www.shuttle.de/schweikh/home.html

Man page HOWTO

Mars 1998

(Adaptation française par Alexandre Devaure *adevaure@mail.dotcom.fr* , 2 juin 1999). Ce HOWTO explique ce que vous devrez avoir en tête quand vous prévoyez d'écrire une documentation en ligne (plus connue sous le nom de page de manuel) que vous voulez rendre accessible via la commande `man(1)`. Tout au long de ce HOWTO, une entrée du manuel sera simplement appelée page de manuel, quelque soit sa longueur réelle et sans intention sexiste.

Contents

1 Quelques évidences à propos de la documentation

Pourquoi écrivons-nous une documentation ? C'est une question bête. Parce que nous voulons que d'autres personnes puissent utiliser notre programme, notre fonction dans une librairie ou quoi que ce soit que nous avons écrit et rendu disponible. Mais écrire une documentation n'est pas suffisant :

- la documentation doit être accessible. Si elle est cachée à un endroit non standard où les outils de recherche relatifs à la documentation ne la trouveront pas, comment peut-elle remplir son rôle ?
- la documentation doit être fiable et précise. Il n'y a rien de plus irritant qu'un programme se comportant différemment de ce qui est écrit dans sa documentation. Les utilisateurs vous maudiront, vous enverront des courriers d'insulte, puis rejetteront à jamais tout autre travail venant de vous.

La méthode traditionnelle pour accéder à la documentation sous UNIX fait appel à la commande `man(1)`. Ce HOWTO décrit ce que vous devez faire pour écrire une page de manuel qui sera correctement traitée par les outils prévus à cet effet, dont les plus importants sont `man(1)`, `xman(1x)`, `apropos(1)`, `makewhatis(8)` et `catman(8)`.

La qualité et la véracité des informations sont, bien sûr, de votre ressort. Malgré tout, vous trouverez dans ce guide quelques idées qui vous permettront d'éviter certains pièges courants.

2 Comment accède-t-on aux pages de manuel ?

Vous devez connaître avec précision le mécanisme d'accès aux pages de manuel afin de savoir donner un nom correct à vos documents, et d'être capable de les installer au bon endroit. Chaque page de manuel appartient à une section spécifique, dénotée par un simple chiffre. Les sections les plus courantes rencontrées sous Linux sont :

- 1 : commandes utilisateurs pouvant être exécutées par tout le monde ;
- 2 : appels systèmes, c'est-à-dire les fonctions fournies par le noyau ;
- 3 : fonctions des bibliothèques ;
- 4 : périphériques, c'est-à-dire les fichiers spéciaux que l'on trouve dans le répertoire `/dev` ;
- 5 : descriptions des formats de fichiers (comme par exemple `/etc/passwd` ;
- 6 : les jeux, sans commentaire...
- 7 : divers (macros, conventions particulières, ...) ;
- 8 : outils d'administration exécutables uniquement par le super utilisateur ;
- 9 : un autre endroit (spécifique à Linux) destiné à la documentation des services offerts par le noyau ;
- n : nouvelle documentation, qui pourra être déplacée vers un endroit approprié ;
- o : ancienne documentation, qui peut être conservée encore un certain temps ;
- l : documentation locale, propre à ce système particulier.

Le nom du fichier source d'une page de manuel (le fichier d'entrée du système de formatage) est le nom de la commande décrite (ou de la fonction, du fichier, etc.), suivi d'un point et du numéro de section. Si, par exemple, vous documentez le format du fichier "*passwd*", vous devez appeler le fichier source "*passwd.5*". Nous avons ici un exemple d'un fichier qui porte le même nom qu'une commande ; nous aurions tout aussi bien avoir une fonction de bibliothèque appelée "*passwd*". L'organisation en sections constitue la méthode habituelle pour résoudre ces ambiguïtés : la description de la commande se trouvera dans le fichier "*passwd.1*" et notre hypothétique fonction de bibliothèque dans "*passwd.3*".

Quelquefois, une lettre est ajoutée au numéro de section comme par exemple "*xterm.1x*" ou "*wish.1tk*". Le but de cette notation est d'indiquer qu'il s'agit respectivement d'une documentation d'un programme X Window ou d'une application Tk. Certains programmes d'affichage du manuel peuvent exploiter cette particularité ; `xman`, par exemple affichera "*xterm(x)*" et "*wish(tk)*" dans la liste des documents disponibles.

S'il vous plaît, n'utilisez pas les sections n, o et l : selon le standard du système de fichiers (File System Standard), ces sections sont déconseillées, utilisez plutôt les sections numériques.

