LA CLASSIFICATION DES FICHIERS LINUX

Dans la présentation de la structure du système de fichiers au paragraphe précédent, nous avons évoqué des fichiers de type répertoire, des fichiers de type ordinaire et des fichiers spéciaux.

La syntaxe d'un nom de fichier n'est pas très stricte. Il est recommandé de limiter le nom d'un fichier à 14 caractères au plus et de n'utiliser que les lettres majuscules ou minuscules (attention, Linux différencie les majuscules des minuscules), les chiffres et quelques autres caractères (le point `.`, le tiret –, le souligné _). Linux autorise jusqu'à 255 caractères pour le nom du fichier. La longueur minimum est de un caractère.

Les caractères spéciaux suivants sont à proscrire absolument :

`\ > < | $ ? & [ ] * ! " ' ( ) ` @ ~ <espace>`

De plus, les utilisateurs ayant des claviers français doivent éviter les caractères accentués. En annexe B, vous trouverez la liste complète des caractères spéciaux à proscrire, car ils ont une signification particulière pour le système.

Le point (`.`) joue un rôle particulier dans le nom d'un fichier. Les fichiers dont les noms commencent par un point (`.`), comme *`.profile`*, sont des fichiers cachés (c'est–à–dire qu'ils n'apparaissent pas dans la liste des fichiers en tapant la commande *`ls`* sans argument).

Le point sert également à suffixer les noms des fichiers. Cette pratique est très recommandée, car elle facilite la gestion des fichiers. Il est vrai qu'il n'existe pas de syntaxe précise ; il existe toutefois un certain nombre de conventions :

`essai.c`	fichier source C
`include.h`	include de C
`essai.o`	fichier binaire objet
`essai.f`	fichier source fortran
`essai.c.old`	convention autorisée mais personnelle (fichier source C, ancienne version).

Certaines commandes de Linux s'appliquent à plusieurs fichiers. Dans ce cas, plutôt que de les énumérer, il est plus commode de les désigner par un nom générique en utilisant des caractères spéciaux, pour remplacer un ou plusieurs caractères dans le nom du fichier (voir paragraphe 7.7).

