man−db – the database cached manual pager suite

Graeme W. Wilford <eep2gw@ee.surrey.ac.uk>
Colin Watson <cjwatson@debian.org>

This document describes the setup, maintenance and use of a generic online manual page system with special reference to the man−db package and its advanced features.

UNIX is a registered trademark of the X/Open Company, Ltd.
NFS is a registered trademark of Sun Microsystems, Inc.
PostScript is a registered trademark of Adobe in the United States.

The general conventions used throughout this manual include

- file names and paths in *italic*, e.g. */usr/share/man*.
- variable strings (usually path components) enclosed within <> and in *italic*, eg. *<sec>*,
- program names in **bold**, eg. **man**.
- commands that can be typed at a shell prompt in a box, eg. man foobar.
- environment variables denoted as follows: $**ENV_VAR**

Copyright © 1995 Graeme W. Wilford
Copyright © 2001, 2002, 2003, 2007 Colin Watson

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the copyright holder.

1. Introduction

1.1. man−db

man−db is a package that is designed to provide users with online information in a fast and friendly manner while at the same time offering flexibility to the system administrator.

It is made up of several user programs:

- **man** − an interface to the on-line reference manuals
- **whatis** − search the manual page names
- **apropos** − search the manual page names and descriptions
- **manpath** − determine search path for manual pages
- **lexgrog** − directly read header information in manual pages

several maintenance programs:

- **mandb** − create or update the manual page index caches
- **catman** − create or update the pre-formatted manual pages

and a special pre-formatter that knows about compressed manual pages:

- **zsoelim** − satisfy .so requests in roff input

In addition to these compiled programs, there are two shell scripts, **mkcatdirs** and **checkman** in the *tools* subdirectory. These scripts aid the creation of cat directories and check for duplicated manual pages, respectively.

The following manual pages are provided with this package to explain correct format and usage. **man**(1), **whatis**(1), **apropos**(1), **manpath**(1), **lexgrog**(1), **manpath**(5), **mandb**(8), **catman**(8) and **zsoelim**(1).

1.1.1. The concept

man−db originally started out life as program suite man−1.1B, written by John W. Eaton <jwe@che.utexas.edu> and maintained by Rik Faith <faith@cs.unc.edu> to which support proposed by the newly formed FSSTND committee regarding cat directories was added.

Since then, man−db's most innovative feature: the database cache scheme[1] has been significantly developed. The basic idea was to reduce manual page search times to a minimum. The following piece of text is included from the man-db-2.2 distribution:

> The theory: If you go to a library to take a book out, what do you do?
>
> a) Go and look where it might be on a micro-fiche/terminal, take a look where it is supposed to be on the shelf, and then go look at the new arrivals if it's not where it's supposed to be?
>
> OR
>
> b) Start at one end of the ground floor, look along every bookshelf until you've completed that floor, then go up a level and start again until you've found what you're looking for?

Since then the database **index** scheme has evolved greatly. Every manual page and stray cat page on the system is registered in an **index** database cache which stores various details about the file including the timestamp, the location and the whatis[2] information. This information is kept up to date by regular runs of

[1] originally conceived after observing the actions of the Perl-based manual pager suite, man-pl written by Tom Christiansen <tchrist@convex.com>

[2] one line description of the manual page

mandb. In some configurations **man** also looks for filesystem changes each time it is invoked and helps to keep the database cache current, but this imposes a penalty on manual page search times.

1.2. The manual page system

The simplest manual page system will have a single manual page hierarchy. This will typically be

/usr/share/man

beneath which will be several subdirectories of the form *man<sec>* where *<sec>* is **1**, **2**, **3**, **4**, **5**, **6**, **7** or **8**. These are referred to as *sections* of the manual. Others may exist and they are not restricted to single character names. eg.

/usr/share/man/manfoo

is a valid section subdirectory. Other common sections include **9**, **n**, **l**, **p** and **o**.

Within these section subdirectories reside the manual pages themselves. Their filenames follow the pattern

/usr/share/man/man<sec>/<name>.<sec><ext>

where in most cases *<ext>* is an empty string. An example is manual page **cp**

/usr/share/man/man1/cp.1

which resides in *section* **1** and has no special *extension*.

1.3. Sections of the manual

The manual is split up into sections to ease access and to cater for manual pages that share the same name. It is common for a program and function to share the same name. **kill** is a good example. This is both a program which can be used to send a process a signal and an operating system call with similar functionality. Their manual pages are stored under sections **1** and **2** respectively. Thus, sections are used to separate out the program manual pages from the function manual pages and so on. The table below shows the *section* numbers of the manual followed by the types of pages they contain.