Attention aux éventuels conflits de noms avec des programmes, fonctions ou fichiers déjà existants. Ce serait certainement une mauvaise idée d'écrire un autre éditeur de texte et de le nommer ed, sed (pour super ed) ou red (pour Roger edition). En vous assurant que le nom de votre programme est unique, vous éviterez que quelqu'un exécute votre programme et qu'il lise la page de manuel d'un autre ou *vice verca*. Vous pouvez éventuellement vous aider de la base de données "lsm" qui recense beaucoup de programmes disponibles pour Linux.

Maintenant que nous savons quel nom donner à notre fichier, la prochaine décision est de choisir le répertoire dans lequel nous l'installerons (quand l'utilisateur lancera la commande "make install"). Sous Linux, toutes les pages de manuel sont dans des sous-répertoires à partir d'une racine mémorisée dans la variable d'environnement MANPATH. Les outils de traitement de la documentation l'utilisent de la même manière que le shell utilise la variable PATH pour trouver les exécutables. En fait, MANPATH a le même format que PATH : toutes les deux sont une liste de répertoires séparés par des ":" (mais MANPATH n'autorise pas de champs vides ou des chemins relatifs, seulement des chemins absolus). Si MANPATH n'existe pas ou si elle n'est pas exportée, /usr/man est utilisée comme valeur par défaut. Dans le but d'accélerer la recherche et pour garder les répertoires de taille raisonable, les répertoires pointés dans MANPATH (aussi appelés répertoires de base) contiennent une multitude de sous-répertoires nommés "*man<s>*" où <s> désigne le caractère correspondant à la section présenté plus haut. Toutes les sections ne sont pas représentées, il n'y a pas, par exemple de raison de garder une entrée "*mano*". Vous pourrez y trouver également des sous-répertoires appelés "*cat<s>*", "*dvi<s>*" et "*ps<s>*", qui contiennent toute la documentation formatée, prête à être affichée ou imprimée : nous reviendrons sur ce sujet plus loin. Le seul fichier à être présent à côté de ces sous-répertoires du répertoire de base s'appelle "*whatis*". Le but et la création de ce fichier sera décrit dans la section 11. La méthode la plus sûre pour installer au bon endroit une page de manuel de la section "s" est de mettre le fichier dans le répertoire "*/usr/man/man<s>*". Toutefois, un bon `Makefile` devra autoriser l'utilisateur de choisir un autre répertoire de base, disons par exemple par le biais d'une variable d'environnement que l'on pourrait nommer MANDIR. La plupart des distributions GNU peuvent être configurées à l'aide de l'option `-prefix=/nom/option`. Les pages de manuels correspondantes seront alors installées à partir du répertoire de base */mon/option/man*. Je vous suggère d'utiliser une méthode similaire pour vos réalisations personnelles.

Depuis l'avènement du "*Système de fichiers standard*" pour Linux (FS-STnd), les choses se sont compliquées. Le FS-STnd 1.2 stipule que :

> des aménagements doivent être faits dans la structure de */usr/man* pour supporter des pages de manuel écrites dans différentes (ou mutiples) langues.

Ceci est fait en introduisant un niveau de répertoires supplémentaire qui distingue les différentes langues. Citant encore le FS-Stnd 1.2 :

> Le nommage des sous-répertoires correspondants aux langues de */usr/man* est basé sur l'appendice E du standard POSIX 1003.1 qui décrit la chaîne de caractères d'authentification *locale* (qui est la méthode la mieux acceptée pour décrire un environement culturel). La chaîne *locale* se présente sous la forme
>
> `<langage>[_<pays>][.<jeu-de-caracteres>][,<version>]`

(Reportez vous au FS-Stnd pour voir quelques chaînes *locale*courantes.) D'après ces recommandations, nous avons nos pages de manuel dans */usr/man/<locale>/man[1-9lno]*. Les versions formatées se trouveraient alors bien entendu dans */usr/man/<locale>/cat[1-9lno]* : nous pourrions ne les fournir que pour une seule langue.

TOUTEFOIS, je (l'auteur du document, pas le traducteur) ne peut pas recommander de passer a cette structure en l'état actuel des choses. Le FS-Stnd 1.2 autorise aussi que

> les systèmes qui n'utilisent qu'une seule langue et jeu de caractères pour toutes les pages de manuel peuvent omettre la sous-chaîne <*locale*> et stocker toutes ces pages dans le répertoire *mandir*. Par exemple, les machines équipées seulement de pages de manuel en anglais codées en ASCII peuvent mettre les pages de manuel (les répertoires *man[1-9]*) directement dans */usr/man/*. Il s'agit en fait de l'arrangement habituel.

Je (l'auteur du document, pas le traducteur) ne changerai pas ma configuration tant que tous les outils (comme `xman`, `info`, `tkman` et beaucoup d'autres) ne seront pas tous adaptés à cette nouvelle structure.