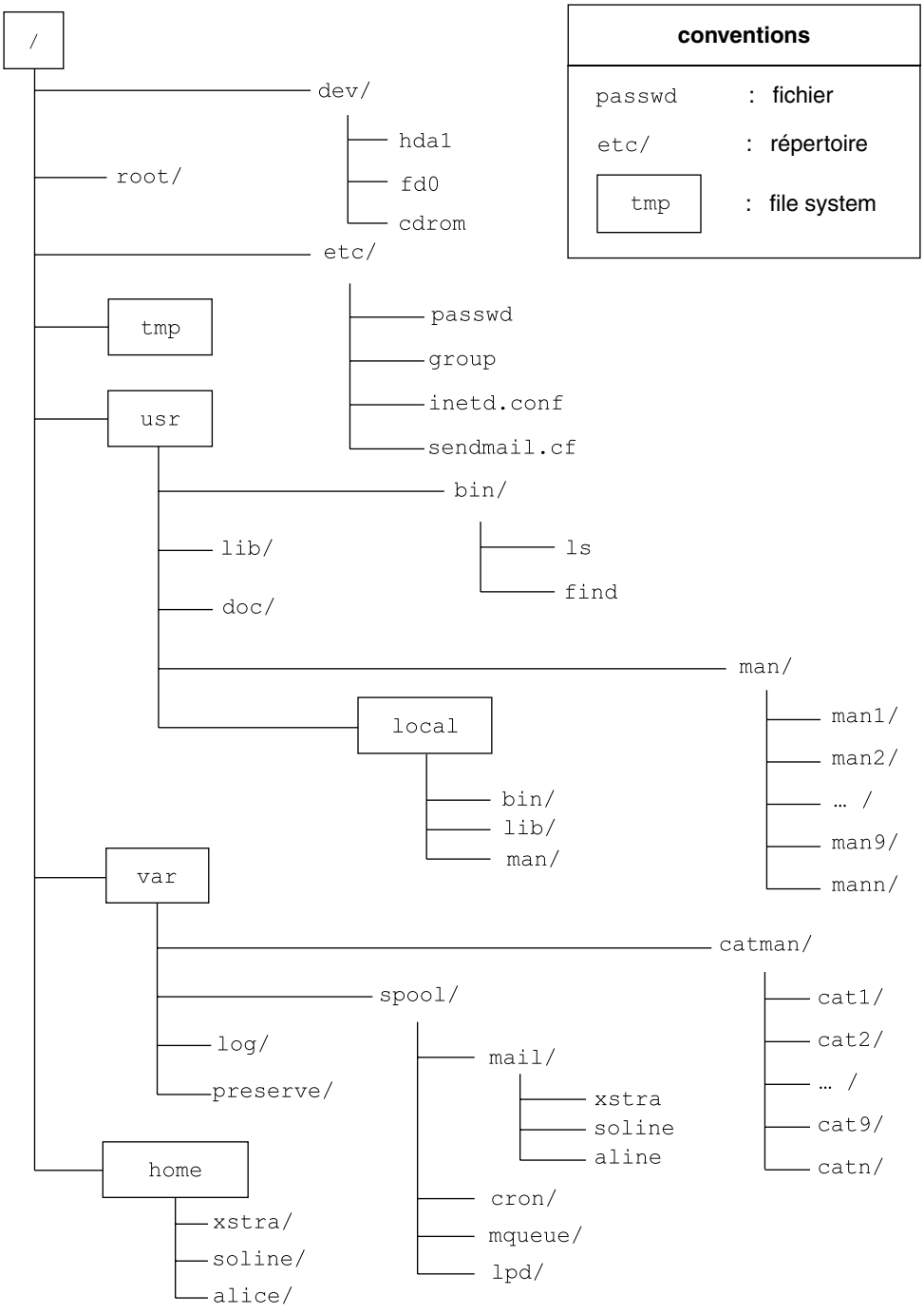

FIGURE 3.1. L'ARBORESCENCE DES FICHIERS LINUX.

Les caractères spéciaux et leur signification

`*`	désigne toute chaîne de 0 à n caractères,
`?`	désigne un caractère quelconque,
`[...]`	désigne un caractère quelconque appartenant à l'ensemble des caractères entre crochets.

Exemples

`fich.*`	désigne tous les fichiers de nom `fich` et ayant un suffixe.
`essai?`	permet d'obtenir tous les fichiers ayant un nom de 6 caractères dont les 5 premiers caractères sont `essai`, le dernier caractère est quelconque.
`[a f]`	désigne n'importe quelle lettre comprise entre `a` et `f`.
`[a z]*`	désigne tous les noms commençant par une lettre minuscule.

3.3 LA DÉSIGNATION DES FICHIERS

Un fichier est repéré par son nom et sa position dans l'arborescence : son chemin d'accès (**pathname**).

La syntaxe de ce chemin d'accès est très précise et peut être décrite des deux manières suivantes :

3.3.1 Le chemin d'accès absolu

Il permet d'accéder à un fichier quelconque dans l'arborescence du système de fichiers. Il est composé d'une suite de noms de répertoires séparés par le caractère `/`. Il commence toujours par le caractère `/` qui désigne le répertoire racine et se termine par le nom du fichier que l'on veut atteindre.

La longueur du chemin d'accès absolu d'un fichier est limitée à 1024 caractères.

Exemples

```
/var/spool/mail/xstra1
/home/xstra/essai
```

3.3.2 Le chemin d'accès relatif

La désignation d'un fichier par son chemin d'accès absolu se révèle rapidement lourde vu le nombre de répertoires intermédiaires à désigner. Tout utilisateur peut se positionner sur n'importe quel répertoire de l'arborescence. Ce répertoire devient courant (**répertoire de travail** ou **current working directory**).

