
    lignes;
}
```

Autre syntaxe possible :

```
commande if expression;
```

## Chapitre 16 • Langage perl

### §16.8 • Structure de contrôle while, until, for

Tant que l'expression est vraie alors on exécute le code :

```
while (expression)
{
    lignes;
}
```

Jusqu'à ce que l'expression soit vraie, on exécute le code :

```
until (expression)
{
    lignes;
}
```

Très proche du langage C. On initialise une variable, on l'incrémente et on teste la valeur de la variable à chaque itération :

```
for ( initialisation ; test; increment )
{
    lignes;
}
```

Prends une liste de données en entrée et en assigne une dans une variable à chaque itération jusqu'à ce que la liste soit vide :

```
foreach $var (@liste)
{
    lignes;
}
```

Opérateur	Numérique	Chaines
Egalité	==	eq
Pas d'égalité	!=	ne
Plus petit que	<	lt
plus grand que	>	gt
Plus petit ou égal à	<=	le
Plus grand ou égal à	>=	ge

Opérateur et logique : `and`

Opérateur ou logique : `or`

Opérateur de négation : `!`

Notion de file descripteur tout comme en C

Un descripteur est représenté par une variable.

Conventionnellement écrit en lettres majuscules :

- entrée standard STDIN
- sortie standard STDOUT
- sortie erreur STDERR

Descripteur STDIN

Pour récupérer une ligne

```
$ligne = <STDIN> ;  
$ligne = <> ;
```

Pour récupérer plusieurs lignes

```
@texte = <STDIN> ;  
@texte = <> ;
```

## Chapitre 16 • Langage perl

## §16.13 • Entrées/sorties : sortie standard, STDOUT

Commandes `print` et `printf` pour écrire sur la sortie standard `STDOUT` du terminal.

Il n'est pas nécessaire de spécifier le descripteur `STDOUT` car celui-ci est pris en compte par défaut :

Exemples d'utilisation de la commande `print` :

```
$toto=10;
print $toto; # print STDOUT $toto;
print "$toto\n"; # print STDOUT "$toto\n";
print 10+20;
print "coucou tout le monde\n";
```

Exemple d'utilisation de la commande `printf` :

```
$foo=35;
$string="Thierry";
printf "Mon nom est %s ; j'ai %d ans !\n", $foo, $string;
```

qui donnera :

```
Mon nom est Thierry ; j'ai 35 ans !
```

## Chapitre 16 • Langage perl

## §16.14 • Entrées/sorties : sortie erreur, STDERR

De la même manière que la sortie standard, on va utiliser le descripteur `STDERR`.

## Chapitre 16 • Langage perl

## §16.15 • Créer, ouvrir et fermer un fichier, open(), close(), eof(), die()

## Pour lire un fichier

```
open( FILE, "nom_du_fichier" );
open( FILE, "<nom_du_fichier" );
```

## Pour créer ou écraser un fichier

```
open( FILE, ">nom_du_fichier" );
```

## Pour écrire à la fin d'un fichier

```
open( FILE, ">>nom_du_fichier" );
```

## Pour fermer un fichier

```
close( FILE );
```

## Tester la fin du d'un fichier

```
eof( FILE );
```

Manière élégante de tester la réussite de l'appel à `open()` via la fonction `die()` :

```
open( FILE, "foo.txt" ) || die "Impossible d'ouvrir fichier.txt!\n";
```

## Chapitre 16 • Langage perl

## §16.16 • Lire et écrire dans un fichier

On utilise les mêmes techniques que celles utilisées dans les flux d'entrées/sorties `STDOUT`, `STDIN` et `STDERR` mais en remplaçant le descripteur par celui du fichier.

```
open( FILE, ">foo.txt" ) || die "Impossible d'ouvrir fichier.txt!\`
print FILE "Hello world!";
close( FILE );
```

```
open( FILE, "foo.txt" ) || die "Impossible d'ouvrir fichier.txt!\r
$string = <FILE>;
print $string;
close( FILE );
```

Pour supprimer un fichier :

```
unlink( "fichier.txt" );
```

Pour supprimer plusieurs fichiers à la fois :

```
unlink( "foo1.txt", "foo2.txt", ... );
```

Dans l'exemple suivant, on renomme le fichier `trace.log` en `old_trace.old` :

```
rename( "trace.log", "old_trace.log" );
```

## Chapitre 16 • Langage perl

§16.19 • Créer un lien symbolique, `symlink()`, `readlink()`

Dans l'exemple suivant, on met en place le lien symbolique `verb !toto !` sur le fichier `data.log` :