3 A quoi ressemble une page de manuel formatée ?

Laissez-moi vous présenter un exemple que j'expliquerai plus tard. En raison de la nature et du mode de réalisation de ce document, nous ne pouvons pas reproduire les caractères accentués, ni les différents enrichissements du texte (gras et italiques principalement) ; consultez la section traitant des polices de caractères pour obtenir des détails sur ces possibilités.

Voici comment se présente la page de manuel de notre programme hyphothétique "`prout`" :

```
PROUT(1)               Manuel utilisateur              PROUT(1)


NAME
       prout - proutibule la bibliotheque plaf

SYNOPSIS
       prout [-plaf] [-c fichier-config ] fichier ...

DESCRIPTION
       prout  proutibule  la  bibliotheque plaf en mouglifiant la
       table des symboles.  Par  defaut,  la  commande  recherche
       tous  les segments glurb et les trie par ordre betagonique
       decroissant afin que  le  gloupeur  gloup(1)  les  trouve.
       L'entree  symdef  est  alors  compactee selon l'algorithme
       NABOB.  Les  fichiers  sont  traites  dans   leur   ordre
       d'apparition sur la ligne de commandes.

OPTIONS
       -b     N'affiche  pas  `bidouille  en cours' sur la sortie
              standard pendant le traitement.

       -c fichier-config
              Utilise le fichier de configuration  fichier-config
              au  lieu  du  fichier global /etc/prout.conf.  Cela
              supprime   aussi    l'effet    de    la    variable
              d'environnement PROUTCONF.

       -a     Traite  egalement  les  en-tetes froutz en plus des
              segments glurb.

       -r     Mode  recursif.  Fonctionne  a  la  vitesse  de  la
              lumiere,  mais  necessite  plusieurs  megaoctets de
              memoire virtuelle.

FICHIERS
       /etc/prout.conf
              Fichier de  configuration  general,  pour  tout  le
              systeme. Voir prout(5) pour plus de details.
       ~/.proutrc
              Fichier  de  configuration propre a chaque utilisa
              teur. Voir prout(5) pour plus de details.

ENVIRONNEMENT
       PROUTCONF
              Si elle existe, cette  variable  peut  contenir  le
              chemin  d'acces  complet a un autre fichier de con
```

```
              figuration global  prout.conf.   L'option  -c  rend
              cette variable inoperante.

DIAGNOSTICS
       Les  messages suivants peuvent etre affiches sur la sortie
       standard d'erreurs :

       Mauvais nombre magique.
              Le fichier d'entree ne semble pas etre  un  fichier
              archive.

       Segments glurb ancien style.
              prout  ne peut traiter que le nouveau style de seg
              ments glurb. Les bibliotheques GROBOL ne  sont  pas
              supportees dans cette version.

BOGUES
       Le  nom de cette commande aurait du etre choisi de maniere
       a mieux refleter sa fonction.

AUTEUR
       Marcel Dugenou    <dugenou@renux.freenix.fr>

VOIR AUSSI
       gloup(1), plaf(1), prout(5).

Linux                      JANVIER 1996                         1
```

Et voici les explications promises :

La section NAME :

C'est la seule section requise. Les pages de manuel sans une section "NAME" sont aussi utiles que des réfrigerateurs au Pôle Nord. Cette section a aussi un format standardisé constitué d'une liste de programmes ou noms de fonctions séparés par des virgules suivie d'un tiret et d'une courte description (habituellement une ligne) de la fonctionnalité que le programme (fonction ou fichier) est supposé dispenser. A l'aide de `makewhatis(8)` les sections NAME sont incluses dans les fichiers de la base de données de `whatis`. `makewhatis` est la raison pour laquelle la section NAME doit exister et pourquoi elle doit adhérer au format que j'ai décrit. Dans le source groff, elle doit ressembler à :

```
.SH NAME prout \- proutibule de la bibliotheque plaf
```

Le \- est important ici : le backslash sert a faire la différence entre le tiret et une marque de césure qui peut apparaître à l'intérieur du nom de la commande ou dans la ligne de description.

Attention : en l'état actuel des choses, vous ne pouvez pas traduire NAME par NOM en français, à moins de modifier la plupart des programmes `makewhatis` existants. C'est bien dommage.

La section SYNOPSYS

... est censée donner un aperçu sur les options du programme. Pour les fonctions, cette section fait la liste des fichiers à inclure et son prototype pour que le programmeur connaisse le type et le nombre d'arguments ainsi que le type de retour.

La section DESCRIPTION

Elle explique en détail pourquoi votre séquence de 0 et de 1 est la meilleure de toutes. C'est ici que vous étalez tout votre savoir ! Gagnez l'estime des autres programmeurs et des utilisateurs en faisant de cette section une source d'information sûre et détaillée. Expliquez à quoi servent les arguments, le format de fichier, les algorithmes qui effectuent le plus dur du travail.

