Administration système UNIX

version 6.3

Partie 1

2003 – 2004

Thierry Besançon – Philippe Weill

Formation Permanente
Université Pierre et Marie Curie - PARIS 6
Tour centrale 13e étage
4, place Jussieu
75252 Paris Cedex 05
Tél. : 01-44-27-58-49 ou 01-44-27-58-50
01-44-27-38-19 ou 01-44-2738-25
Fax : 01-44-27-27-15
e-mail : formation.permanente@admp6.jussieu.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/2003-2004/UNIX/cours/

Si vous améliorez ce cours, merci de m'envoyez vos modifications ! :-)

"...the number of UNIX installations has grown to 10, with more expected..."
- Dennis Ritchie and Ken Thompson, June 1972

Table des matières

N° de transparent

Chapitre 1 : UNIX : Généralités – Historique

§ 1.1 UNIX, un système d'exploitation

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

◇ Terminologie :

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

◇ Exemples :

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

◇ Concepts novateurs d'UNIX :

« UNIX est construit autour d'une idée forte : la puissance d'un système provient plus des relations entre les programmes que des programmes eux-mêmes. Beaucoup de programmes UNIX font, de façon isolé des traitements triviaux ; combinés avec d'autres, ils deviennent des outils généraux et performants.

La solution d'un problème sous UNIX ne passe pas forcément par l'écriture d'un programme spécifique mais souvent pas une utilisation combinée et élégante des outils standard.»

§ 1.2 Panorama de quelques UNIX du marché

Marque	*Site web*	*Version d'UNIX*	*Constructeur de hardware*
APPLE	`http ://www.apple.com`	MacOS X 10.x	oui
COMPAQ	`http ://www.digital.com`	Tru64 Unix 5.x	oui
CRAY	`http ://www.cray.com`	Unicos ?. ?	oui
HP	`http ://www.hp.com`	HP-UX 11.x	oui
IBM	`http ://www.ibm.com`	AIX 5.x	oui
SGI	`http ://www.sgi.com`	IRIX 6.x.y	oui
SUN	`http ://www.sun.com`	Solaris 9	oui
SANTA CRUZ	`http ://www.sco.com`	Unixware 7.x	non
LINUX	`http ://www.kernel.org`	noyau 2.6.x	non
FREEBSD	`http ://www.freebsd.org`	FreeBSD 5.x	non
NETBSD	`http ://www.netbsd.org`	NetBSD 1.x	non
OPENBSD	`http ://www.openbsd.org`	OpenBSD 3.x	non

L'arbre généalogique d'UNIX est très complexe. Se reporter à l'annexe pour un schéma détaillé.

§ 1.3 Les différentes familles d'UNIX

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 !

§ 1.4 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, `http://www.mandrake.com` (société au bord de la faillite)
– Debian, `http://www.debian.org`
– Knoppix, `http://www.knoppix.org`

§ 1.5 UNIX à la formation permanente

La salle de TP de la Formation Permanente est équipée de PC sous Red Hat 7.2.

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.

◇ Thierry Besançon ou Philippe Weill

email : `Thierry.Besancon@formation.jussieu.fr`

email : `Philippe.Weill@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

§ 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 (piratage informatique, articles pédophiles)
– 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.

§ 2.2 Quelques règles de bon sens

1. Votre pire ennemi, c'est vous.
   ```
   rm /tmp *
   ```
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.
   ```
   mv foo foo.0
   cp foo.0 foo
   vi foo
   ```
5. Documentez ce que vous faites.
6. Faites comme si vous ne pouviez pas venir demain.
7. Attention à l'expansion des caractères génériques.
   ```
   rm -rf .*
   ```

Administrateur système == technicité + rigueur + bon sens

§ 2.3 Connaissances de base d'un administrateur

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 sytème 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 utilisateur «normaux» de le déconfigurer.

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

§ 3.1 Votre compte UNIX : `login`, mot de passe

Un utilisateur Unix est equivalent 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 $\equiv$ réglements informatiques.

§ 3.2 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.

Cf `http://www.cru.fr/securite/Cours/mot-de-passe-jplg.html`

§ 3.3 Changer son mot de passe UNIX

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

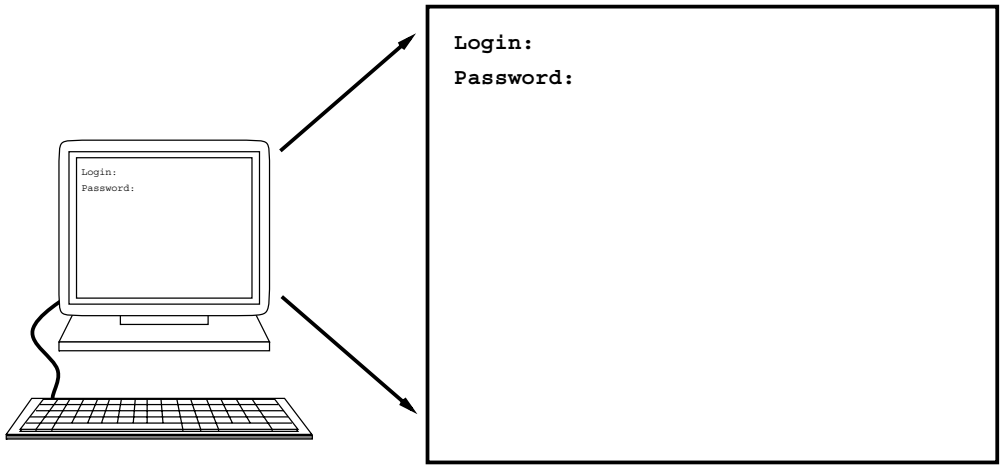
Sur les systèmes UNIX qui utilisent un mécanisme de centralisation des mots de passe (appelé NIS), la commande pour changer son mot de passe est `passwd`.

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

§ 3.4 Connexion sur un terminal texte UNIX

Terminal texte (modèle VT100)

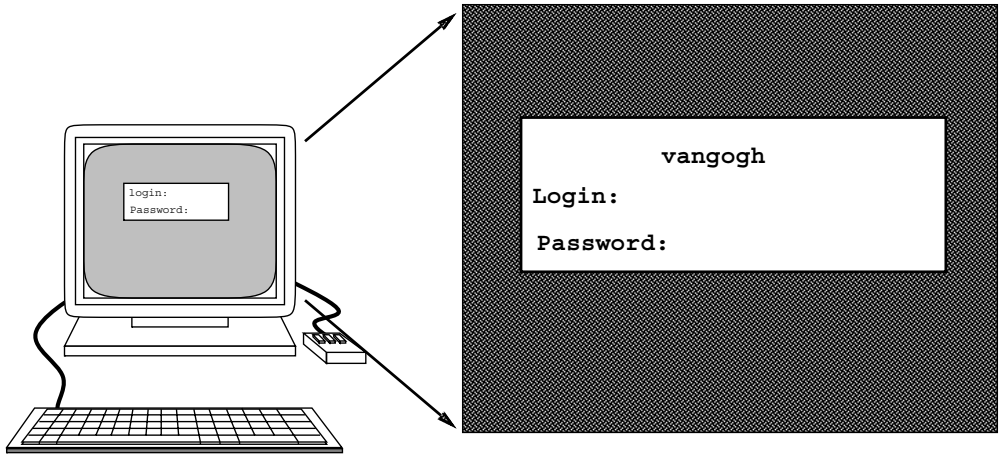
La demande du login et du mot de passe ressemble globalement à :



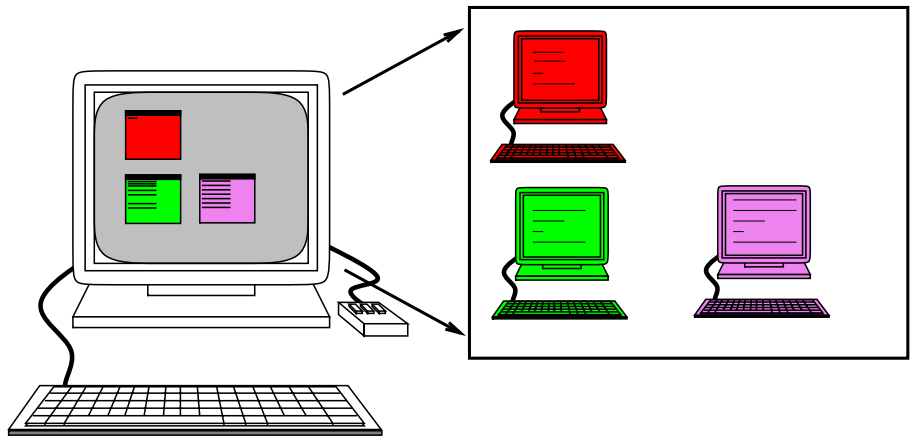
§ 3.5 Connexion sur un terminal graphique UNIX

Station de travail UNIX (SUN Blade 100)

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



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 :



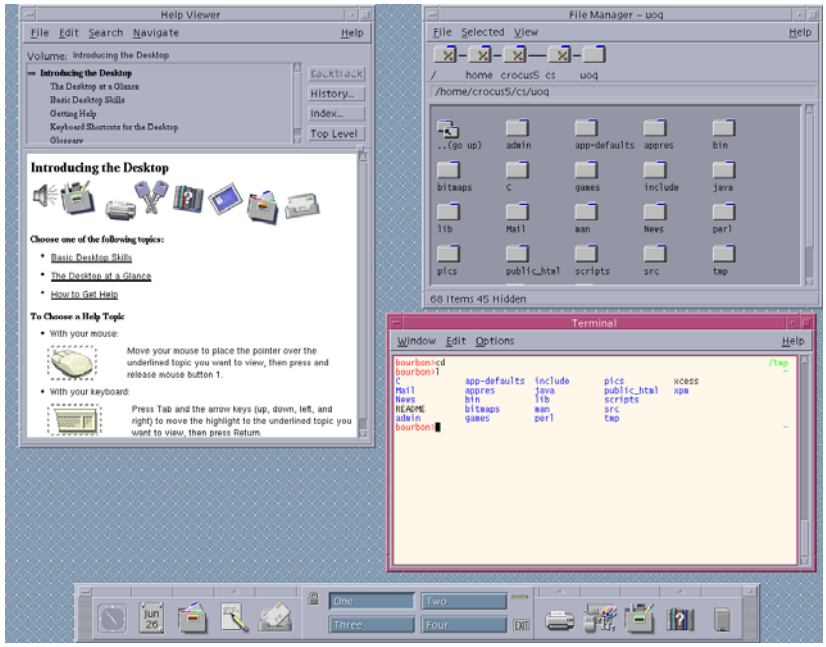
L'émulateur de terminal s'appelle `xterm`.

§ 3.6 L'interface graphique UNIX

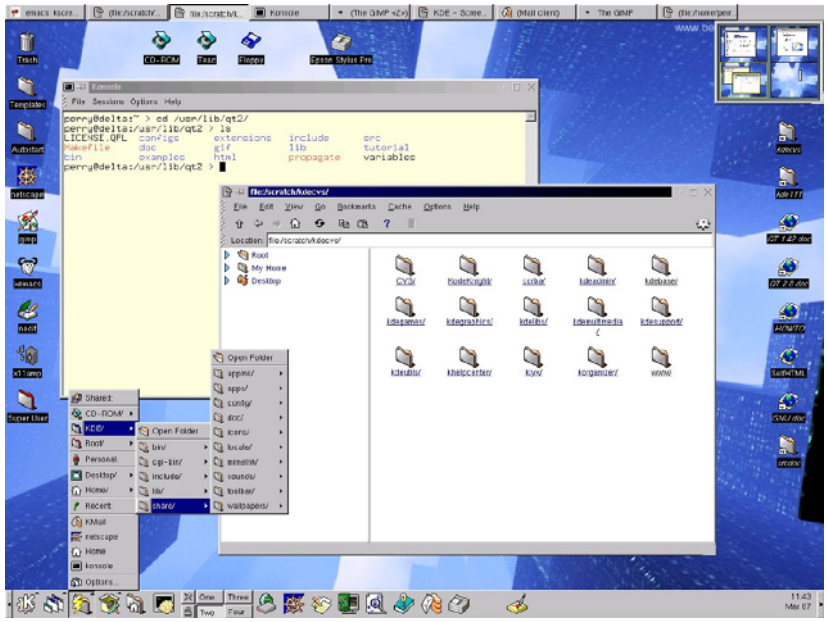
Unix dispose d'un grand nombre d'interfaces graphiques comparé à des systèmes comme Windows ou Macintosh :
SunView, NeWS, OpenWindows, View, CDE, KDE, GNOME, Berlin, etc.