Dès l'entrée en session de l'utilisateur, le système le place dans un répertoire de travail initial (**répertoire d'accueil** ou **home directory**). Ce répertoire a été créé au moment de l'établissement du compte de l'utilisateur. Le système associe alors en

permanence, à tout processus ou tâche, le chemin d'accès complet du répertoire de travail courant de l'utilisateur. Ainsi, l'usager peut désigner un fichier en ne donnant que son chemin d'accès relatif au répertoire de travail courant.

A partir de ce répertoire courant, l'utilisateur construit son propre sous–arbre de répertoires et de fichiers.

Exemple

chemin absolu	`/home/xstra/develop/prog1`
répertoire courant	`/home/xstra`
chemin relatif	`develop/prog1`

Remarque

Tout répertoire contient au moins deux entrées :

`.` § représente le répertoire lui–même.

`..` § représente le répertoire père.

Ces entrées de répertoire ne sont en général pas imprimées par les utilitaires du système. Elles permettent de référencer le répertoire courant sans l'obligation de citer son nom de chemin d'accès absolu ou de référencer avec un chemin d'accès relatif un fichier dans un sous–arbre.

Exemple

répertoire courant	`/home/xstra`
chemin d'accès relatif	`../xstra/develop/prog1`
chemin d'accès absolu équivalent	`/home/xstra/develop/prog1`

Par commodité, nous utiliserons dans la suite de l'ouvrage le terme nom de fichier pour désigner le chemin d'accès à ce fichier.

3.4 LA MANIPULATION DES RÉPERTOIRES

Après l'ouverture de sa session, l'utilisateur se trouve sous le contrôle d'un interpréteur de commandes. Celui–ci est prêt à lire, analyser et éventuellement exécuter les commandes qui lui sont soumises.

Chaque commande se compose d'un ensemble de champs séparés par un ou plusieurs blancs et se termine par une fin de ligne (<return> ou " line feed "). Le premier de ces champs est obligatoirement un nom de commande. Les autres champs définissent des paramètres dont l'interprétation dépend de la commande considérée. Nous reviendrons ultérieurement sur cette syntaxe.

Pour bien organiser son espace de travail, il est souvent utile de grouper ses fichiers par centre d'intérêt en créant des sous–répertoires. Les principales commandes pour gérer les répertoires sont :

pwd Print Working Directory

Affiche le chemin d'accès du répertoire courant. Juste après connexion d'un utilisateur xstra, la commande *pwd* lui précisera son répertoire d'accueil.

Exemple

```
xstra> pwd
/home/xstra
xstra>
```

cd Change Directory

Permet de changer de répertoire de travail.

Exemple

```
xstra> cd .. § Permet de remonter au
            § répertoire père.
xstra> pwd
/home
xstra>
```

Exemple

```
xstra> cd    § Permet de se repositionner
             § sur son répertoire d'accueil.
xstra> pwd
/home/xstra
xstra>
```

mkdir MaKe DIRectory

Crée un nouveau répertoire.

Exemple

xstra vient de se connecter. Il veut se créer un répertoire *perl* à partir du répertoire courant.

```
xstra> mkdir perl
xstra> cd perl
xstra> pwd
/home/xstra/perl
xstra>
```

rmdir ReMove DIRectory

Supprime un répertoire, s'il est vide.

Exemple

L'utilisateur décide de supprimer le répertoire précédemment créé. Ce répertoire est bien vide ; suppression possible.

```
xstra> cd
xstra> rmdir perl
xstra>
```

du Disk Usage

Donne l'occupation disque en bloc [un bloc valant 512 octets ou 1 Kilo–octets (Ko)] des sous–répertoires du répertoire spécifié ou, si aucun répertoire n'est précisé, du répertoire courant (nous reviendrons ultérieurement sur cette commande, en particulier dans le paragraphe 10.2).

L'utilisation de cette commande, et en particulier la capacité d'un bloc, est à vérifier sur votre machine par la commande *man du* ou *info du*.

find FIND

Recherche un fichier à partir du répertoire donné.