```
symlink( "data.log", "toto" );
```

Nota : Il est possible d'obtenir le lien pointé par un lien symbolique en utilisant la fonction `readlink()`.

## Chapitre 16 • Langage perl

§16.20 • Modifier les propriétés d'un fichier, `chmod()`, `chown()`, `chgrp()`

```
chmod( 0744, "fichier.txt" );  
chmod( 0700, 4740, ..., "fichier1.txt", "fichier2.txt", ... );  
chown( 456, "foo.txt" );  
chgrp( 65600, "foo.txt" );
```



## Chapitre 16 • Langage perl

§16.21 • Créer et supprimer un répertoire, `mkdir()`, `rmdir()`

```
mkdir( "/tmp/rep_tempo", 0744 );  
rmdir( "/tmp/rep_tempo" );
```

## Chapitre 16 • Langage perl

§16.22 • Exécuter un programme, `exec()`, `open()`

Commande `exec()` pour exécuter un programme :

```
exec("ls -la");
```

De la même manière, Perl vous autorise à exécuter un programme de votre système et à lui transmettre des informations. Dans l'exemple suivant nous allons envoyer un mail en utilisant un serveur smtp local (sendmail) :

```
open( MAIL, "| /usr/bin/sendmail -oi -n -t" );  
print MAIL << __MAIL__;  
To:root@localhost.localdomain  
From:root  
Hello world!  
__MAIL__  
close( MAIL );
```

Même exemple mais cette fois ci pour récupérer une sortie standard :

```
open( MESSAGES, "| tail -f /var/log/messages" );
while(!eof(MESSAGES ))
{
    $str = ;
    print $str;
}
close( MESSAGES );
```

## Chapitre 16 • Langage perl

### §16.23 • Expressions régulières, \$\_

Vous pouvez tester facilement une chaîne avec une expression régulière en l'entourant de 2 caractères slash /. Le test se fera sur la variable \$\_.

```
$_ = "hello world";
if (/ [Hh]ello [ ]*world/)
{
    # teste la variable $_
    print "Hello world\n";
}
```

## Chapitre 16 • Langage perl

## §16.24 • Expressions régulières : substitution dans une chaîne, s///, =~

Pour cela on va utiliser l'opérateur de substitution :

```
s/expression_régulière/chaine_de_replacement/
```

```
$_ = "hello world! hello all! ";
s/hello/Hello/; # devient "Hello world! hello all!"
```

On peut aussi utiliser l'option `g` pour que la substitution se passe sur toutes les occurrences de la ligne.

```
$_ = "Je m'appelle gordie, gordie Lachance.";
s/gordie/jerome/g; # devient "Je m'appelle jerome, jerome Lachance."
```

On peut aussi vouloir appliquer une substitution sur une variable autre que

`$_` :

```
$str = "Je m'appelle gordie, gordie Lachance.";
$str =~ s/gordie/jerome/g;
```

## Chapitre 16 • Langage perl

## §16.25 • Expressions régulières : split(), join()

Pour découper vos chaînes de caractères suivant une expression régulière :

```
$str = "un deux trois quatre";
@liste = split(/ /, $str); # découpe par rapport aux espaces
# @liste = ( "un", "deux", "trois", "quatre" );
```

Pour joindre vos chaînes de caractères suivant une expression régulière :

```
@liste = ( "un", "deux", "trois", "quatre" );
$str = join(/-/ , @liste);
# $str = "un-deux-trois-quatre";
```

◇ Déclaration d'une fonction :

```
sub nom_fonction
{
    ligne_1;
    ligne_2;
    ...
    ligne_n;
}
```

◇ Invoquer une fonction :

Pour invoquer une fonction, faire précéder le nom de la fonction du caractère & :

```
&foo;
&call_fonction(12,13);
```

Il est possible aussi de passer des paramètres à la fonction : dans ce cas, tous les arguments seront disponibles sous la forme d'une liste simple dans la variable « @\_ »

◇ Déclaration d'une variable locale dans une fonction :

Pour déclarer une variable locale dans une fonction Perl, vous devez utiliser la fonction « local() ».

```
sub foo
{
    local($toto); # déclaration locale d'une variable
    ...
}
```

La durée de vie de votre variable sera ainsi limitée à l'appel de la fonction.

◇ Code de retour :

Le code retourné par une fonction est la valeur de la dernière expression évaluée.

De préférence, utiliser la commande « return » :

```
sub foo
{
    local($code_retour); # déclaration locale d'une variable
    ...
    return $code_retour; # code de retour
}
```