Cf le site `http://www.plig.org/xwinman/`

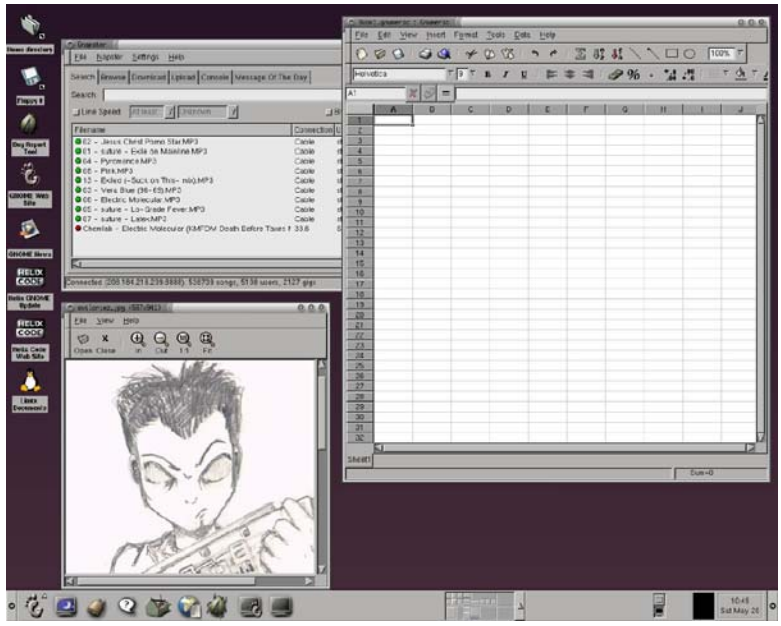
Bureau de travail sous CDE :



Bureau de travail sous KDE (`http://www.kde.org`) :



Bureau de travail sous GNOME (`http://www.gnome.org`) :



§ 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, une console texte DIGITAL VT100.

A droite, un teletype DIGITAL (genre de machine à écrire).

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 :

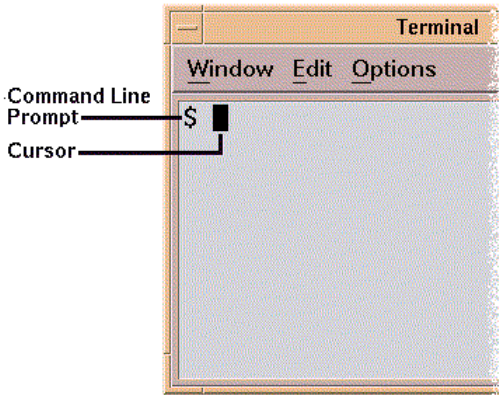
Shell	**Program**	**Description**
Bourne shell	sh	disponible sur toute plateforme Unix
C shell	csh	shell développé pour BSD
Korn shell	ksh	Bourne shell amélioré par AT&T
Bourne again shell	bash	Shell distribué avec linux ; version améliorée de sh et csh

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.

La suite du support de cours comportera des exemples comme :

```
% ls
```

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
```

§ 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 ⇒ 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 des options :

`ls -a -l` ≡ `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` ≡ `ls -al` ≡ `ls -la` ≡ `ls -l -a`

Chapitre 4 : Sources de documentation

Beaucoup de documentation disponible.

Souvent en anglais.

§ 4.1 Documentation Unix en ligne : `man`

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
– 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
```

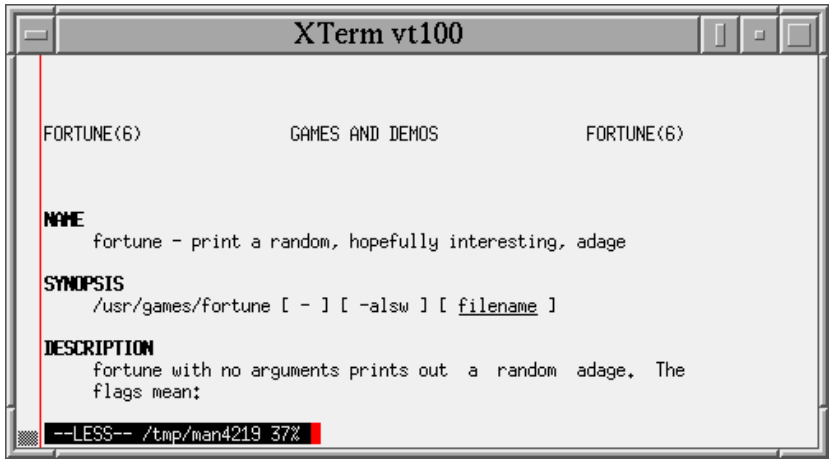
avec en particulier comme option :

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

⇒ Inconvénient : il faut connaitre 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 :



On remarque :

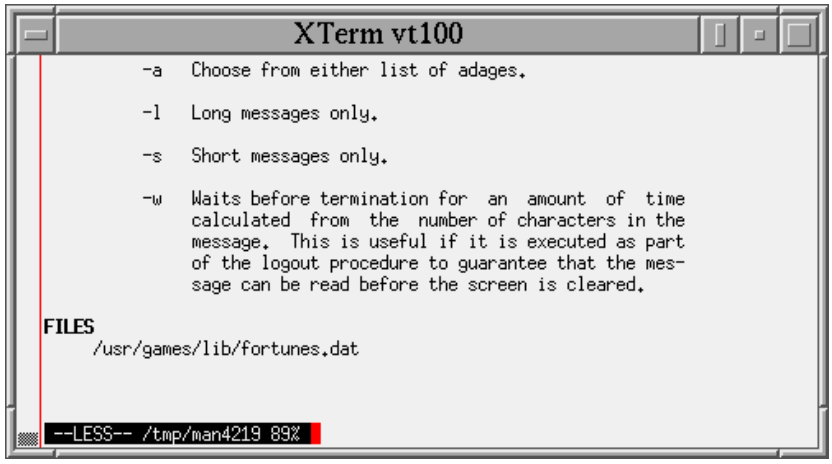
– affichage page d'écran par page d'écran pour mieux lire la doc

– plusieurs rubriques (NAME, SYNOPSIS, DESCRIPTION, . . .)

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

– la touche SPC pour avancer (ou f ≡ *forward*)

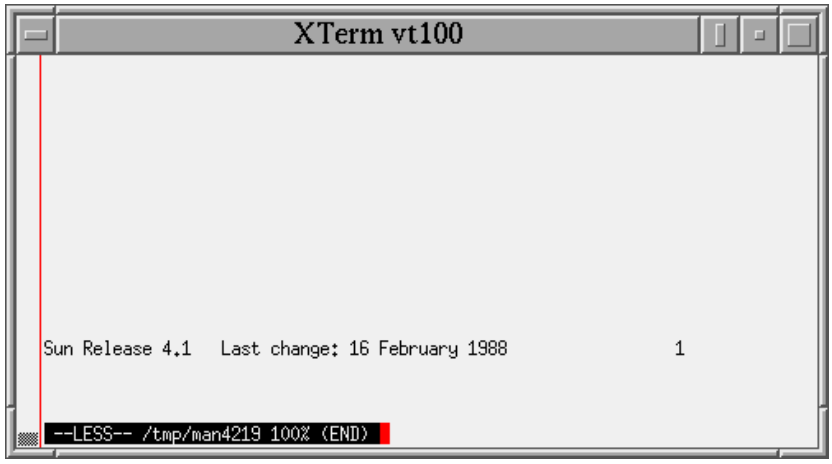
– la touche b pour reculer (b ≡ *backward*)



On quitte :

– quand on arrive à la fin de la documentation

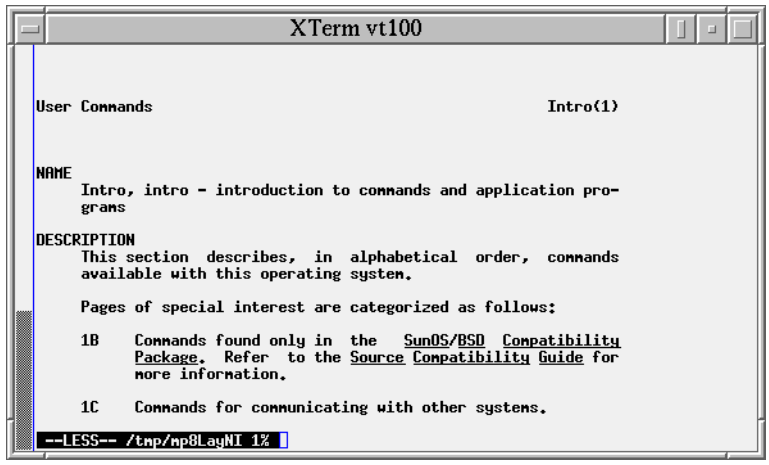
– prématurément par la touche q (q ≡ *quit*)



Quelques pages de manuel intéressantes :

– la page de la commande `man` que l'obtient par `man man`

– les pages d'introduction de chaque section que l'on obtient par `man 1 intro`, `man 2 intro`, `man 3 intro`, etc.



Quand on ne connait pas le nom de la commande, on peut demander les noms des commandes dont le descriptif contient une certaine chaîne de caractères :

```
man -k chaine
```

Exemple :

```
% man -k tune
xvidtune(1)    - video mode tuner for XFree86
xvidtune(1)    - video mode tuner for XFree86
fortune(6)     - print a random, hopefully interesting, adage
tunefs(8)      - tune up an existing file system
```

§ 4.2 RFC, Internet drafts

Documents de référence en anglais récupérables aux adresses :

– `ftp://ftp.lip6.fr/pub/rfc/rfc/`

– `ftp://ftp.lip6.fr/pub/rfc/internet-drafts/`

§ 4.3 FAQ

(en anglais *Frequently Asked Questions*, en fran cais *Foire Aux Questions*)

Documents en anglais récupérables aux adresses :

– `ftp://ftp.lip6.fr/pub/doc/faqs/`

§ 4.4 HOWTO Linux

Documents en anglais récupérables aux adresses :

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

§ 4.5 Newsgroups

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`

§ 4.6 Moteur de recherche Google

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

§ 4.7 Documentations constructeur online

Certains constructeurs Unix mettent des documentations et documents online.

Se reporter par exemple à :

– `http://docs.sun.com/`

§ 4.8 Librairies parisiennes

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/`

§ 4.9 Magazines

De nombreux magazines parlent de LINUX.

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

§ 4.10 Formats des documentations

Formats les plus répandus (en vrac) :

- **format PDF** ; à lire avec
  - Acrobat Reader ; `http://www.acrobat.com/` ou `http://www.adobe.com/products/acrobat/`
  - Ghostscript ; `http://www.ghostscript.com/`
  - xpdf ; `http://www.foolabs.com/xpdf/`

- **format Postscript** ; à lire avec
  - Ghostscript ; `http://www.ghostscript.com/`
  - ghostview ; `ftp://ftp.lip6.fr/pub/gnu/ghostview/`
  - gv ; `http://wwwthep.physik.uni-mainz.de/~plass/gv/`

- **format Microsoft Word** ; à lire avec
  - Microsoft Word ; `http://www.microsoft.com/office/word/`
  - Microsoft Word viewer ; `http://www.microsoft.com/???/`
  - Star Office ; `http://wwws.sun.com/software/star/staroffice/6.0/`
  - Open Office ; `http://www.openoffice.org/`
  - antiword ; `http://www.antiword.org/`

- **format texte** ; n'importe quel éditeur de texte

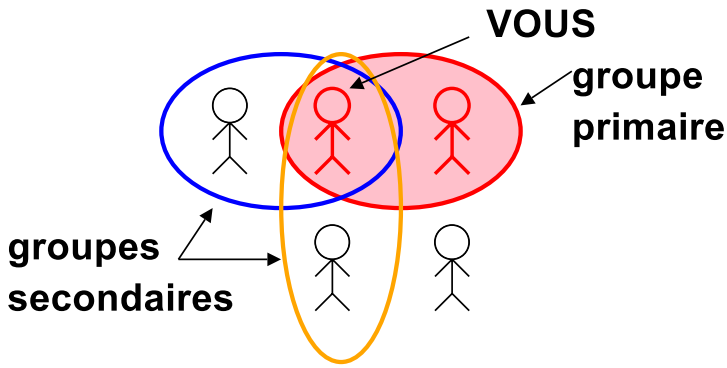
- **format HTML** ; n'importe quel browser web

Chapitre 5 : Commandes de manipulations de base des objets Unix

§ 5.1 Propriétaires des objets sous Unix

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



§ 5.2 Arborescence des objets, chemins absolus et relatifs

Sur Unix, les objets sont identifiés par leur chemin dans l'arborescence, qui peuvent contenir n'importe quel caractère (sauf /) et peuvent faire jusqu'à 256 caractères de long, voire plus selon les Unix.