Exemples

1) Recherche du fichier *.bash profile* chez l'utilisateur connecté, puis affichage à l'écran de la liste des fichiers.

```
xstra> find . -name .bash_profile -print
```

2) Recherche de tous les fichiers de taille supérieure à 400 000 caractères à partir du répertoire courant et affichage à l'écran de la liste de ces fichiers.

```
xstra> find . -type f -size +400000c -print
```

3) Etant positionné sur le répertoire de l'utilisateur xstra, recherche de tous les fichiers de nom *core*. Puis suppression de ces fichiers.

```
xstra> cd /home/xstra
xstra> find . -name core -exec rm {} \;
```

3.5 LA MANIPULATION DES FICHIERS

Quel que soit le travail que vous allez faire sur la machine, vous aurez à effectuer certaines tâches élémentaires telles que lister le contenu d'un répertoire, copier, effacer, ou afficher des fichiers. Nous présentons ci–dessous brièvement les commandes qui les réalisent :

ls LiSt files

Permet d'obtenir la liste et les caractéristiques des fichiers contenus dans un répertoire. Si aucun argument n'est donné, la commande *ls* affiche la liste des noms des fichiers du répertoire courant par ordre alphabétique.

Exemples

1) Positionnement sur le répertoire de l'utilisateur xstra. *ls* permet d'obtenir la liste des fichiers et répertoires existants à ce niveau.

```
xstra> cd /home/xstra
xstra> ls
bin develop essai projet1
xstra>
```

2) Même démarche mais en voulant obtenir toutes les entrées.

```
xstra> ls -a
.          .bash_history   bin   projet1
..         .bash_logout develop
.bashrc    .bash_profile essai
xstra>
```

3) En étant à la racine, la commande suivante permet de lister le contenu du répertoire */home/xstra* et d'obtenir toutes les informations.

```
xstra> cd /
xstra> ls -l /home/xstra
total 8
drwxr xr x 2 xstra  512  jan 18 10:21 bin
drwxr xr x 2 xstra  512  jan 15 16:05 develop
 rwxr  r   1 xstra   15  jan 16 14:40 essai
drwxr xr x 2 xstra  512  jan 18 10:21 projet1
xstra>
```

cat conCATenate

La commande *cat* est une commande multi–usage qui permet d'afficher, de créer, de copier et de concaténer des fichiers.

Exemples

1) Affichage du contenu du fichier */etc/passwd*.

```
xstra> cat /etc/passwd
§ (cf. résultat dans l'exemple
§ du paragraphe 2.1.3)
```

2) Création d'un fichier

```
xstra> cat >essai
Bonjour
Il fait beau
<ctrl-d>      § caractère de fin de fichier
xstra>
```

locate

Permet d'afficher le nom complet de tout fichier ou répertoire correspondant à un critère de recherche donné.

Exemple

Recherche les fichiers et répertoires contenant la chaîne de caractères *touch*.

```
xstra> locate touch
/usr/share/man/man1/touch.1.gz
/usr/X11R6/man/man4/mutouch.4x.gz
/bin/touch
xstra>
```

more MORE

less LESS (jeu de mots sur « more or less » : *less* est une amélioration de *more*)

Permettent d'afficher page par page à l'écran le contenu d'un fichier texte. La commande *more* est traditionnelle. Un utilisateur de Linux doit lui préférer la commande *less*, équivalente mais plus élaborée. *less* est utilisée par la commande *man* pour l'affichage de la documentation en ligne.

Exemple

```
# afficher page par page le contenu du
# fichier /etc/passwd
xstra> less /etc/passwd
```

Voir l'annexe A pour l'utilisation de *more* et *less*. La commande *less*, qui permet de remonter en marche arrière dans le texte, est beaucoup plus configurable. La variable d'environnement *LESS* permet d'en fixer les options.

cp CoPy

Cette commande permet la copie de fichiers. Elle s'utilise sous quatre formes :

1) La copie d'un fichier source dans un fichier destination.

Exemple

Dans le répertoire *xstra*, copie du fichier *essai* dans *essai1*.

```
xstra> cd /home/xstra
xstra> cp essai essai1
xstra>
```

Remarque

Il n'existe aucun contrôle sur le fichier destination : si le fichier *essai1* existe, son contenu est écrasé par le contenu du fichier *essai*.

2) La copie d'un fichier dans un répertoire.