La section OPTIONS

Elle donne une description pour chaque option, comment elle affecte le fonctionnement du programme. Vous le saviez, n'est-ce pas ?

La section FICHIERS

Elle indique les fichiers utilisés par le programme ou la fonction. Par exemple, les fichiers de configuration, les fichiers de démarrage, les fichiers sur lesquels le programme agit. Ce serait une bonne idée de donner les chemins absolus de ces fichiers et d'avoir un processus d'installation qui modifie la partie répertoire selon les préférences de l'utilisateur : les manuels de `groff` ont comme préfixe par défaut */usr/local*, donc ils référencent */usrl/local/lib/groff/** par défaut. Cependant, si vous installez en utilisant "`make prefix=/opt/gnu`", les références dans la page de manuel change en */opt/gnu/lib/groff/**.

La section ENVIRONNEMENT

fait la liste de toutes les variables d'environnement qui affectent votre programme ou fonction et, bien sûr, explique comment. La plupart du temps, les variables contiendront les chemins, nom de fichiers, options par défaut.

La section DIAGNOSTIQUES

Elle doit donner une vue d'ensemble des messages d'erreurs les plus courants de votre programme et des éventuelles solutions à ces problèmes. Il n'est pas nécessaire d'expliquer les messages d'erreurs du système (de `perror(3)`) ou des signaux fatals (de `psignal(3)`) qui peuvent apparaître pendant l'exécution de tout programme.

La section BOGUES

Devrait idéalement ne pas exister. Si vous êtes brave, vous pouvez décrire ici les limitations, les inconvénients, les caractéristiques que certains pourraient prendre pour des défauts. Si vous n'êtes pas brave, renommez-la en section "A FAIRE".

La section AUTEUR

Il est appréciable de l'avoir quand il y a des erreurs grossières dans la documentation ou dans le comportement du programme et que vous voulez envoyer un rapport de bogue.

La section VOIR AUSSI

C'est une liste de pages de manuel relatives à l'application citées par ordre alphabétique. Par convention, c'est la dernière section.

Vous êtes libres d'en inventer d'autres si elles n'empietent pas sur celles décrites au-dessus. Nous avons volontairement décrit une version francisée de page de manuel, puisque ce document est destiné aux pays francophones. Néanmoins, vous devez avoir conscience que si vous devez diffuser une application dans le monde entier, il vous faudra fournir un manuel en langue anglaise (ce qui est la version standard, traditionnelle), et que les noms "officiels" de ces sections sont en réalité, dans l'ordre : NAME, SYNOPSIS, DESCRIPTION, OPTIONS, FILES, ENVIRONMENT, DIAGNOSTICS, BUGS, AUTHOR et SEE ALSO.

Donc comment générer cette page de manuel ? J'attendais cette question, voici le source :

```
.\" Formater ce fichier par la commande :
.\" groff -man -Tlatin1 prout.1  (si vous avez saisi des accents Iso-8859-1)
.\" groff -man -Tascii  prout.1  (cas general )
.\"
.TH PROUT 1 "JANVIER 1996" Linux "Manuel utilisateur"
.SH NAME
prout \- proutibule la bibliotheque plaf
.SH SYNOPSIS
.B prout [-plaf] [-c
.I fichier-config
.B ]
.I fichier
.B ...
.SH DESCRIPTION
.B prout
proutibule la bibliotheque plaf en mouglifiant la table des symboles.
Par defaut, la commande recherche tous les segments glurb et les trie
par ordre betagonique decroissant afin que le gloupeur
.BR gloup (1)
les trouve. L'entree symdef est alors compactee selon l'algorithme NABOB.
Les fichiers sont traites dans leur ordre d'apparition sur la ligne
de commandes.
.SH OPTIONS
.IP -b
N'affiche pas `bidouille en cours' sur la sortie standard pendant
le traitement.
.IP "-c fichier-config"
Utilise le fichier de configuration
.I fichier-config
au lieu du fichier global
.IR /etc/prout.conf .
Cela supprime aussi l'effet de la variable d'environnement
.B PROUTCONF.
.IP -a
Traite egalement les en-tetes froutz en plus des segments glurb.
.IP -r
Mode recursif. Fonctionne a la vitesse de la lumiere, mais necessite
plusieurs megaoctets de memoire virtuelle.
.SH FICHIERS
.I /etc/prout.conf
.RS
Fichier de configuration general, pour tout le systeme. Voir
.BR prout (5)
pour plus de details.
.RE
.I ~/.proutrc
.RS
Fichier de configuration propre a chaque utilisateur. Voir
.BR prout (5)
pour plus de details.
```

```
.SH ENVIRONNEMENT
.IP PROUTCONF
Si elle existe, cette variable peut contenir le chemin d'acces complet
a un autre fichier de configuration global
.IR prout.conf .
L'option
.B -c
rend cette variable inoperante.
.SH DIAGNOSTICS
Les messages suivants peuvent etre affiches sur la sortie standard d'erreurs :

Mauvais nombre magique.
.RS
Le fichier d'entree ne semble pas etre un fichier archive.
.RE
Segments glurb ancien style.
.RS
.B prout
ne peut traiter que le nouveau style de segments glurb. Les bibliotheques
GROBOL ne sont pas supportees dans cette version.
.SH BOGUES
Le nom de cette commande aurait du etre choisi de maniere a mieux
refleter sa fonction.
.SH AUTEUR
Marcel Dugenou    <dugenou@renux.freenix.fr>
.SH "VOIR AUSSI"
.BR gloup (1),
.BR plaf (1),
.BR prout (5).
```