Au concept d'objet est associé la notion de répertoire (en anglais directory). Un répertoire est simplement une collection d'objets organisée de manière arborescente.

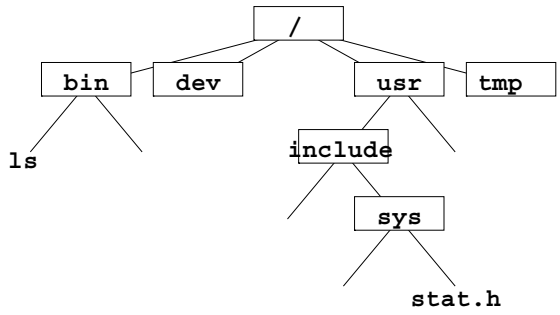
Un objet peut être référencé par son chemin d'accès, qui est constitué du nom d'objet, précédé par le nom de répertoire qui le contient sous la forme :

chemin d'accès = répertoire / nom

◇ Arborescence

Principe classique de l'**arborescence** :

– une racine : désignée par "/" (en anglais «**slash**»)
– nœuds : répertoires
– feuilles : fichiers



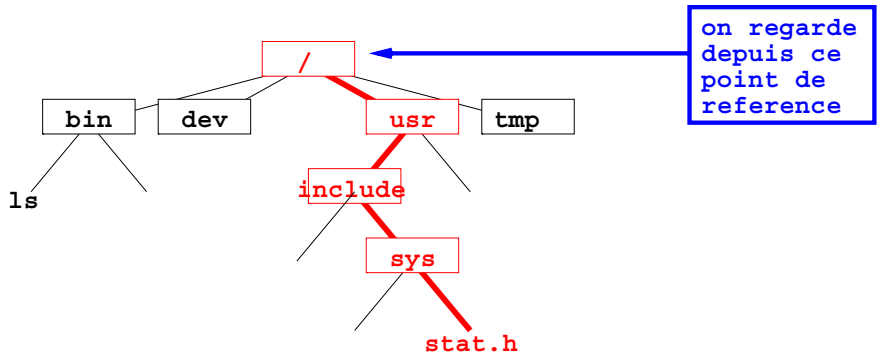
Selon ce schéma, 3 désignations possibles d'un fichier.

◇ **chemin d'accès absolu** :

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

Si le nom de répertoire commence par /, il s'agit d'une référence absolue par rapport au répertoire racine /, constituée d'une liste des répertoires à parcourir depuis la racine pour accèder au fichier.

par exemple : `/usr/include/sys/stat.h`



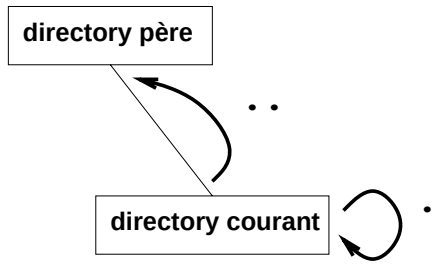
◇ **chemin d'accès relatif** :

chemin d'accès = répertoire / nom

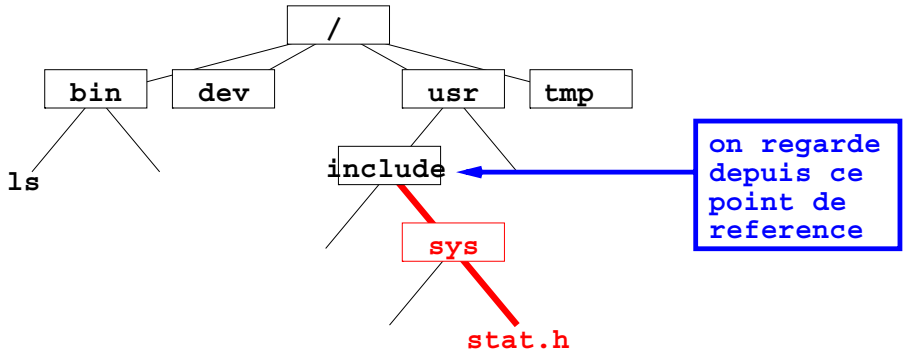
Si le nom de répertoire ne commence pas par /, il s'agit d'une référence relative par rapport au répertoire courant.

Le répertoire courant est noté .

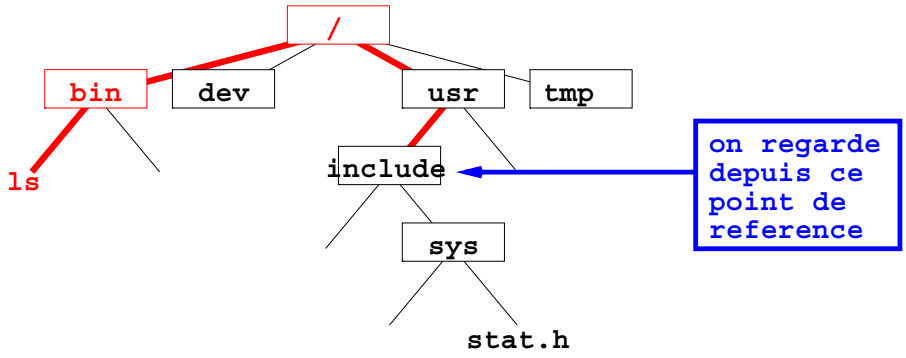
Le répertoire parent du répertoire courant est noté ..



par exemple, depuis `/usr/include/` : `sys/stat.h`



par exemple, depuis `/usr/include/` : `../../bin/ls`



Le répertoire courant est noté `.`

Exemples d'utilisation qui seront revus ultérieurement :

– commande `find` pour lancer une recherche à partir de l'endroit courant :

  « `find . -name exemple -print` »

– lancer une commande dans le répertoire courant :

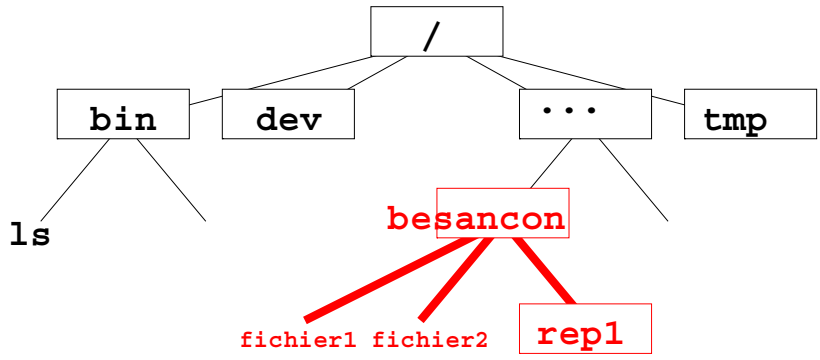
  « `./a.out` »

– etc.

◇ **chemin d'accès relatif par rapport à un utilisateur** :

Sur le principe identique à un chemin relatif sauf que le point de départ est le répertoire d'un utilisateur que l'on désigne par le signe `~` suivi du nom de l'utilisateur :

```
% ls ~besancon
fichier1      fichier2      rep1
```



§ 5.3 Analogies/Différences Unix/Windows/Mac

Monde Unix	**Microsoft Windows**	**Monde Macintosh**
UNE SEULE arborescence	plusieurs volumes (« `C:` », « `D:` », etc.)	plusieurs volumes
écriture `/a/b/c`	écriture `\a\b\c`	écriture `a:b:c`

§5.4 Liste des fichiers : `ls`

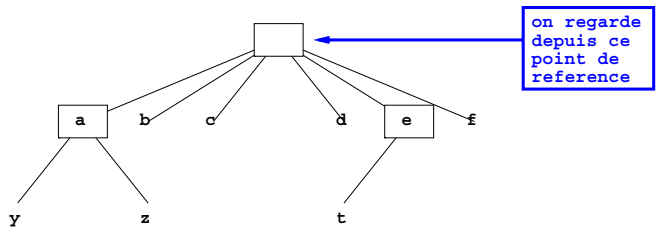
(en anglais *list*)

Syntaxe : `ls [options] objets`

Principales options (cumulables) :

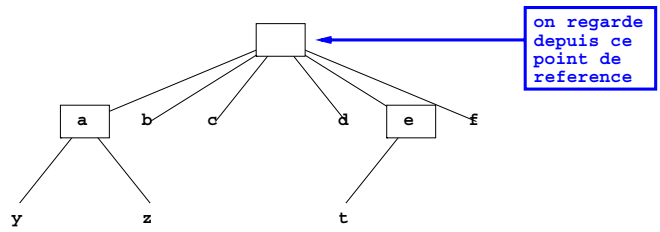
– option `-l` : affichage au format long des informations relatives aux objets
– option `-g` : affichage des groupes propriétaires des objets
– option `-R` : liste récursive
– option `-d` : affichage des noms des répertoires et non de leur contenu
– option `-F` : affichage des objets avec un suffixe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

◇ Exemple 1



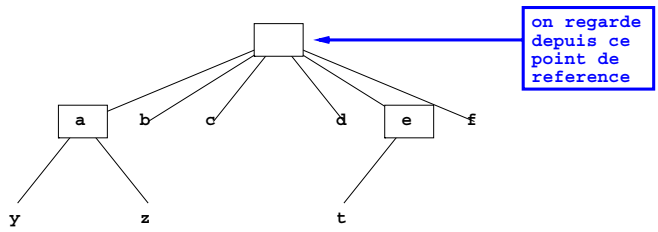

```
% ls
a       b       c       d       e       f
```

◇ Exemple 2



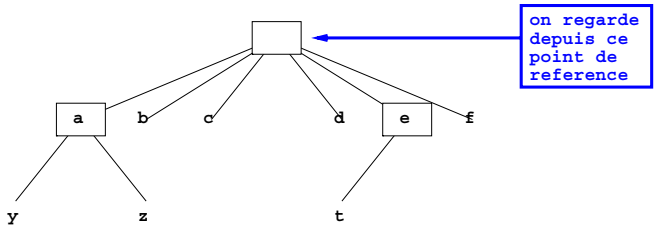

```
% ls -l
total 2
drwxr-xr-x  2 besancon      512 Oct  1 16:42 a
-rw-r--r--  1 besancon        0 Oct  1 16:42 b
-rw-r--r--  1 besancon        0 Oct  1 16:42 c
-rw-r--r--  1 besancon        0 Oct  1 16:42 d
drwxr-xr-x  2 besancon      512 Oct  1 16:42 e
-rw-r--r--  1 besancon        0 Oct  1 16:42 f
```

◇ Exemple 3




```
% ls -R
a       b       c       d       e       f

a:
y       z

e:
t
```

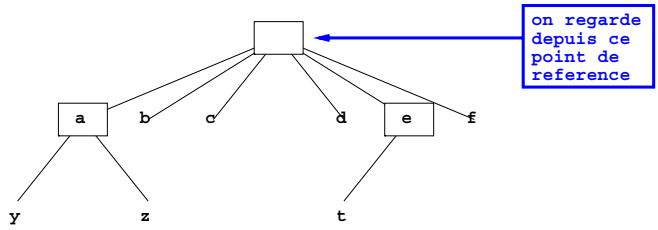
◇ Exemple 4




```
% ls -Rl
total 2
drwxr-xr-x  2 besancon   512 Oct  1 16:43 a
-rw-r--r--  1 besancon     0 Oct  1 16:42 b
-rw-r--r--  1 besancon     0 Oct  1 16:42 c
-rw-r--r--  1 besancon     0 Oct  1 16:42 d
drwxr-xr-x  2 besancon   512 Oct  1 16:43 e
-rw-r--r--  1 besancon     0 Oct  1 16:42 f
```