Exemple 1

Copie du fichier *essai* du répertoire *xstra* dans le répertoire *xstra/projet1*.

```
xstra> cd /home/xstra
xstra> cp essai /home/xstra/projet1
xstra>
```

Exemple 2

Copie du fichier *essai1* du répertoire père */home/xstra* vers le répertoire courant */home/xstra/projet1*.

```
xstra> cd /home/xstra/projet1
xstra> cp ../essai1 .
xstra>
```

3) La copie d'un répertoire dans un autre (seuls les fichiers sont copiés : on obtient un message d'erreur pour la copie des répertoires).

Exemple

Copie du contenu du répertoire *xstra* dans */home/xstra/projet2*.

```
xstra> cd /home/xstra
xstra> mkdir projet2
xstra> cp * /home/xstra/projet2
xstra>
```

4) La copie récursive permet de copier une arborescence.

Exemple

Copie de l'arborescence de *xstra/projet1* sous *xstra/projet2*.

```
xstra> cd /home/xstra/projet1
xstra> cp -r * /home/xstra/projet2
xstra>
```

mv MoVe

Change le nom d'un fichier ou d'un répertoire. En première analyse, cette commande est équivalente à une copie, suivie d'une suppression. Elle s'utilise sous deux formes :

1) Transfert de *fichier1* dans *fichier2* et suppression de *fichier1*. Si *fichier2* existe, il est effacé :
mv fichier1 fichier2

Exemple

Transfert du fichier *essai1* dans *toto*.

```
xstra> cd /home/xstra
xstra> mv essai1 toto
xstra>
```

2) Transfert de(s) fichier(s) cité(s) dans le répertoire avec le(s) même(s) nom(s) : *mv fichier(s) répertoire*

Exemple

Transfert du fichier *toto* dans le répertoire */home/xstra/projet1*.

```
xstra> cd /home/xstra
xstra> mv toto /home/xstra/projet1
xstra> ls toto
ls: toto: No such file or directory
xstra> ls /home/xstra/projet1/toto
toto
xstra>
```

Remarque

La philosophie de cette commande sera détaillée au chapitre 10.

rm ReMove
Supprime un (ou plusieurs) fichier(s) d'un répertoire :
rm fichier(s)

Exemple

Suppression du fichier *toto* du répertoire *projet1*.

```
xstra> cd /home/xstra/projet1
xstra> rm toto
xstra>
```

grep

Recherche, dans un ou plusieurs fichiers, de toutes les lignes contenant une chaîne donnée de caractères (cette commande sera détaillée au chapitre 14).

Exemple

Recherche de la chaîne de caractères *beau* dans le fichier *essai*.

```
xstra> grep beau essai
Il fait beau
xstra>
```

wc Word Count
Cette commande permet le dénombrement des mots, lignes et caractères dans un fichier. Un mot est défini comme une suite de caractères précédée et suivie par des espaces, des tabulations, le début ou la fin de la ligne.

wc lwcL fichier

l	affiche le nombre de lignes
w	affiche le nombre de mots
c	affiche le nombre de caractères
L	affiche la longueur de la ligne la plus longue
fichier	liste de noms de fichiers à parcourir (ou entrée standard si vide)

Exemple

Impression du nombre de mots dans le fichier *essai*.

```
xstra> wc  w essai
4
xstra>
```

ln LiNk
Permet de désigner un fichier par plusieurs noms différents.

Exemple

Le fichier *f1* existe, le fichier *New f2* est créé sans occupation disque et est lié au fichier *f1*.

```
xstra> ln f1 New_f2
xstra> ls
f1 New_f2
xstra>
```

Le fichier (en tant qu'espace disque) porte les deux noms.

Remarque

La philosophie de cette commande sera détaillée au chapitre 10.

touch TOUCH
Cette commande permet (entre autres) de créer un fichier vide.

Exemple

```
xstra> touch f1
xstra>
```

echo ECHO
Affiche à l'écran le texte qui suit la commande *echo*.