4 Comment documenter plusieurs choses dans une seule page de manuel ?

De nombreux programmes (`grep`, `egrep`) et fonctions (`printf`, `fprintf`,...) sont documentées dans une seule page de manuel. Cependant, ces pages seraient inutilisables si elles n'étaient accessibles que par un seul nom. Nous ne pouvous nous attendre à ce qu'un utilisateur se souviennent que la page de manuel de `egrep` est en fait celle de `grep`. Il est par conséquent indispensable que la page soit accessible sous différents noms. Vous avez plusieurs possibilités pour y arriver :

1. avoir des copies identiques pour chaque nom ;
2. connecter toutes les pages de manuels en utilisant des liens physiques ;
3. utiliser les liens symboliques pointant la page de manuel ;
4. utiliser le mécanisme de "source" de `groff` fournie par la macro "`.SO`".

La première possibilité est une perte de place. La deuxième n'est pas recommandée parce que les versions intelligentes du programme `catman` peuvent gagner beaucoup de temps en regardant le type du fichier et son contenu. Les liens physiques réduiraient l'efficacité de cet outil (dont le but est de formater toutes les pages de manuel pour qu'elles soient affichées plus rapidement). La troisième alternative comporte un piège si vous êtes concerné par la portabilité, vous devez savoir qu'il existe des systèmes de fichiers qui ne supportent pas les liens symboliques. En bref, la Meilleure Chose (TM) est d'utiliser le mécanisme source de `groff`.

Voila comment l'utiliser : si vous voulez que votre page soit accessible sous les noms `truc` et `bidule` dans la section 1, alors mettez la page de manuel dans `truc.1` et réalisez le fichier `bidule.1` contenant :

```
.SO man1/truc.1
```

Il est important de spécifier le répertoire *man1/* aussi bien que le nom du fichier `truc.1` car lors de l'exécution de `groff`, celui-ci aura comme répertoire courant le répertoire de base des pages de manuel, et il interprètera les arguments de `.SO` comme étant relatifs à cet emplacement.

5 Quel ensemble de macros utiliser ?

Il y a de nombreux ensembles de macros étudiés spécialement pour écrire des pages de manuel. Ils sont habituellement dans le répertoire de macro de `groff` */usr/lib/groff/tmac*. Les noms de fichiers sont du genre *tmac.<quelque chose>*, où *<quelque-chose>* est l'argument de l'option -m de `groff`. `groff` utilisera *tmac.<quelque-chose>* quand l'option `-m <quelque-chose>` sera donnée. Souvent, l'espace entre "-m" et <quelque-chose> est oublié, on écrira donc `groff -man` pour formater des pages de manuel en utilisant l'ensemble de macro tman.an. Voilà la raison de ce nom bizarre "*tman.an*".

En plus de *tman.an*, il existe un autre ensemble de macro populaire, *tman.doc*, originaire de l'université de Californie à Berkeley (UCB). De nombreuses pages de manuels de BSD l'utilisent et il semble que UCB en a fait son standard pour la documentation. Les macros de *tman.doc* sont plus souples mais, hélas, il y a des lecteurs de pages de manuels qui ne les utilisent pas mais qui appellent toujours `groff -man`. Par exemple, toutes les versions du programme `xman` que j'ai rencontrées faisaient la tête devant les pages de manuels requérant *tman.doc*. Donc faîtes-vous une faveur : utilisez *tmac.an*, utiliser un autre ensemble de macros est considéré comme hasardeux. *tmac.andoc* est un pseudo ensemble de macros qui regarde le source et charge soit *tmac.an* ou *tmac.doc*. En fait, tous les programmes de lecture du manuel devraient l'utiliser mais jusqu'à présent, peu le font, aussi il vaut mieux assurer le coup en se cantonnant au bon vieux *tmac.an*. Tout ce que je dirais à partir de maintenant concernant les macros est valable seulement pour *tmac.an*. Si vous voulez quand même utiliser les macros de *tmac.doc*, voici un pointeur vers leur documentation et un mode d'emploi très détaillé : *www.bsdi.com/bsdi-man* . Vous trouverez un formulaire de recherche sur cette page. Entrez `mdoc` et il vous trouvera `mdoc(7)` et `mdoc.samples(7)`, un didacticiel sur la réalisation des pages de manuel BSD.