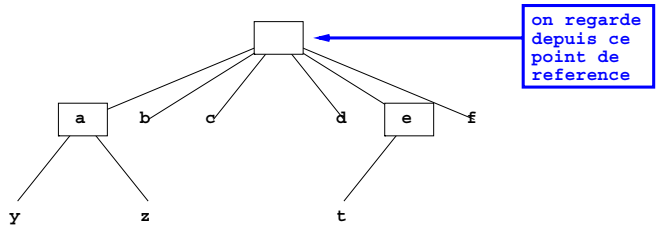
```
a:
total 0
-rw-r--r--  1 besancon     0 Oct  1 16:43 y
-rw-r--r--  1 besancon     0 Oct  1 16:43 z

e:
total 0
-rw-r--r--  1 besancon     0 Oct  1 16:43 t
```

◇ Exemple 5




```
% ls -F
a/      b       c       d       e/      f

% ls -lF
total 2
drwxr-xr-x  2 besancon      512 Oct  1 16:43 a/
-rw-r--r--  1 besancon        0 Oct  1 16:42 b
-rw-r--r--  1 besancon        0 Oct  1 16:42 c
-rw-r--r--  1 besancon        0 Oct  1 16:42 d
drwxr-xr-x  2 besancon      512 Oct  1 16:43 e/
-rw-r--r--  1 besancon        0 Oct  1 16:42 f
```

◇ Exemple 6




```
% ls a
y z

% ls -d a
a
```

```
% ls -l a
-rw-r--r--  1 besancon    0 Oct  1 16:43 y
-rw-r--r--  1 besancon    0 Oct  1 16:43 z

% ls -ld a
drwxr-xr-x  2 besancon  512 Oct  1 16:43 a
```

◇ Exemple 7

```
% ls -aF ~besancon
./                      .lessrc                 .tvtwmrc
../                     .login@                 .twmrc
.Xauthority             .logout@                .weblink
.Xdefaults              .lynxrc                 .workmandb
.Xresources             .mailcap@               .workmanrc
```

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

§5.5 Affichage du contenu d'un fichier texte : `cat`

(en anglais *concatenate*)

Syntaxe : `cat [options] fichiers`

Par exemple :

```
% cat /etc/motd
SunOS Release 4.1.4 (EXCALIBUR [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,
  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.
```

§ 5.6 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 [options] fichiers`

– Caractère q pour quitter
– Caractère espace pour avancer d'une page d'écran
– Caractère b pour revenir en arrière d'une page (backward)
– Caractère f pour avancer d'une page d'écran (forward)

La commande man affiche en fait les pages du manuel au moyen de la commande more.

§ 5.7 Destruction d'un fichier : `rm`

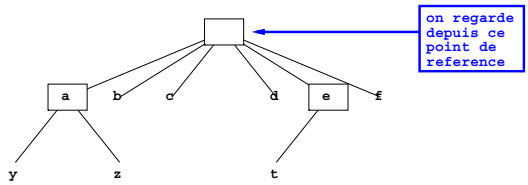
(en anglais *remove*)

Syntaxe : `rm [options] fichiers`

Quelques options :

– option "`-i`" : confirmation à chaque suppression (garde fou)
– option "`-r`" : suppression récursive
– option "`-f`" : suppression en force d'un fichier 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 -F
a/      b       c       d       e/      f

% rm f
% ls -F
a/      b       c       d       e/

% rm a
rm: a is a directory

% rmdir a
rmdir: a: Directory not empty
```

```
% ls a
y       z

% rm -rf a
% ls -F
b       c       d       e/

% rm -i b
rm: remove b? y
% ls -F
c       d       e/
```

§ 5.8 Duplication d'un fichier : cp

(en anglais *copy*)

Syntaxes :

```
cp [options] fichier-départ fichier-destination
cp [options] fichiers-départ répertoire-destination
```

La destination peut être un fichier ou un répertoire.

– Dans le cas d'une destination fichier, on renomme le fichier original.

– Dans le cas d'une destination répertoire, on déplace le fichier original en lui conservant son nom.

Quelques options :

"-i" : confirmation à chaque écrasement de fichier

"-r" : copie récursive

```
% ls -lF
total 3
drwxr-xr-x  2 besancon      512 Oct  1 16:43 a/
-rw-r--r--  1 besancon        0 Oct  1 16:42 b
-rw-r--r--  1 besancon        0 Oct  1 16:42 c
-rw-r--r--  1 besancon        0 Oct  1 16:42 d
drwxr-xr-x  2 besancon      512 Oct  1 16:43 e/
-rw-r--r--  1 besancon      342 Oct  1 17:25 f

% cp f g
% ls -lF
total 4
drwxr-xr-x  2 besancon      512 Oct  1 16:43 a/
-rw-r--r--  1 besancon        0 Oct  1 16:42 b
-rw-r--r--  1 besancon        0 Oct  1 16:42 c
-rw-r--r--  1 besancon        0 Oct  1 16:42 d
drwxr-xr-x  2 besancon      512 Oct  1 16:43 e/
-rw-r--r--  1 besancon      342 Oct  1 17:25 f
-rw-r--r--  1 besancon      342 Oct  1 17:25 g

% cp /tmp/motd h
% ls -l h
-rw-r--r--  1 besancon      752 Oct  1 17:26 h
```

§ 5.9 Déplacement et renommage d'un fichier : mv

(en anglais *move*)

Syntaxes :

```
mv [options] fichier-départ fichier-destination
mv [options] objets-départ répertoire-destination
mv [options] répertoire-depart répertoire-destination
```

Quand source de départ et destination sont du même type (fichier - fichier ou répertoire - répertoire), on réalise un **renommage**. Sinon on **déplace** les objects.

Quelques options :

"-i" : confirmation à chaque écrasement de fichier

```
% ls -lF
total 3
-rw-r--r--  1 besancon        0 Oct  1 16:42 d
drwxr-xr-x  2 besancon      512 Oct  1 16:43 e/
-rw-r--r--  1 besancon      752 Oct  1 17:25 f

% mv f toto
% ls -lF
total 3
-rw-r--r--  1 besancon        0 Oct  1 16:42 d
drwxr-xr-x  2 besancon      512 Oct  1 16:43 e/
-rw-r--r--  1 besancon      752 Oct  1 17:25 toto
```

Dans le cas d'une destination répertoire, on **déplace** le fichier original en lui conservant son nom.

```
% mv toto /tmp
% ls -l toto
toto not found
% ls -l /tmp/toto
-rw-r--r--  1 besancon      752 Oct  1 17:25 /tmp/toto
```

quelques options :

"`-i`" : confirmation à chaque écrasement de fichier

```
% mv -i b c
remove c? y
% ls -l b c
b not found
-rw-r--r--  1 besancon      752 Oct  1 17:30 c
```

§ 5.10 Création de répertoires : mkdir

(en anglais *make directory*)

Syntaxe : `mkdir [options] répertoires`

```
% mkdir z
% ls -ldF z
total 1
drwxr-xr-x  2 besancon      512 Oct  1 17:40 z/
```

Création directe de sous répertoires en cascade :

```
% mkdir -p repertoire1/ss-repertoire2/ss-ss-repertoire3
% ls -R
repertoire1/

repertoire1:
ss-repertoire2/

repertoire1/ss-repertoire2:
ss-ss-repertoire3/

repertoire1/ss-repertoire2/ss-ss-repertoire3:
```

§ 5.11 Suppression de répertoires : `rmdir`

(en anglais *remove directory*)

Syntaxe : `rmdir répertoires`

```
% cp /etc/motd z/toto
% rmdir z
rmdir: z: Directory not empty
```

Pas d'options dignes d'intérêt.

§ 5.12 Positionnement et position dans les répertoires : cd, pwd

Changement de répertoire courant : `cd répertoire`

(en anglais *change directory*)

Affichage du répertoire courant : `pwd`

(en anglais *present working directory*)

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

§ 5.13 Comptage de lignes dans un fichier : wc

(en anglais *word count*)

Syntaxe : `wc [option] fichiers`

Quelques options intéressantes :

"`-c`" : nombre de caractères uniquement

"`-w`" : nombre de mots uniquement

"`-l`" : nombre de lignes uniquement

Par exemple :

```
% wc fichier
       3      16      82 fichier
% wc -l fichier
       3 fichier
```

§5.14 Comparaison de 2 fichiers : diff

(en anglais *difference*)

Pour réaliser la comparaison ligne à ligne du fichier texte `fichier2` par rapport au fichier texte `fichier1` :

```
diff [-c] fichier1 fichier2
```

```
% diff fichier1 fichier2
1c1
< Deux fotes d'ortographe dans cette phrase.
---
> Deux fautes d'orthographe dans cette phrase.

% diff -c fichier1 fichier2
*** fichier1  Sun Sep  9 19:06:13 2001
--- fichier2  Sun Sep  9 19:06:24 2001
***************
*** 1 ****
! Deux fotes d'ortographe dans cette phrase.
--- 1 ----
! Deux fautes d'orthographe dans cette phrase.
```

§ 5.15 Extraction des premières lignes de fichiers : `head`

(en anglais *head*)

Syntaxe : `head [-N] fichiers`

◇ Exemple 1

```
% 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,
```

◇ Exemple 2

```
% head -2 fichier1 fichier2
==> fichier1 <==
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.

==> fichier2 <==
moteur;ferrari;30
moteur;porsche;epuise
```

§ 5.16 Extraction des dernières lignes de fichiers : `tail`

(en anglais *tail*)

Syntaxes : `tail [-N] fichiers`

ou `tail [+N] fichiers`

◇ Exemple 1

```
% 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 2

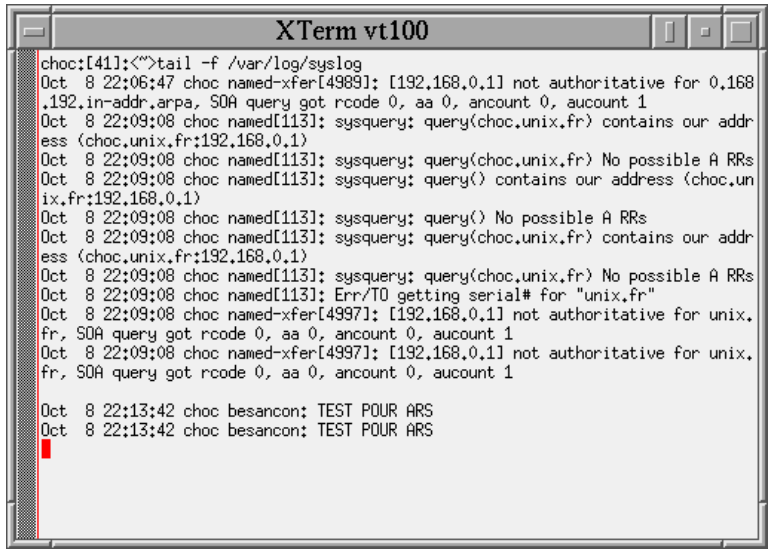
```
% tail -2 fichier1 fichier2
==> fichier1 <==
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.

==> fichier2 <==
moteur;ford;40
moteur;skoda;epuise
```

◇ Exemple 3

```
% 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.
```

Affichage des dernières lignes en temps réel :

Syntaxe : `tail -f fichier`



§ 5.17 Extraction de colonnes : `cut`

(en anglais *cut*)

Syntaxe : `cut [options] fichiers`

Plusieurs modes d'utilisation :

– 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
```

§5.18 Collage de colonnes : `paste`

(en anglais *paste*)

Syntaxe : `paste [options] fichiers`

Collage simple des colonnes de deux fichiers (TAB comme séparateur de colonnes après collage) :

```
% cat fichier1
AAA
BBB
% cat fichier2
aaaa
bbbb
cccc
% paste fichier1 fichier2
AAA     aaaa
BBB     bbbb
        cccc
```

§ 5.19 Tri d'un fichier : `sort`

(en anglais *sort*)

Syntaxe : `sort [options] fichiers`

Quelques options :

– option `-n` : tri numérique
– option `-r` : tri par ordre décroissant
– option `-t` : permet de spécifier le séparateur de mots
– option `-k` : spécifie la colonne sur laquelle trier

◇ Exemple 1