Exemple

```
xstra> echo Il y a du soleil
Il y a du soleil
xstra>
```

Nous venons de décrire de façon succincte quelques commandes de base de Linux. Ces commandes sont présentées en annexe A de façon plus détaillée avec d'autres options.

3.6 MANUAL, LE MANUEL LINUX

La commande *man* permet de rechercher des informations sur les commandes. *man* est le manuel Unix en ligne. Cette commande recherche les informations, le cas échéant, dans deux répertoires et leurs sous–répertoires :

/usr/man
/usr/local/man

La liste des répertoires *man* référencés est définie dans le fichier `/etc/man.config`.

L'exécution de la commande permettant d'obtenir des informations sur la commande *ls* est :

```
xstra> man ls § La documentation Linux relative
             § à ls apparaîtra à l'écran page par page.
xstra>
```

Lors de l'appel à cette aide en ligne, vous trouverez des chiffres entre parenthèses situés après les noms de commandes. Ces chiffres précisent à quel chapitre de la documentation Linux sont décrites ces commandes.

La commande *man* fait appel à *less* pour présenter l'aide page par page. La variable *MANPAGER* permet de configurer le comportement de *less* dans *man*. Il est commode de garder dans *man* les options habituelles de *less* :

Exemple

```
xstra> export LESS=' z 2 -rsiaj3$'   § la commande export
xstra> export MANPAGER="less $LESS"  § est décrite au
                                     § paragraphe 6.7
```

Dans le projet Gnu, la documentation n'est pas au format du man, mais dans un format plus récent permettant de produire indifféremment la documentation en ligne et des manuels papiers, et comportant des renvois d'une section vers une autre : le format texinfo. La commande *info* permet de consulter et d'imprimer cette documentation qui se trouve dans le répertoire `/usr/info`. Une grande partie des logiciels disponibles dans une distribution Linux étant d'origine Gnu, la commande *info* sera de fait préférée à la commande *man*, d'autant que la commande *info* renvoie automatiquement sur le *man* au cas où la documentation n'existe qu'au format man. Un excellent tutoriel sur la commande *info* est obtenu par la commande :

```
info info
```

Il est possible de constituer simplement sa propre documentation en créant dans un répertoire précis un fichier dans lequel sont mises les pages de manuel désirées. Pour réaliser cette manipulation, l'administrateur du système doit intervenir.

La figure 3.2 représente la sous–arborescence où se trouvent la documentation standard, et éventuellement la documentation locale.

Les fichiers qui se trouvent dans les répertoires *man1*,..., *man9* sont des fichiers formatés au format man. Dans les répertoires *cat1*,..., *cat9* sont stockés des fichiers en clair (éventuellement compressés par le programme *gzip*). Les fichiers qui ne sont pas dans les répertoires *cat1*,..., *cat9* sont mis en forme à partir de ce qu'il y a dans les répertoires *man1*,..., *man9*. La documentation technique fournie se situe aux niveaux :