6 Quels préprocesseurs puis-je utiliser ?

`groff` est fourni avec au moins 3 préprocesseurs, `tbl`, `eqn` et `pic` (certains systèmes les nomment `gtbl`, `geqn` et `gpic`). Ils sont destinés à traduire leurs macros et leurs données en code source `troff` standard. `tbl` est un préprocesseur de tableaux, `eqn` en est un d'équation et de mathématiques et `pic` gère les images. Consultez leurs pages de manuel pour découvrir les fonctionalités qu'ils proposent.

Mais autant être clair : n'écrivez pas de pages de manuel qui utilisent des préprocesseurs.

`eqn` produira généralement un résultat catastrophique sur des périphériques du genre télétype, qui malheureusement représentent 99% des visualtions de pages de manuel. Par exemple *XAllocColor.3x* contient des formules avec des exposants. A cause de la nature de ces terminaux, l'exposant sera sur la même ligne que la base. «*N puissance deux*» s'affichera "N2".

Il vaut mieux éviter `tbl` aussi, car je n'ai jamais vu aucun `xman` qui fonctionne avec lui. `xman 3.1.6` utilise la ligne de commande suivante pour formater les pages de manuel, par exemple *signal(7)* :

```
gtbl /usr/man/man7/signal.7 | geqn | gtbl | groff -Tascii -man \
/tmp/xmana01760 2> /dev/null
```

qui coince sur toutes les sources utilisant `gtbl`, car sa sortie est redirigée encore une fois vers `gtbl`. Le résultat donne une page de manuel sans votre tableau. Je ne sais pas si c'est un bogue ou une particularité de `gtbl` qui s'étrangle sur sa propre sortie ou si `xman` devrait être un peu plus gentil et ne pas utiliser `gtbl` deux fois... De toute façon, si vous voulez un tableau, formatez-le vous-même et mettez-le entre les lignes `.nf` et `.fi` ce qui permettra de ne pas le formater. Vous ne pourrez pas avoir de gras ou d'italique par cette méthode mais elle permettra d'avoir votre tableau dans tous les cas.

Je n'ai jamais vu une page de manuel nécessitant le préprocesseur `pic` mais je n'aimerais pas ça. Comme vous pouvez le voir plus haut, `xman` ne l'utilise pas et `groff` ferait sûrement la danse de Saint-Guy en voyant les données en entrée.

7 Dois-je distribuer les sources et/ou la documentation déjà formatée ?

Voyons les avantages (+) et les inconvénients (-) de quelques possibilités choisies :

1. code source uniquement :
   - (+) distribution plus petite ;
   - (-) inutilisable sur les systèmes ne disposant pas de `groff`.
2. verison formatée non compactée uniquement :
   - (+) utilisable même sur des systèmes dépourvus de `groff` ;
   - (-) l'utilisateur ne peut pas générer un fichier dvi ou PostScript ;
   - (-) gâchis de l'espace disque sur les systèmes sachant gérer aussi les pages compressées.
3. version formatée et compactée seulement :
   - (+) utilisables même sur des systèmes dépourvus de `groff` ;
   - (-) l'utilisateur ne peut pas générer de fichier dvi ou PostScript ;
   - (-) quel format de compactage utiliser ? .Z ? .z ? .gz ? Tous ?.
4. code source et la version formatée non compactée :
   - (+) accessible même sur les systèmes ne disposant pas de `groff` ;
   - (-) taille de la distribution plus grande ;
   - (-) certains systèmes peuvent nécessiter des pages de manuels formattées et compactées ;
   - (-) informations redondantes sur les systèmes équipés de `groff`.

A mon avis, la meilleure solution est de distribuer uniquement le code source. L'argument selon lequel la documentation ne pourra pas être accessible sur les systèmes sans `groff` n'a aucune importance. Plus de 500 pages de manuel du Projet de Documentation de Linux ne sont que sous forme de code source. Les pages de manuel de XFree86 ne sont disponibles que sous forme de code source. Les pages de manuel de la FSF n'existent que sous forme de code source. En fait, j'ai rarement vu des logiciels distribués avec les pages de manuels formatées. Si un administrateur a besoin que les pages de manuel soient accessibles, il aura forcément installé `groff`.

8 Quelles sont les conventions pour les fontes ?

Avant tout, n'utilisez pas les opérateurs directs de fonte comme `\fB` `\fP`, etc. Employez des macros avec des arguments. Cette méthode vous évitera une erreur classique : oublier un changement de fonte à la fin d'un mot ce qui provoque la continuation du gras ou de l'italique jusqu'au prochain changement de fonte. Croyez-moi, ça arrive plus souvent qu'on ne le pense !