Section	Section contents
1	user executable programs or shell commands
2	system calls (functions provided by the kernel)
3	library calls (functions within system libraries)
4	special files (usually found in */dev*)
5	file formats and conventions eg. */etc/passwd*
6	games
7	macro packages and conventions eg. **man**(7), **groff**(7).
8	system administration commands
9	kernel routines [Non-standard]
n	new [obsolete]
l	local [obsolete]
p	public [obsolete]
o	old [obsolete]

1.4. The format of manual pages

The format in which manual pages are stored is **NROFF/TROFF** or more generally ROFF. This is a typesetter style language[3] which requires formatting before being viewed. In fact some manual pages require preformat processing to correctly format tables or equations.

[3] similar in some aspects to **TeX**

If the page is to be viewed on screen in a text environment, **NROFF** is used as the primary formatter. If the page is to be printed or displayed in a graphical environment, **TROFF** is used. Traditionally, **TROFF** formatted files for a **C/A/T** (Computer aided Typesetter) which is now obsolete.

The **GNU** ROFF (**GROFF**[4]) suite of programs offer a choice of output types including **X**, **dvi** and **postscript**. When configuring man−db, the preference is to use **GROFF** rather than **TROFF**.

1.5. Arguments to configure

To allow the configuration program, **configure**, to be non-interactive, it can be passed various options to alter the default settings. Generic **configure** options are discussed in *docs/INSTALL*. Options that are specific to the man−db package are described below.

−−enable−cache−owner[=ARG]
: By default, system-wide cache files will be owned by user man. Use this option with an argument to change the cache file owner.

−−disable−cache−owner
: Use this option to leave the ownership of system-wide cache files unconstrained. Users will be allowed to modify them.

−−disable−setuid
: By default, **man** will be installed as a setuid program to the user that owns the system-wide cache files. Use this option to install **man** as a non-setuid program instead.

−−enable−mandirs=OS
: By default, man−db supports manual page directories in any of several layouts used by free and proprietary versions of **UNIX**. However, in certain cases, this can cause man−db to find the wrong page by mistake, especially when the names of some manual pages on the system contain periods. Use this option with an argument of GNU, HPUX, IRIX, Solaris, or BSD (or more than one of these, separated by commas) to support only the layouts typically used on each of those systems. Note that man−db is not currently capable of writing cat pages in the proper BSD layout.

−−with−device=DEVICE
: Use this flag to alter the default output device used by **NROFF**. DEVICE is passed to **NROFF** with the −T option. **configure** will test that **NROFF** will run with the supplied device argument.

−−with−db=LIBRARY
: configure will look for database interface libraries in the order gdbm, Berkeley DB and finally ndbm and will #define appropriate variables relative to the first one found. To override the built-in order on platforms having a choice of interface library, use this option to specify which library to use.

−−enable−automatic−create
: If this flag is used, **man** will automatically create index databases for users' private manual page hierarchies.

−−disable−automatic−update
: Normally, **man** will update entries in index databases if it finds newly installed manual pages (if the **−−update** flag is used) or delete entries if manual pages are removed. This flag suppresses this behaviour.

−−disable−cats
: Normally, **man** will automatically try to create cat files corresponding to manual files when a manual page is read. This flag suppresses this behaviour.

---

[4] Written by James Clark <jjc@jclark.com> and now maintained by Ted Harding <ted.harding@nessie.mcc.ac.uk> and Werner Lemberg <wl@gnu.org>

2. The specifics of Sections

2.1. Package specific manual page sections

The use of package specific manual page sections is discouraged as packages large enough to warrant their own section probably contain manual pages that span other sections. An example might be package **foo** that has its own section

/usr/share/man/manfoo

which contains manual pages describing its programs, the library routines it offers and the format of several of its configuration files. These pages would normally be allocated to sections **1**, **3** and **5** respectively and thus combining them all under section **foo** is misleading. Subtle problems will arise if there are any base name-space clashes with standard manual pages, e.g. **exit**(3), **exit**(foo) and the order in which they should be shown.

There are two standard solutions to this problem.

(1) Create a separate manual page hierarchy for the package's manual pages such as

/usr/local/packages/foo/man

(2) Install the pages in their relevant sections, with a unique extension appended to the filename such that

/usr/share/man/manfoo/exit.foo

would instead be installed as

/usr/share/man/man1/exit.1foo

Only (2) offers a complete solution to manual page ordering problems and allows users to access the desired page directly.

2.2. Selecting a section type

2.2.1. Specifying a section

This is done via use of the section argument to man

```
man 1 exit
```

will look for *exit.1** in section **1** of the manual. If *exit.1* exists, it will be displayed in preference to *exit.1foo*

```
man 1foo exit
```

will look for *exit.1foo** in section **1** of the manual. The asterisk (*) represents a wild-card of any type or length, including length zero.