```
% cat toto
arbre
12
ascenceur
2
ordinateur
```

```
% sort toto
12
2
arbre
ascenceur
ordinateur
```

◇ Exemple 2

```
% cat toto
12
2
3
33
22
```

```
% sort toto
12
2
22
3
33
```

```
% sort -n toto
2
3
12
22
33
```

```
% sort -rn toto
33
22
12
3
2
```

◇ Exemple 3

```
% cat toto
or:100000
argent:40000
bois:5
```

```
% sort toto
argent:40000
bois:5
or:100000
```

```
% sort -t: -k 2 toto
or:100000
argent:40000
bois:5
```

```
% sort -t: -k 2 -n toto
bois:5
argent:40000
or:100000
```

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

(en anglais *uniq*)

Syntaxe : `uniq [options] fichier`

Quelques options :

– option `-c` : précéde chaque ligne du résultat du nombre d'occurences de cette ligne dans le fichier original

◇ Exemple 1

```
% cat toto
Ceci est un test.
Ceci est un test.
TEST
unix
TEST
TEST
```

```
% uniq toto
Ceci est un test.
TEST
unix
TEST
```

◇ Exemple 2

```
% cat toto
Ceci est un test.
Ceci est un test.
TEST
unix
TEST
TEST
logiciel
```

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

§ 5.21 Création d'un fichier vide : `touch`

(en anglais *touch*)

Syntaxe : `touch fichiers`

ATTENTION : cette commande ne permet cela que si les fichiers mentionnés n'existent pas déjà !

◇ Exemple

```
% ls -l toto
toto: No such file or directory

% touch toto

% ls -l toto
-rw-r--r--   1 besancon adm           0 Sep 27 12:58 toto
```

§ 5.22 Modification des dates d'un fichier : `touch`

(en anglais *touch*)

Syntaxe : `touch [options] [-t time] fichier`

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 toto
-rw-r--r--   1 besancon adm           49 Sep 27 13:07 toto

% touch -m -t 199901012233 toto
% ls -l toto
-rw-r--r--   1 besancon adm           49 Jan  1  1999 toto

% touch -m -t 09012233 toto
% ls -l toto
-rw-r--r--   1 besancon adm           49 Sep  1 22:33 toto
```

§ 5.23 Création de fichiers temporaires : /tmp

(en anglais *temporary*)

Le répertoire /tmp sert à stocker des fichiers temporaires.

C'est l'équivalent de \windows\temp sur Microsoft Windows.

Ses droits d'accès :

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

c'est-à-dire 1777 exprimé en octal :

– signification de 777 : tout le monde sur la machine peut créer, modifier, effacer des fichiers

– signification de 1000 : un utilisateur ne peut effacer que les fichiers qui lui appartiennent

§ 5.24 Manipulation des noms de fichiers : `basename`

(en anglais *base of name*)

Syntaxe : `basename fichier [suffixe]`

Utilisation principale : supprimer d'un nom de fichier un préfixe :

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

◇ Exemple

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 dans les chapitres ultérieurs)

§5.25 Liens sur fichiers : `ln`, `ln -s`

(en anglais *link*)

A chaque fichier peuvent être associés plusieurs noms.

Chaque nom est un **lien**.

Il y a un compteur de liens pour chaque fichier :

– incrémenté lors de la création d'un lien

– décrémenté lors de la suppression d'un lien

– le contenu d'un fichier est détruit lorsque le dernier lien est supprimé

Deux types de liens :

– lien **hard** ; pas d'équivalent dans les mondes Windows / Macintosh

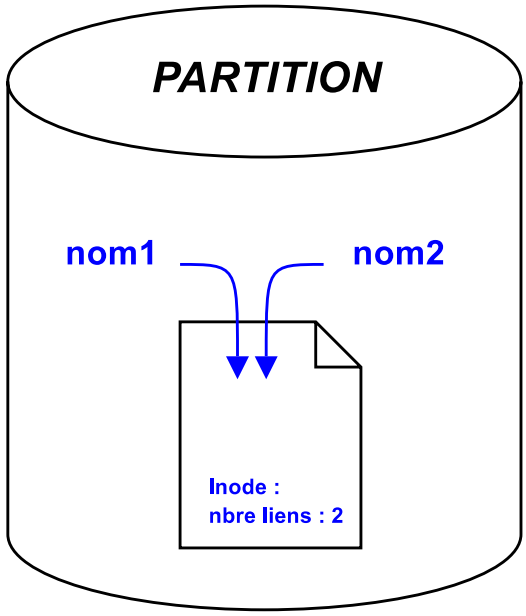
– lien **symbolique** ;
  équivalent des « raccourcis vers » du monde Windows ou des « alias » du monde Macintosh

- lien **hard** : limité au sein d'un même disque dur, où est garantie l'unicité d'un inode («matricule de fichier»)

  La commande à utiliser est : `ln original synonyme`

  Suppression par `rm`

```
% ls -l fichier1
-rw-r--r--  1 besancon  software  9919 Oct 17 18:25 fichier1

% ln fichier1 fichier2

% ls -l fichier1 fichier2
-rw-r--r--  2 besancon  software  9919 Oct 17 18:25 fichier1
-rw-r--r--  2 besancon  software  9919 Oct 17 18:25 fichier2

% ls -li fichier1 fichier2
689357 -rw-r--r--  2 besancon  software  9919 Oct 17 18:25 fichier1
689357 -rw-r--r--  2 besancon  software  9919 Oct 17 18:25 fichier2

% rm fichier1

% ls -li fichier2
689357 -rw-r--r--  1 besancon  software  9919 Oct 17 18:25 fichier2
```


PARTITION
nom1
nom2
Inode :
nbre liens : 2

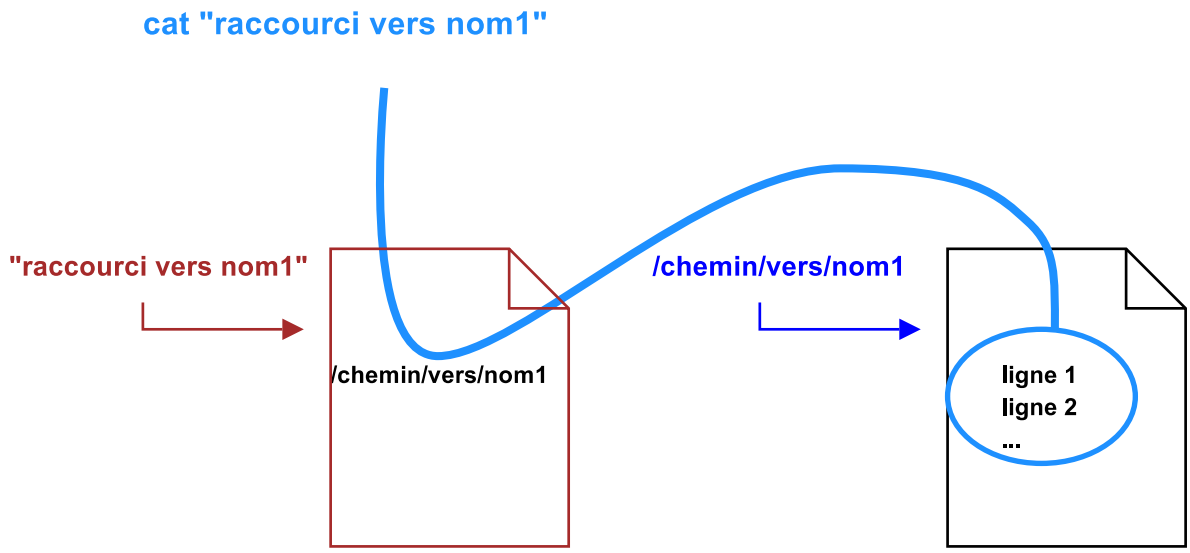
- lien **symbolique** : non limité à un disque parce qu'utilisant le nom d'un fichier et non pas son «matricule»

  En fait, c'est un fichier contenant le nom du fichier source.

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

  Suppression par `rm`

```
% ls -l fichier1
-rw-r--r--  1 besancon  software  9919 Oct 17 18:25 fichier1
% ln -s fichier1 fichier2

% ls -liF fichier1 fichier2
689357 -rw-r--r--  1 besancon  software  9919 Oct 17 18:25 fichier1
689358 lrwxr-xr-x  1 besancon  software     8 Oct 17 18:26 fichier2@ -> fichier1

% ls -lL fichier1 fichier2
-rw-r--r--  1 besancon  software  9919 Oct 17 18:25 fichier1
-rw-r--r--  1 besancon  software  9919 Oct 17 18:25 fichier2

% rm fichier1
% ls -liL fichier2
689358 lrwxr-xr-x  1 besancon  software     8 Oct 17 18:26 fichier2@ -> fichier1

% cat fichier2
cat: fichier2: No such file or directory
```


cat "raccourci vers nom1"
"raccourci vers nom1"
/chemin/vers/nom1
/chemin/vers/nom1
ligne 1
ligne 2
...

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 fichier1 fichier2
-rw-r--r--  1 besancon  software  9919 Oct 17 18:25 fichier1
lrwxr-xr-x  1 besancon  software     8 Oct 17 18:26 fichier2 -> fichier1

% chmod 600 fichier2
% ls -l fichier2
lrwxr-xr-x  1 besancon  software     8 Oct 17 18:26 fichier2 -> fichier1

% ls -l fichier1
-rw-------  1 besancon  software  9919 Oct 17 18:25 fichier1

% ls -lL fichier1 fichier2
-rw-------  1 besancon  software  9919 Oct 17 18:25 fichier1
-rw-------  1 besancon  software  9919 Oct 17 18:25 fichier2
```

§5.26 Nature d'un fichier : `file`

(en anglais *file*)

Syntaxe : `file fichier`

Cette commande permet d'intuiter à quelle application est lié le fichier. Elle s'appuie pour rendre un avis sur la reconnaissance de motifs connus dans le contenu du fichier (pour voir les motifs, se reporter au fichier `/etc/magic`).

Version améliorée de la commande :
`ftp://ftp.astron.com/pub/file/file-3.37.tar.gz`

Quelques exemples non exhaustifs :

```
% ls -l
total 720110
-rw-------   1 besancon adm        23185 Dec 18 18:41 C3866435d01
-rw-------   1 besancon adm      357515264 Dec 18 18:33 DF0D61DCd01
-rw-------   1 besancon adm      2306816 Dec 18 18:41 _CACHE_001_
-rw-------   1 besancon adm      2670592 Dec 18 18:41 _CACHE_002_
-rw-------   1 besancon adm      5816320 Dec 18 18:41 _CACHE_003_
-rw-------   1 besancon adm       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_003_: 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
```

§ 5.27 Commande de traduction de caractères : `tr`

(en anglais *translate*)

Commande de base sur tous les Unix.

Syntaxe : `tr [options] jeu1 jeu2`
où `jeu1` désigne le jeu de caractères à remplacer à raison de un pour un par le jeu de caractères `jeu2`.

Soit le fichier contenant les lignes suivantes :

```
toto
cheval
```

◇ Exemple 1

On veut convertir les lettres o en lettres e :

```
% tr 'o' 'e' < fichier
tete
cheval
```

◇ Exemple 2

On veut convertir les lettres minuscules en lettres majuscules :