Les macros *tmac.an* offrent les possibilités suivantes :

- `.B` caractères gras
- `.BI` gras et italiques en alternance
- `.BR` gras et romain en alternance
- `.I` italiques
- `.IB` italiques et gras en alternance
- `.IR` italiques et romain en alternance
- `.RB` romain et gras en alternance
- `.RI` romain et italiques en alternance
- `.SM` taille réduite (9/10 du corps normal)
- `.SB` gras, taille réduite (NON petit et gras en alternance)

X et Y en alternance signifie que les arguments impairs seront imprimés en X et les pairs en Y. Par exemple :

```
.BI "Arg 1 est gras, " "arg2 est en italiques, " "arg3 en gras"
```

Les guillemets sont nécessaires pour placer des espaces dans un argument.

Voilà donc pour ce qui est possible. Voyons maintenant comment il faut utiliser ces possibilités (des parties ont été honteusement copiées de `man(7)`) :

Bien qu'il existe de nombreuses conventions typographiques pour les pages de manuel dans le monde UNIX, l'existence de plusieurs centaines de pages de manuel spécifiques à Linux définit nos standards :

Pour les fonctions, les arguments sont toujours en italiques, même dans la section SYNOPSYS, alors que le reste est en gras. Vous écrirez donc :

```
.BI "mafonction(int " argc ", char **" argv );
```

Les noms de fichiers sont toujours en italiques, hormis dans la section SYNOPSYS où les fichiers à inclure sont en gras. Vous écrirez alors :

```
.I /usr/include/stdio.h
```

et

```
.B #include <stdio.h>
```

Les noms des macros, qui sont habituellement en majuscules, sont en gras :

```
.B MAXINT
```

Lors de l'énumération d'une liste de codes d'erreurs, ces codes sont en gras. Cette liste fait généralement appel à la macro `.TP` (paragraphe avec titre) comme ci-dessous :

```
.TP
.B EBADF
.I fd n'est pas un descripteur de fichier valide
.TP
.B EINVAL
.I fd ne convient pas pour être lu
```

Toute référence à une autre page de manuel (ou à la page courante) est en gras. Si le numéro de la section du manuel est indiqué, il s'écrit en roman, sans espace :

```
.BR man (7)
```

Les acronymes sont plus élégants lorsqu'ils apparaissent dans un corps plus petit. Je recommande donc :

```
.SM UNIX
.SM ASCII
.SM TAB
.SM NFS
.SM LALR(1)
```

9 Comment dois-je présenter ma page de manuel ?

Voilà quelques conseils pour rendre votre documentation plus sûre, plus lisible et plus «formatable» :

- Les exemples doivent fonctionner : testez-les (utilisez le copier-coller pour passer à votre shell ce que contient votre page de manuel) et redirigez la sortie de votre commande dans votre page, ne tapez pas ce que vous PENSEZ que votre programme affichera.
- relisez-vous, corrigez toutes les éventuelles fautes de frappe ou d'orthographe, faîtes-vous relire par un tiers (surtout si vous ne rédigez pas le texte dans votre langue natale) . (d'ailleurs ce HOWTO n'a pas été relu... Y a-t-il un volontaire ?)

- testez votre page de manuel : est-ce que `groff` trouve des erreurs lors du formatage ? C'est agréable de trouver dans un commentaire la ligne de commande qu'il faut taper pour le formatage. Est-ce que la commande `man(1)` affiche des erreurs ou des avertissements lorsqu'on appelle `"man votre_programme"` ? Est-ce que la façon dont `man(1)` utilise le système de formatage produit le résultat escompté ? Est-ce que cela fonctionne aussi bien avec `xman(1x)` et `tkman(1tk)` ? `XFree86 3.1` contient la version 3.1.6 de `xman` qui décompacte les pages avec :

```
gzip -c -d < %s > %s
zcat < %s > %s
```

- Est-ce que `makewhatis(8)` pourra extraire la ligne de description de la section NAME ?

10 Comment puis-je avoir un texte en pur ASCII sans tous ces fichus ^H^ de contrôle ?

Jetez un oeuil à la commande `col(1)`, `col` peut enlever ces caractères d'effacement. Pour les impatients, voici la commande :

```
$ groff -t -e -mandoc -Tascii manpage.1 | col -bx > manpage.txt
```

Les options `-t` et `-e` disent à `groff` d'utiliser les préprocesseurs `tbl` et `eqn`. C'est inutile pour les pages de manuel ne nécessitant pas de préprocesseur mais cela ne gêne pas, si ce n'est une surcharge du processeur. D'un autre côté, ne pas utiliser `-t` alors qu'il est nécessaire fera que les tableaux seront très mal formatés. Vous pourrez même trouver ("deviner" serait un terme plus exact) la commande nécessaire pour traiter tel ou tel document groff (pas uniquement des pages de manuel) par le biais de `grog` :

```
$ grog /usr/man/man7/signal.7
groff -t -man /usr/man/man7/signal.7
```

En fait, `grog` signifie "*GROff Guess*", et cet outil fait bien ce qu'il dit (en anglais, guess = deviner...) : il tente de deviner la commande nécessaire pour formater un document groff en fonction de son contenu. S'il était parfait, nous n'aurions jamais plus besoin d'options. Mais s'il arrive qu'il détermine un mauvais jeu de macros, je ne l'ai par contre jamais vu se tromper sur les préprocesseurs à employer.