For an argument to be interpreted as a section name rather than a page name, it must either begin with a digit, or be included in the standard section list. The default section list is defined in *include/manconfig.h* to be **1**, **n**, **l**, **8**, **3**, **2**, **5**, **4**, **9**, **6** and **7**. This should be modified in order and content to meet the local conventions. It may be altered at run-time using the **SECTION** directive in the man−db configuration file.

Every subdirectory section name in the entire system must be in the list, including sections found in imported manual page hierarchies. It is not necessary to list sections with extensions unless a special ordering for those extensions is desired. The order is important because in normal operation, **man** will only display the first manual page it finds that meets the search criteria. Using the **−−all** argument will cause **man** to attempt to display all manual pages that meet the criteria. See **man**(1) for further information.

Having an excess of sections listed will not slow **man** down.

2.2.2. Specifying an extension

If the section is unknown, but the package extension is, it is possible to use the extension argument

```
man −e foo exit
```

to search in all sections for manual pages named *exit* from package *foo*.

3. Filesystem structure

3.1. Manual page hierarchies

It is often common for manual page systems to have more than one manual page hierarchy. Indeed one of the systems I use has the following globally accessible hierarchies

/usr/man
/usr/local/man
/usr/local/tex/man
/usr/local/pbm/man
/usr/X11R6/man
/usr/openwin/man
/usr/local/packages/pvm/man

A full system $**MANPATH** would be a colon separated list of these directories. The order is important, and is observed by man−db's search algorithms. The order is very much related to the user's $**PATH** environment variable, and should be set on a per user basis, or not set at all. If a user's $**PATH** causes

/usr/local/packages/bin/foobar

to be executed in preference to

/usr/bin/foobar,

it is essential that

man foobar

displays the manual page located within

/usr/local/packages/man

rather than within

/usr/share/man

To ensure correct order, the program **manpath** may be used to set the $**MANPATH** environment variable. See **manpath**(1) and **manpath**(5) for details.

3.2. Setting the MANPATH

If using a Bourne style login shell such as **bash**, **ksh**, or **zsh**, the commands

```
export MANPATH
MANPATH=`manpath −q`
```

can be added to **$HOME**/*.profile*

If using a C style login shell such as **csh** or **tcsh**, the commands

```
setenv MANPATH `manpath −q`
```

can be added to **$HOME**/*.login*

N.B. $**PATH** must be set prior to using **manpath**. The setting of $**MANPATH** is actually unnecessary as the man−db utilities will dynamically determine the manpath if $**MANPATH** is unset.

3.3. Determination of the internal manpath

All man−db utilities, **manpath** included, will use the user's $**MANPATH** environment variable if set and not equal to "". Otherwise the user's $**PATH** environment variable is queried. If this is unset or is set to "", the determined manpath will simply be any

MANDATORY_MANPATH

elements defined in the man−db config file.

Assuming that a $**PATH** exists, each path element it contains is scanned for in the config file. If found, the corresponding manpath element is appended to the internal manpath. However, if the element is not mentioned in the config file, a man directory relative to it will be sought. The subdirectories *../man*, *man*, *../share/man*, or *share/man* relative to the path component are appended to the internal manpath if they exist. Finally, the internal manpath is stripped of duplicate paths before being processed by the **NLS** and 'Other OS' routines. These may add to or modify the separate path elements giving priority to **NLS** manual pages or add OS-relative manpaths.

3.4. Other OS's manual pages

It is common to have collections of heterogeneous computer systems linked together in a network. In some circumstances[5] it is advantageous to be able to access the manual pages of these other systems directly from your system. This feature is known as alternate system support. The accepted way to setup this support is to NFS mount the respective systems' manual page hierarchies under the native manual page hierarchies. An example:

System	Manual page hierarchy
<local>	/usr/share/man
newOS	/usr/share/man/newOS
userix	/usr/share/man/userix
<local>	/usr/local/man
newOS	/usr/local/man/newOS
userix	/usr/local/man/userix

Rather than have multiple NFS mounts from a single machine, this may be accomplished by NFS mounting

<other-sys>:/usr

somewhere on the local system and using symbolic links within the manual hierarchies. To access these *alternate systems* using **man** use the **−m** or **−−systems** option, eg.

```
man −−all −−systems userix:newOS 5 passwd
```

would provide manual pages showing the structure of */etc/passwd* on systems **userix** and **newOS** in that order. A manual page would *not* be displayed about the local systems conventions. Please read the relevant man−db utility's manual page for further and more specific information.

3.5. NLS manual pages

NLS manual pages should be installed in NLS subdirectories of a standard manual page hierarchy. The subdirectory names should be made up of language, territory, and character set components as necessary to specify the locale of the manual page.