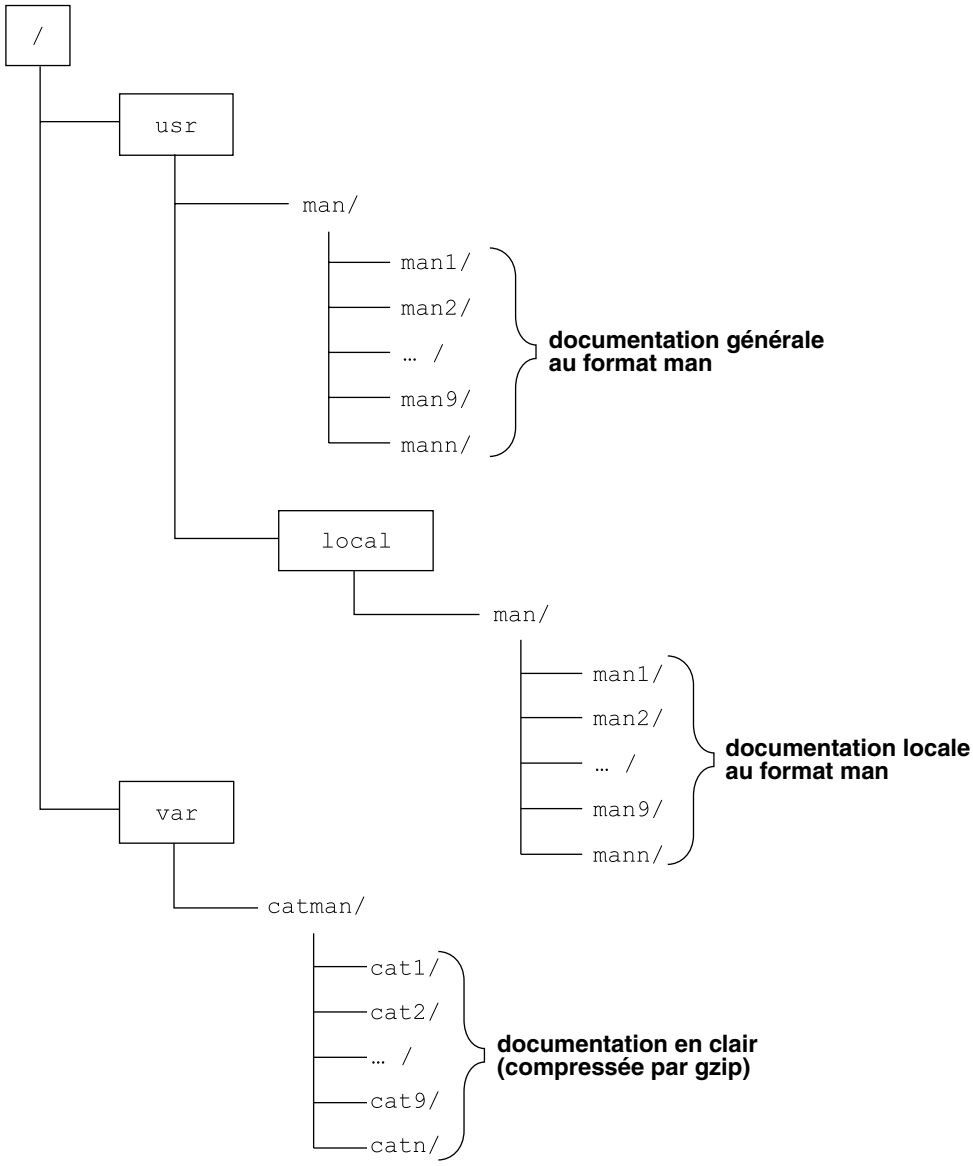

FIGURE 3.2. SOUS-ARBORESCENCE DU MANUEL.

/usr/man/man[1 9] pour le man de base
/usr/local/man/man[1 9] pour le man additionnel

Pour constituer une nouvelle documentation, l'administrateur du système positionnera les nouveaux fichiers dans l'un des deux répertoires suivants :

/usr/man/mann
/usr/local/man/mann

Les répertoires *mann* contiennent les fichiers formatés. Le suffixe de ces fichiers doit être *.n*.

Exemples

```
/usr/man/mann/madoc1.n
/usr/local/man/mann/madoc2.n
```

L'information sur *madoc1* ou *madoc2* sera obtenue à l'aide des commandes *man* ou *info* :

```
xstra> man madoc1
§ La documentation de madoc1 apparaîtra
§ page par page.
xstra> info madoc2 § idem
```

3.7 EXERCICES

Exercice 3.7.1

Dans votre répertoire d'accueil, créez l'arborescence suivante, en n'utilisant que des chemins relatifs :

```
rep1
 |---fich11
 |---fich12
 |---rep2
 |   |---fich21
 |   |---fich22
 |---rep3
 |   |---fich31
 |   |---fich32
```

Exercice 3.7.2

Vérifiez.

Exercice 3.7.3

Comment déplacer toute l'arborescence rep3 sous le répertoire rep2 ? Supprimez tout sauf rep1, fich11 et fich12.

Exercice 3.7.4

À l'aide de la commande `id`, déterminez votre UID et votre groupe (nom de groupe et GID). Combien y a-t-il d'utilisateurs dans votre groupe ?