```
% tr '[a-z]' '[A-Z]' < fichier
TOTO
CHEVAL
```

§5.28 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	`mcopy fichier a:`
Transfert de la disquette vers Unix	`mcopy a:fichier .`
Affichage du contenu de la disquette	`mdir a:`

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

Chapitre 6 : Editeurs de texte Unix

§ 6.1 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 :

– Le seul éditeur de texte standard sous Unix : `vi`
  (cf. `http://www.math.fu-berlin.de/~guckes/vi/` pour de la doc)
– `emacs`. Très puissant, complexe à maîtriser, simple une fois qu'on sait s'en servir. Cf `http://www.emacs.org`.
– `xemacs`. Variante plus graphique d'emacs. Cf `http://www.xemacs.org`.
– `nedit`. Simple et intuitif grâce à ses menus. Cf `http://www.nedit.org`

§ 6.2 Editeur de fichier texte : `vi`

(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 posséde deux modes de fonctionnement :

– un mode de saisie de commandes à appliquer au texte

– un mode de saisie du texte

◇ Commande passant en mode édition

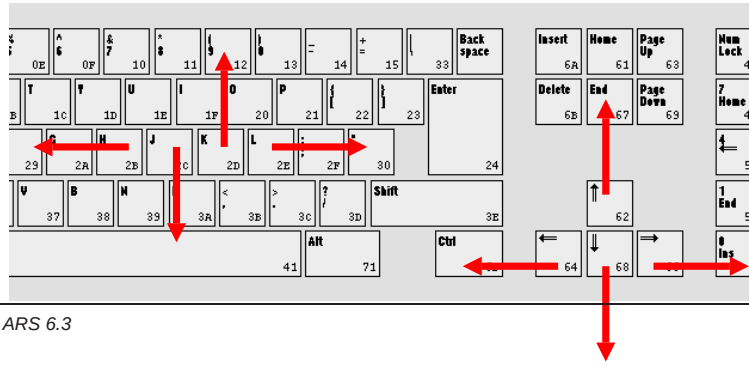
Séquence	**Action**
`i`	Insérer à la position courante du curseur
`a`	Insérer à la position suivante du curseur
`I`	Insérer en début de ligne
`A`	Insérer en fin de ligne
`o`	Ouvrir une nouvelle ligne en dessous du curseur
`O`	Ouvrir une nouvelle ligne au dessus du curseur
`cw`	Changer un mot
`c$`	Changer jusqu'à la fin de ligne

◇ Sortie du mode édition

On passe du mode saisie de texte au mode commande 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
nombre G	Aller à la ligne «nombre»
CTRL G	Affiche le numéro de la ligne courante

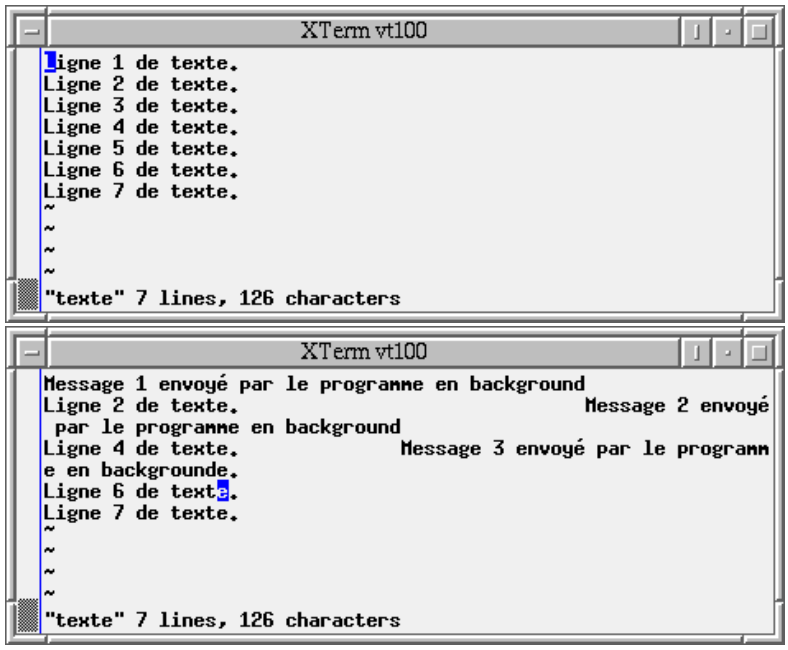


◇ Commandes principales

Séquence	**Action**
`x`	Détruire le caractère sous le curseur
`r`	Remplacer le caractère sous le curseur
`dd`	Effacer la ligne courante
`dw`	Effacer le mot sous le curseur
`.`	Répéter la dernière commande
`J`	Joindre la ligne suivante avec la ligne courante
`/toto`	Rechercher toto dans le texte
`n`	Répéter la dernière recherche
`u`	Annulation de la dernière commande (undo)
CTRL `L`	Rafraichir l'écran

Exemple d'utilisation du CTRL L :

Supprimer les affichages parasites par exemple de programmes en tâche de fond :



◇ Sauvegarde / Sortie de vi

Séquence	**Action**
`:w`	Sauver le fichier édité
`:w toto`	Sauver dans le fichier toto
`:q`	Quitter vi
`:q!`	Quitter vi sans sauvegarder la moindre chose
`:wq`	Sauver puis quitter vi
`:e toto`	Editer maintenant le fichier toto
`:r toto`	Importer le contenu du fichier toto

◇ Commandes de copier/coller

Séquence	**Action**
`yy`	Copier la ligne courante dans la mémoire copier/coller
`p`	Coller dans le texte le contenu de la mémoire précédente
nombre `yy`	Copier «nombre» lignes dans la mémoire copier/coller

◇ Commandes de substitution

Séquence	**Action**
`:s/toto/titi/`	Sur la ligne du curseur, remplacer le premier mot «toto» par «titi»
`:s/toto/titi/g`	Sur la ligne du curseur, remplacer tous les mots «toto» par «titi»
`:1,$s/toto/titi/`	De la ligne 1 à la dernière ligne ($), remplacer le premier mot «toto» par «titi»
`:1,$s/toto/titi/g`	De la ligne 1 à la dernière ligne ($), remplacer tous les mots «toto» par «titi»

Autres exemples de séquences de substitution :

1. La séquence «`:1,$s/toto/titi/g`» remplace de la première ligne à la dernière ligne chaque mot `/toto` par `titi`.
2. La séquence «`:%s/toto/titi/g`» remplace de la première ligne à la dernière ligne chaque mot `/toto` par `titi`.

   ⇒ **On peut employer `%` à la place de `1,$`.**

3. La séquence «`:1,$s/toto//g`» remplace de la première ligne à la dernière ligne chaque mot `toto` par rien du tout, c'est-à-dire que l'on supprime de la première ligne à la dernière ligne chaque mot `toto`

4. La séquence «`:1,$s/\/toto/titi/g`» remplace de la première ligne à la dernière ligne chaque mot `/toto` par `titi`.
5. La séquence «`:1,$s;/toto;titi;g`» remplace de la première ligne à la dernière ligne chaque mot `/toto` par `titi`.

   ⇒ **On peut employer d'autres caractères de séparation que le `/`.**

6. La séquence «`:1,.s/toto/titi/g`» remplace de la première ligne à la ligne courante (désignée par **.**) chaque mot `toto` par `titi`.

7. La séquence «`:.,$s/toto/titi/g`» remplace de la ligne courante (désignée par **.**) jusqu'à la dernière ligne (désignée par **$**) chaque mot `toto` par `titi`.

8. La séquence «`:.,.+3s/toto/titi/g`» remplace de la ligne courante (désignée par **.**) à 3 lignes plus bas (désignée par **.+3**) chaque mot `toto` par `titi`.

9. La séquence «`:.-3,.s/toto/titi/g`» remplace de 3 lignes plus haut que la ligne courante (**.-3**) à la ligne courante chaque mot `toto` par `titi`.

◇ Principales options

Séquence	**Action**
`:set all`	Afficher toutes les options possibles
`:set` **`opt`**	Positionner l'option « opt » à vrai
`:set no`**`opt`**	Positionner l'option « opt » à faux
`:set nu`	Afficher les numéros de ligne
`:set nonu`	ne pas afficher les numéros de ligne

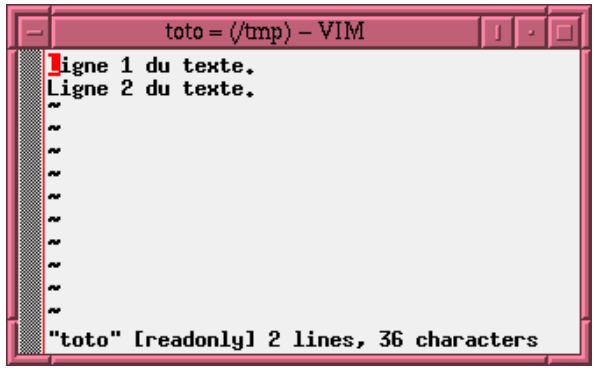
◇ Divers

En cas de plantage de `vi`, utiliser la commande `vi -r exemple.txt` pour essayer de récupérer ce qui est récupérable.

Pour consulter un fichier sans le modifier, faire `vi -R exemple.txt` (ne pas confondre avec au dessus).

§6.3 Editeur de fichier texte : `view`

La commande `view` lance `vi` en mode readonly.



On a donc : `view exemple.txt` ≡ `vi -R exemple.txt`

Commande très pratique.

Chapitre 7 : Attributs des objets Unix

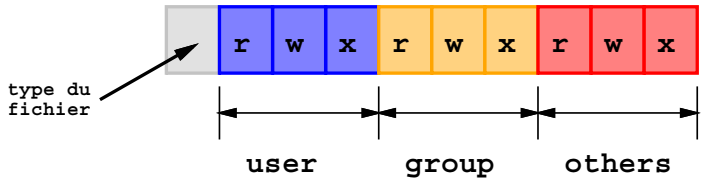
§ 7.1 Définition des droits d'accès d'un objet Unix

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

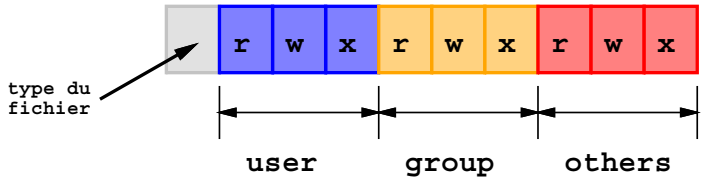
Les droits d'accès des objets se voient grâce à la commande `ls -l` :

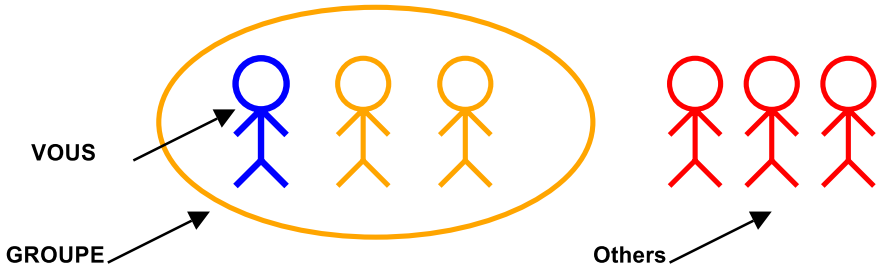
```
% ls -l
total 16
-rw-r--r--  1 besancon    15524 Sep 15 15:17 toto
```

Plus particulièrement, les droits sont indiqués par les 10 premiers caractères de chaque ligne affichée par `ls -l` :

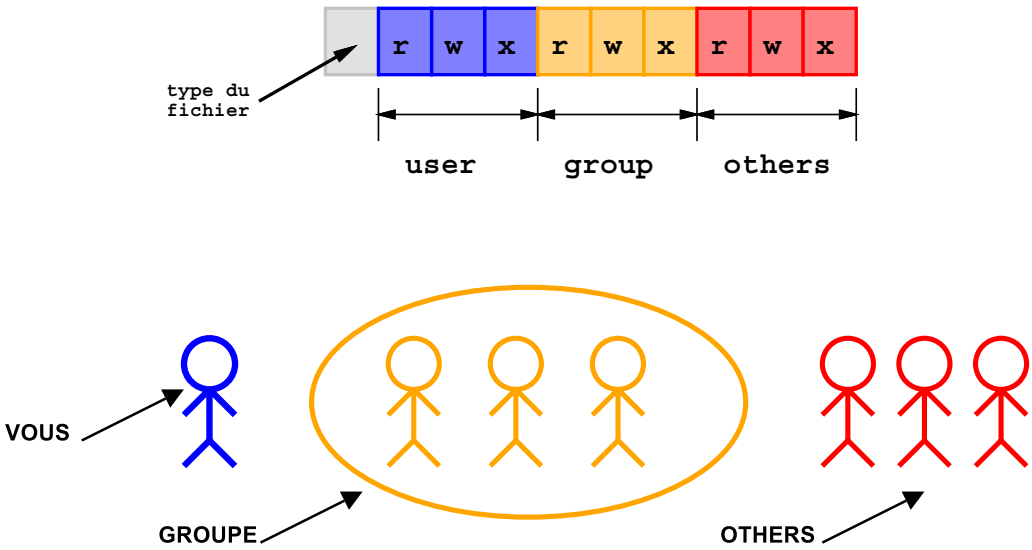


Cas le plus courant :





Cas moins courant mais possible :



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

– droits du propriétaire (u $\equiv$ user)
– droits des membres du groupe (g $\equiv$ group)
– droits des autres utilisateurs (o $\equiv$ others)

Il existe trois types de permissions :

– droit en lecture (r $\equiv$ read)
– droit en écriture (w $\equiv$ write)
– droit en exécution (x $\equiv$ execute access)

§ 7.2 Changements des droits d'accès d'un objet : `chmod`

(en anglais *change modes*)

Syntaxe : `chmod [options] modes objets`

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` ≡ 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 fichier
-rw-r--r--  1 besancon  software  249 Sep 20 22:43 fichier

% chmod g+w fichier
% ls -lg fichier
-rw-rw-r--  1 besancon  software  249 Sep 20 22:43 fichier

% chmod o=rw fichier
% ls -lg fichier
-rw-rw-rw-  1 besancon  software  249 Sep 20 22:43 fichier

% chmod 640 fichier
% ls -lg fichier
-rw-r-----  1 besancon  software  249 Sep 20 22:43 fichier
```