Voici un petit script Perl réalisé par l'auteur de ce document, qui peut supprimer les en-têtes et les pieds de page, ce qui permet de gagner quelques longueurs de papier lorsque l'on imprime de longues et complexes pages de manuel. Sauvez-le dans un fichier nommé *strip-header* et mettez-le en mode 755.

```
#!/usr/bin/perl -n
#  pour qu'il avale tout le fichier en une seule fois:
undef $/;
#  on enleve les sauts de page:
s/\n{4}\S.{50,}\n{6}\S.{50,}\n{3}/\n/g;
#  le premier en-tete et le dernier pied de page:
s/\n\S.{50,}\n//g;
#  transorme deux ou plus lignes vides consecutives en une seule:
s/\n{3,}/\n\n/g;
#  et voila ce qui reste...
print;
```

Il faut appeler ce programme en tant que premier filtre après la comande man, car il se base sur le nombre de sauts de ligne issus de groff. Par exemple :

```
$ man bash | strip-headers | col -bx > bash.txt
```

11 Comment avoir une belle page de manuel en PostScript ?

```
$ groff -t -e -mandoc -Tps manpage.1 > manpage.ps
```

Imprimez-la à l'aide de votre imprimante PostScript préférée ou d'un interpréteur. Voir la section précédente pour les options.

12 Comment faire fonctionner les programmes `apropos` et `whatis` ?

Supposons que vous recherchiez les compilateurs installés sur votre système et comment les invoquer (nous considérons le cas courant, où tout le manuel est en langue anglaise). Pour répondre à cette question (fréquemment posée), il faut faire :

```
$ apropos compiler
f77 (1) - Fortran 77 compiler
gcc (1) - GNU C and C++ compiler
pc (1) - Pascal compiler
```

`apropos` et `whatis` sont utilisées pour obtenir une réponse rapide sur les pages de manuels qui contiennent des informations sur un certain sujet. Les deux programmes cherchent dans des fichiers nommés *whatis* qui sont dans chaque répertoire de base du manuel. Comme je l'ai déjà dit, les fichiers de la base de données *whatis* contiennent une entrée d'une ligne pour chaque page de manuel dans l'arborescence des répertoires successifs. En fait, cette ligne est exactement celle de la section NAME (pour être précis : tout est réduit à une seule ligne et le tiret est supprimé, la section étant placée entre parenthèses). Ces fichiers sont créés à l'aide du programme `makewhatis(8)`. Il en existe plusieurs versions, donc référez-vous à la page de manuel du programme pour connaître les options possibles. Afin que `makefile` puisse extraire les sections NAME correctement, il est important que vous, le rédacteur du manuel, respectiez le format de cette section décrit dans la partie 2. La différence entre `apropos` et `whatis` est ce qu'ils recherchent et où. `apropos` (qui est l'équivalent de `man -k`) cherche la chaîne de caractères qui lui est passée en argument n'importe où dans la ligne alors que `whatis` (équivalent de `man -f`) recherche dans la partie avant le tiret un nom de commande complet. Par conséquence, `whatis` dira s'il y a un manuel de `cc` mais restera muet pour `gcc`.

13 La langue française

C'est bien sûr à vous de décider si vous allez rédiger votre manuel en français, en anglais ou dans ces deux langues. Le français possède des règles typographiques très différentes de l'anglais : n'espérez pas pouvoir les respecter avec les outils de formatage du manuel. Consultez la documentation de `groff` si vous désirez lui faire prendre en compte les motifs de césure de la langue de Molière, mais en ayant conscience que ce ne sera sans doute pas possible sur tous les systèmes sur lesquels votre documentation est susceptible d'être exploitée.

Vous pouvez utiliser les caractères accentués, pourvu qu'ils soient saisis selon la norme ISO-8859-1 (standard sous Linux). Les pages devront alors être formatées avec l'option -Tlatin1 . Mais il faudra que toute la chaîne de visualisation soit capable de gérer les caractères ISO sur 8 bits, ce qui est rarement le cas sans une configuration particulière des utilitaires `more` ou `less` généralement employés.

Vous voilà prévenu !

14 Les conditions de copie