§ 7.3 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 [-S]`

```
% umask
022

%umask -S
u=rwx,g=rx,o=rx
```

– 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` ≡ `umask 022`

On veut par défaut les droits `rw- r-- r--` ≡ `umask 133`

On veut par défaut les droits `rw- --- ---` ≡ `umask 177`

Le mode paranoiaque ≡ droits par défaut `rwx --- ---` ≡ `umask 077`

◇ Exemple 1

```
% umask
022

% vi prog.c
% ls -l prog.c
-rw-r---w-   1 besancon adm            0 Oct 12 14:45 prog.c

% gcc prog.c -o prog
% ls -l prog
-rwxr-xr-x   1 besancon adm         6076 Oct 12 14:45 prog
```

◇ Exemple 2

```
% umask 027

% vi prog.c
% ls -l prog.c
-rw-r-----   1 besancon adm            0 Oct 12 14:45 prog.c

% gcc prog.c -o prog
% ls -l prog
-rwxr-x---   1 besancon adm         6076 Oct 12 14:45 prog
```

◇ Exemple 3

```
% umask 000

% vi prog.c
% ls -l prog.c
-rw-rw-rw-   1 besancon adm            0 Oct 12 14:45 prog.c

% gcc prog.c -o prog
% ls -l prog
-rwxrwxrwx   1 besancon adm         6076 Oct 12 14:45 prog
```

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

§ 7.4 Régler son 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` »

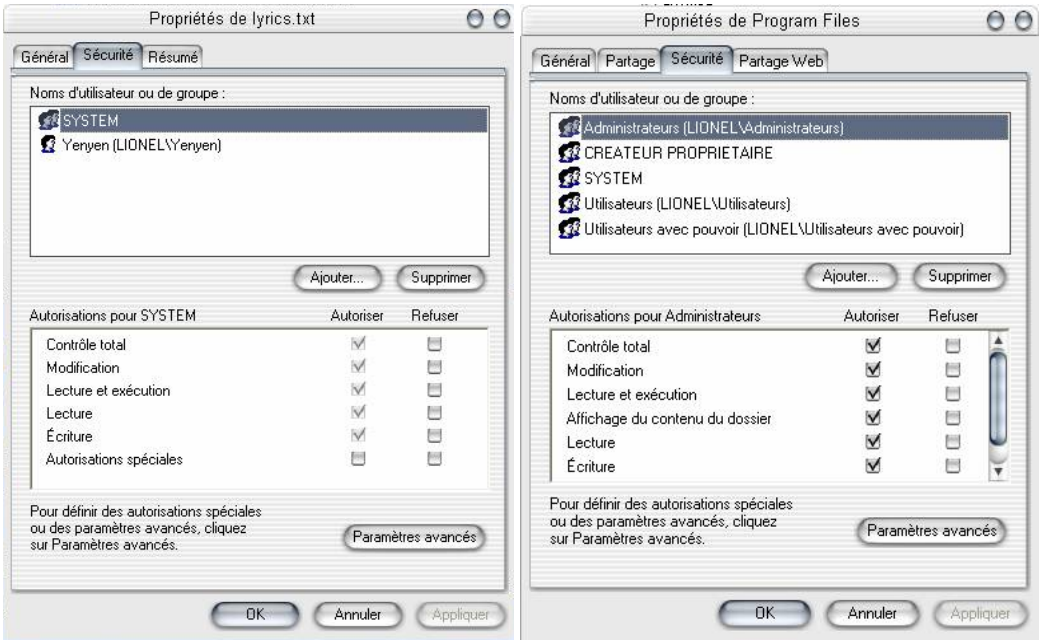
```
umask 022
```

§ 7.5 Définition des droits d'accès d'un objet sous WINDOWS

A completer...

Commandes de manipulations des droits NTFS :

– `cacls`, intégré dans W2K
– `xcacls`, à télécharger
  `http://download.microsoft.com/download/win2000platform/Update/1.00.0.1/NT5/EN-US/xcacls_setup.exe!`


Propriétés de lyrics.txt
Général
Sécurité
Résumé
Noms d'utilisateur ou de groupe :
SYSTEM
Yenyen (LIONEL\Yenyen)
Ajouter...
Supprimer
Autorisations pour SYSTEM
Autoriser
Refuser
Contrôle total
Modification
Lecture et exécution
Lecture
Écriture
Autorisations spéciales
Pour définir des autorisations spéciales ou des paramètres avancés, cliquez sur Paramètres avancés.
Paramètres avancés
OK
Annuler
Appliquer
Propriétés de Program Files
Général
Partage
Sécurité
Partage Web
Noms d'utilisateur ou de groupe :
Administrateurs (LIONEL\Administrateurs)
CREATEUR PROPRIETAIRE
SYSTEM
Utilisateurs (LIONEL\Utilisateurs)
Utilisateurs avec pouvoir (LIONEL\Utilisateurs avec pouvoir)
Ajouter...
Supprimer
Autorisations pour Administrateurs
Autoriser
Refuser
Contrôle total
Modification
Lecture et exécution
Affichage du contenu du dossier
Lecture
Écriture
Pour définir des autorisations spéciales ou des paramètres avancés, cliquez sur Paramètres avancés.
Paramètres avancés
OK
Annuler
Appliquer

§ 7.6 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 -lgF a b
-rwxr-xr-x   1 besancon  software     249 Sep 20 22:43 a
-rwxr-xr-x   1 besancon  software     249 Sep 20 22:43 b
% chmod u+s a
% chmod 4711 b
% ls -lgF a b
-rwsr-xr-x   1 besancon  software     249 Sep 20 22:43 a
-rws--x--x   1 besancon  software     249 Sep 20 22:43 b
```

Attention à l'affichage du bit setuid !

Classiquement :

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

Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon adm            0 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 adm            0 Jan 24 20:21 exemple.txt
% chmod 4644 exemple.txt
% ls -l exemple.txt
-rwSr--r--   1 besancon adm            0 Jan 24 20:21 exemple.txt
```

Bref :

– affichage « S » ≡ « bit 04000 seul »

– affichage « s » ≡ « bit x + bit S »

§ 7.7 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 a b
-rwxr-xr-x   1 besancon  software     249 Sep 20 22:43 a
-rwxr-xr-x   1 besancon  software     249 Sep 20 22:43 b
% chmod g+s a
% chmod 2711 b
% ls -lgF a b
-rwxr-sr-x   1 besancon  software     249 Sep 20 22:43 a
-rwx--s--x   1 besancon  software     249 Sep 20 22:43 b
```

Attention à l'affichage du bit setgid !

Classiquement :

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

Moins classiquement :

```
% touch exemple.txt
% ls -l exemple.txt
-rw-r--r--   1 besancon adm            0 Jan 24 20:21 exemple.txt
% chmod g+s exemple.txt
chmod: WARNING: exemple.txt: Execute permission required for set-ID on execution
% ls -l exemple.txt
-rw-r--r--   1 besancon adm            0 Jan 24 20:21 exemple.txt
% chmod 2644 exemple.txt
% ls -l exemple.txt
-rw-r-lr--   1 besancon adm            0 Jan 24 20:21 exemple.txt
```

Bref :

– affichage « l » ≡ « bit 02000 seul »

– affichage « s » ≡ « bit x + bit l »

§ 7.8 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 sur le répertoire système de stockage des fichiers temporaires

```
% ls -lgd /tmp
drwxrwxrwt  11 amavis   amavis      2580 Aug  2 15:40 /tmp
% cd /tmp
% mkdir tmp2
% ls -lgd tmp2
drwxr-xr-x   2 besancon adm          117 Aug  2 15:40 tmp2
% chmod 1777 tmp2
% ls -lgd tmp2
drwxrwxrwt   2 besancon adm          117 Aug  2 15:40 tmp2
% chmod 777 tmp2
% ls -lgd tmp2
drwxrwxrwx   2 besancon adm          117 Aug  2 15:40 tmp2
```

Historiquement : le sticky bit positionné sur un exécutable le chargeait en mémoire virtuelle et ne l'effacait 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 /tmp/exemple.dir
% ls -ld /tmp/exemple.dir
drwxr-xr-x   2 besancon adm          117 Jan 25 11:09 /tmp/exemple.dir
% chmod 1777 /tmp/exemple.dir
% ls -lgd /tmp/exemple.dir
drwxrwxrwt   2 besancon adm          117 Jan 25 11:09 /tmp/exemple.dir
```

Moins classiquement :

```
% mkdir -p /tmp/exemple.dir
% ls -ld /tmp/exemple.dir
drwxr-xr-x   2 besancon adm          117 Jan 25 11:09 /tmp/exemple.dir
% chmod 1700 /tmp/exemple.dir
% ls -lgd /tmp/exemple.dir
drwx-----T   2 besancon adm          117 Jan 25 11:12 /tmp/exemple.dir
```

Bref :

– affichage « T » ≡ « bit 01000 seul »

– affichage « t » ≡ « bit x + bit T »

§ 7.9 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* (≡ modification time)

– date de dernier accès dite *atime* (≡ access time)

– date de dernière modification des attributs dite *ctime* (≡ change time)

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 mdates

– « `ls -ltu` » : classement par ordre chronologique décroissant des atimes

– « `ls -ltc` » : classement par ordre chronologique décroissant des ctimes

§ 7.10 Consultation de l'horloge : `date`

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

§ 7.11 Modification des dates d'un objet : `touch`

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 toto
-rw-r--r--   1 besancon adm           49 Sep 27 13:07 toto

% touch -m -t 199901012233 toto
% ls -l toto
-rw-r--r--   1 besancon adm           49 Jan  1  1999 toto

% touch -m -t 09012233 toto
% ls -l toto
-rw-r--r--   1 besancon adm           49 Sep  1 22:33 toto
```

Chapitre 8 : Expressions régulières et commandes Unix associées

§ 8.1 Regular expressions (regexps)

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 chaines de caractères.

Un exemple de motif :

«*les lignes commencant par la lettre a*»

La regexp est la traduction en langage Unix du motif, ce qui permettra de le reconnaitre 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 des écritures :

`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
`^`	désigne la contrainte d'être en début de ligne
`$`	désigne la contrainte d'être en fin de ligne
`{m,n}`	indique que la contrainte mentionnée juste avant doit être satisfaite entre m et n 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)

ATTENTION : principes FONDAMENTAUX des regexps

– on analyse chaque ligne indépendemment de la précédente et de la suivante (sauf mention contraire mais rare)
– 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 chaine 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

Application des regexps :

– sélectionner des lignes de fichiers texte qui satisfont la regexp
  $\Rightarrow$ 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
  $\Rightarrow$ 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 muet une contrainte pour satisfaire la regexp globalement. . .

§ 8.2 Recherche de regexp dans un fichier : `grep`

Syntaxe : `grep [options] regexp fichiers`

Quelques options intéressantes :

- "`-i`" : pas de différenciation entre lettres minuscules et majuscules
- "`-n`" : affichage des numéros de ligne
- "`-l`" : n'affiche que les noms de fichiers
- "`-v`" : affichage des lignes ne contenant pas la chaîne précisée

(grep ≡ `g/re/p` dans vi)

◇ 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 fichier
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 fichier
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 fichier
3:Mais là on ne met pas la chaîne de l'exemple.
```

◇ Exemple 4

Soit le fichier :

```
19101259;18111971;LUKOVIC               ;DRAGICA
10024113;09051982;LUMINET               ;AMANDE
10009175;09051980;LUND                  ;BRIANA NICOLE
10004978;21041976;LUNDQUIST             ;ANNA
10023887;24081983;MEKDJIAN              ;SARAH
19404107;19121975;MEKOUAR               ;DRISS
19803654;13111975;MEKOUAR               ;MOUNA
10003714;22121965;MILON                 ;NICOLAS
19603414;12101970;MILOSAVLJEVIC         ;DRAGANA
19504774;03031977;MILOUDI               ;HANAN
```

On cherche les prénoms qui commencent par DR :

```
% grep ';DR[^;]*$' fichier
19101259;18111971;LUKOVIC               ;DRAGICA
19404107;19121975;MEKOUAR               ;DRISS
19603414;12101970;MILOSAVLJEVIC         ;DRAGANA
```

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 l'Unix de SUN, Solaris :

```
egrep [options] 'regexp' fichiers
```

– sur LINUX :

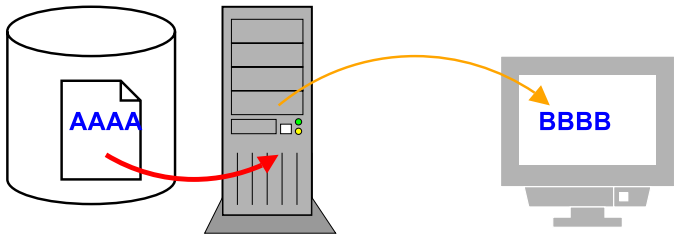
```
grep -E [options] 'regexp' fichiers
```

§8.3 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 fichier`

NB : sed modifie un flux, pas un contenu de fichier : après l'application de la commande, le fichier appliqué reste 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 la forme :

```
[adresse1[,adresse2]]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
– une regexp que l'on écrit entre `/`, soit `/regexp/`

Les fonctions proposées dans sed :

– suppression de ligne : `d`

– affichage de ligne : `p`

– substitution au sein des lignes : `s/regexp/remplacement/modifiers`

◇ Exemple 1

Soit le fichier :

```
% cat fichier
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' fichier
Ceci est la troisieme ligne.
Ceci est la quatrieme ligne.
Ceci est la cinquieme ligne.
```

◇ Exemple 2

Soit le fichier :

```
% cat fichier
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' fichier
Ceci est la premiere ligne.
Ceci est la deuxieme ligne.
```

◇ Exemple 3

Soit le fichier :

```
% cat fichier
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/' fichier
moteur;ferrari;30
moteur;porsche;0
carrosserie;porsche;0
moteur;ford;40
moteur;skoda;0
```

◇ Exemple 4

Soit le fichier :

```
% cat fichier
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/' fichier
moteur;ferrari;30
moteur;porsche;0
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;0
```

◇ Exemple 5

Soit le fichier :

```
% cat fichier
Les courses de chevals se font a Vincennes.
Les chevals ont 4 jambes. Les chevals sont des equides.
```

● On veut remplacer le mot «chevals» du fichier par chevaux **lors de son affichage** :

```
% sed -e 's/chevals/chevaux/' fichier
Les courses de chevaux se font a Vincennes.
Les chevaux ont 4 jambes. Les chevals sont des equides.
```

⇒ Le remplacement n'a pas lieu sur tous les mots «chevals» au sein d'une ligne !

● On veut remplacer le mot «chevals» du fichier par chevaux **lors de son affichage** :

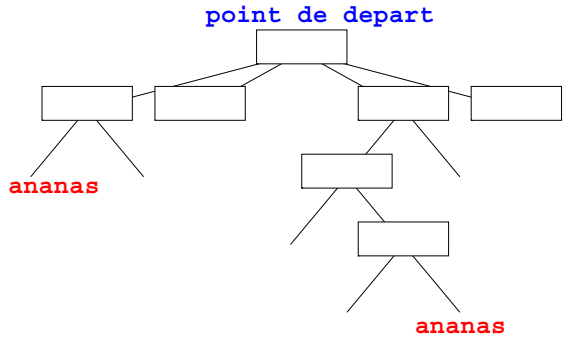
```
% sed -e 's/chevals/chevaux/g' fichier
Les courses de chevaux se font a Vincennes.
Les chevaux ont 4 jambes. Les chevaux sont des equides.
```

⇒ Le remplacement a lieu sur tous les mots «chevals» au sein d'une ligne.

Chapitre 9 : Commande de recherche de fichiers : `find`

§9.1 Recherche de fichiers : `find`

On recherche **à partir des répertoires indiqués** les fichiers répondant aux critéres exprimés par des expressions.



Syntaxe : `find répertoires expressions`

Les expressions indiquent :

– des conditions

– des actions à effectuer

Quelques expressions :

– critère de nom : `-name nom`

– critère de droits d'accès : `-perm permissions`

– critère de type (fichier, répertoire) : `-type type` (d pour directory, f pour file, etc.)

– critère de taille : `-size N`

– critère de date récente : `-newer fichier`

– critère de date : `-atime N`, `-mtime N`, `-ctime N`

– ou logique entre conditions : `condition1 -o condition2`

– et logique entre conditions : `condition1 -a condition2` (en fait le `-a` est facultatif)

– affichage du nom du fichier trouvé : `-print`

– exécution d'une commande : `-exec commande {} \;`

Par exemple :

– Recherche des fichiers d'extension '`.c`' :

```
% find . -name \*.c -print
% find . -name '*.c' -print
```

– Recherche des répertoires :

```
% find . -type d -print
```

– Recherche des fichiers de plus de 1000000 caractères :

```
% find ~ -size +1000000c -print
```

– Recherche de tous les fichiers 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 fichiers 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 toto -print
```

§ 9.2 Confusion courante

Ne pas confondre la commande find et la commande ls

Rechercher un fichier nommé « toto » dans une arborescence :

– OUI : « `find . -name toto -print` »

– NON : « `ls -Rl | grep toto` »

Pourquoi ?

→ La commande `ls` pourrait renvoyer des données parasites.

Dans le répertoire courant, chercher les fichier commencant par « toto » :

– NON : « `find . -name 'toto*' -print` »

– OUI : « `ls toto*` »

Pourquoi ?

→ La commande `find` va faire une recherche récursive et dépasser le niveau du répertoire courant en descendant plus bas.

§ 9.3 Quelques Difficultés

La directive -exec n'est pas très pratique.

Exemple où -exec n'est pas adapté :

Rechercher les fichiers commencant par toto et les renommer en 2003-toto...

→ La directive -exec symbolise le fichier à afficher par { } mais ne permet pas de construire des commandes non basiques :

```
% ls toto*
toto1   toto2

% find . -name 'toto*' -exec echo {} {} \;
./toto1 ./toto1
./toto2 ./toto2

% find . -name 'toto*' -exec echo {} 2003-{} \;
./toto1 2003-{}
./toto2 2003-{}
```

Les fichiers trouvés partent du point de recherche et le mentionnent.

```
% ls toto*
toto1   toto2

% find . -name 'toto*' -print
./toto1
./toto2
```

Attention si vous devez donc retraiter les noms des fichiers trouvés (ici parasitage du « `./` »).

Chapitre 10 : Commandes Unix de gestion de place disque

§ 10.1 Information sur la consommation d'espace disque : `df`

(en anglais *disk filesystems*)

La commande `df -k` vous indiquera le taux de remplissage des disques durs locaux de l'ordinateur ainsi que des disques durs réseau qui stockent vos fichiers.

Par exemple (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
```

§ 10.2 Calcul de la place disque occupée : du

(en anglais *disk usage*)

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).

```
% du -k
372     ./csi
107     ./cru
1793
```

§ 10.3 Compression de fichiers : `compress`, `uncompress`, `zcat`

Syntaxes de quelques commandes de compression ou décompression :

```
compress [options] fichier
uncompress [options] fichier.Z
zcat fichier.Z
```

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

Le fichier décompressé retrouve son nom `fichier`.

◇ Exemples

```
% compress fichier1

% compress -v fichier2
fichier2: Compression: 14.70% -- replaced with fichier2.Z

% ls -l fichier1.Z fichier2.Z
-rw-r--r--   1 besancon adm       97 Aug  3 13:01 fichier1.Z
-rw-r--r--   1 besancon adm       87 Aug  3 13:06 fichier2.Z

% zcat fichier2.Z
moteur;ferrari;30
moteur;porsche;epuise
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;epuise
```

§10.4 Compression de fichiers : gzip, gunzip, gzcat

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

Syntaxes de quelques commandes de compression ou décompression :

```
gzip [options] fichier
gunzip [options] fichier.gz
gzcat fichier.gz
```

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

Le fichier décompressé retrouve son nom `fichier`.

Répandu mais non standard. Cf `ftp://ftp.lip6.fr/pub/gnu/gzip/`

◇ Exemples

```
% gzip fichier1

% gzip -v fichier2
fichier2:                 35.2% -- replaced with fichier2.gz

% ls -l fichier1.gz fichier2.gz
-rw-r--r--   1 besancon adm       84 Aug  3 13:01 fichier1.gz
-rw-r--r--   1 besancon adm       87 Aug  3 13:06 fichier2.gz

% gzcat fichier2.gz
moteur;ferrari;30
moteur;porsche;epuise
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;epuise
```

§ 10.5 Compression de fichiers : bzip2, bunzip2, bzcat

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

Syntaxes de quelques commandes de compression ou décompression :

```
bzip2 [options] fichier
bunzip2 [options] fichier.bz2
bzcat fichier.bz2
```

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

Le fichier décompressé retrouve son nom `fichier`.

Pas encore très répandu. Cf `ftp://ftp.lip6.fr/pub/gnu/bzip2/????`

◇ Exemples

```
% bzip2 fichier1

% bzip2 -v fichier2
fichier2:  1.052:1,  7.608 bits/byte,  4.90% saved, 102 in, 97 out.

% ls -l fichier1.bz2 fichier2.bz2
-rw-r--r--   1 besancon adm      102 Aug  3 13:01 fichier1.bz2
-rw-r--r--   1 besancon adm       97 Aug  3 13:06 fichier2.bz2

% bzcat fichier2.bz2
moteur;ferrari;30
moteur;porsche;epuise
carrosserie;porsche;epuise
moteur;ford;40
moteur;skoda;epuise
```

§ 10.6 Archivage de fichiers/répertoires : `tar`

(en anglais *tape archive*)

La commande `tar` permet d'archiver dans un seul fichier une arborescence. Conjuguée à une commande de compression, cela permet de mettre de côté une arborescence dont on n'a plus besoin. Au passage, on gagne de la place.

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

– Création d'une archive : `tar cvf archive.tar fichiers`
– Affichage du contenu d'une archive : `tar tvf archive.tar`
– Extraction de l'archive complète : `tar xvf archive.tar`
– Extraction d'un fichier précis de l'archive : `tar xvf archive.tar fichier`

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

- Utilisation de `compress` :
  - Création d'une archive : `tar cvZf archive.tar.Z fichiers`
  - Affichage du contenu d'une archive : `tar tvZf archive.tar.Z`
  - Extraction de l'archive complète : `tar xvZf archive.tar.Z`
  - Extraction d'un fichier précis de l'archive : `tar xvZf archive.tar.Z fichier`

- Utilisation de `gzip` :
  - Création d'une archive : `tar cvzf archive.tar.gz fichiers`
  - Affichage du contenu d'une archive : `tar tvzf archive.tar.gz`
  - Extraction de l'archive complète : `tar xvzf archive.tar.gz`
  - Extraction d'un fichier précis de l'archive : `tar xvzf archive.tar.gz fichier`

- Utilisation de `bzip2` :
  - Création d'une archive : `tar cvjf archive.tar.gz fichiers`
  - Affichage du contenu d'une archive : `tar tvjf archive.tar.gz`
  - Extraction de l'archive complète : `tar xvjf archive.tar.gz`
  - Extraction d'un fichier précis de l'archive : `tar xvjf archive.tar.gz fichier`

§ 10.7 Comparaison avec le monde Microsoft Windows

Classiques du monde Microsoft Windows :

– WinZip, `http://www.winzip.com`

– PowerArchiver, `http://www.powerachiver.com`

En fait : Winzip, PowerArchiver $\equiv$ archivage + compression

Monde Microsoft Windows	**Monde Unix**

Winzip, PowerArchiver	`tar` + `compress`
	`tar` + `gzip`
	`tar` + `bzip2`

§ 10.8 Compression de fichiers : zip, unzip

Syntaxes de quelques commandes de compression ou décompression :

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

◇ Exemples

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

% 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
```

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

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

A completer...

URL zip/unzip

Chapitre 11 : Commandes Unix réseau de base

§ 11.1 Impression : `lpr`, `lpq`, `lprm`

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

Une impression nécessite de connaitre 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, le nom de l'imprimante est **hpars**.

§ 11.2 Impression de fichiers texte : a2ps

(en anglais *ascii to postscript*)

Pour convertir du texte au format PostScript.

Nombreuses options.

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

```
% a2ps -P hpars 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`