The character set component describes the encoding of the manual page itself, and not the encoding in use by the user; a manual page installed under the **fr.UTF-8** subdirectory will be used in the **fr_FR.ISO-8859-1** locale as well as **fr_FR.UTF-8**, and converted between encodings as necessary. If no character set is specified in the subdirectory name, man−db will attempt to detect whether each page is encoded using UTF-8 or a legacy character set appropriate for the language. Accordingly, the recommended scheme for installing manual pages is to encode them in UTF-8 (or, if that is not practical, in the legacy character set) and install them in directories *without* a character set component in their names.

The territory should normally be omitted unless it is necessary to describe the manual page text. For example, Brazilian Portuguese is quite distinct from Portuguese and so should be installed under the **pt_BR**

[5] writing portable software instantly comes to mind

subdirectory, but a single German manual page will typically suffice in Austria as well as in Germany and so should be installed under the **de** subdirectory.

The following table gives some examples.

Language	Territory	Character Set	Directory
French	any	UTF-8 or ISO-8859-1	/usr/share/man/fr
French	Canada	ISO 8859-1	/usr/share/man/fr_CA
French	any	UTF-8	/usr/share/man/fr.UTF-8
German	Germany	UTF-8	/usr/share/man/de_DE.UTF-8
German	Switzerland	ISO 8859-1	/usr/share/man/de_CH.ISO-8859-1
Japanese	Japan	UTF-8 or EUC-JP	/usr/share/man/ja_JP
Japanese	Japan	EUC-JP	/usr/share/man/ja_JP.EUC-JP
Japanese	any	UTF-8	/usr/share/man/ja.UTF-8

On systems supporting UTF-8, it is recommended that all manual pages be encoded using UTF-8 where possible, in order to simplify the task of editing a variety of pages without reconfiguring editors and terminals and the like.

Each of these directories are then interpreted as manual page hierarchies themselves and may contain the usual section subdirectories. Access to NLS manual pages is achieved via use of the **setlocale**(3) function which queries user environment variables to determine the current locale. Internally to the man−db utilities, this locale string is appended to each manpath element and the resultant NLS manpath element is searched before the standard manpath element. In this way, an NLS manual page that matches the search criteria will be shown before or in place of the standard American English page.

If a user's $**MANPATH** consists of or is determined as

/usr/local/man:/usr/share/man:/usr/X11R6/man

and their locale is set to **de_DE**, the command

```
man −−systems userix:man foobar
```

would produce the following internal man−db manpath elements

/usr/local/man/userix/de_DE
/usr/local/man/userix/de
/usr/local/man/userix
/usr/share/man/userix/de_DE
/usr/share/man/userix/de
/usr/share/man/userix
/usr/X11R6/man/userix/de_DE
/usr/X11R6/man/userix/de
/usr/X11R6/man/userix
/usr/local/man/de_DE
/usr/local/man/de
/usr/local/man
/usr/share/man/de_DE
/usr/share/man/de
/usr/share/man
/usr/X11R6/man/de_DE
/usr/X11R6/man/de
/usr/X11R6/man

foobar would be searched for in the order of manual page hierarchies listed. Additional directories corresponding to manual pages encoded in different character sets would be used if present.

3.5.1. ISO 8859-1 (latin1) manual pages

By default **NROFF** will format manual pages into a form suitable for a typewriter style device, e.g. a terminal screen. **GNU NROFF** is capable[6] of formatting ROFF into a form suitable for 8-bit latin1 capable output devices. To enable output for such a device, give the option

−−with−device=DEVICE

to **configure** where DEVICE is the suitable and supported output format, in this case **latin1**.

3.5.2. Displaying non-ASCII characters on a Linux virtual terminal

To view non-ASCII characters at the Linux console, you must have one of the kbd[7] and console−tools packages installed. If your system does not come with suitable configuration already, then please see the documentation in the kbd or console−tools package for details on how to configure the console for your locale. On modern systems, the best choice is likely to be to use the UTF-8 encoding with a font suitable for your language. Make sure that your locale environment variables match the encoding displayed by the console. For display under the “X Window System”, a suitable 8-bit-clean terminal emulator is required.

3.5.3. Viewing ASCII pages formatted for latin1 output device

When formatting an ASCII manual page for a latin1 output device, **GNU NROFF** will take advantage of the extra characters available and will always produce a text page containing some latin1 (8-bit) symbols. The table[8] below, taken from **man**(1), illustrates the differences.

Description	Octal	ISO 8859-1	ASCII
continuation hyphen	255		−
bullet (middle dot)	267	•	o
acute accent	264	´	’
multiplication sign	327	×	x

To display such symbols on a 7 bit terminal or terminal emulator, they must be translated back into standard ASCII. The **−7** option with **man** will enable this simple reverse translation.

This option may be useful if your site has both 7 and 8-bit capable output devices and nroff is using the latin1 output device to format manual pages.

3.6. Cat pages

It has become standard practice to store the formatted manual pages on disk so that subsequent requests for the manual page do not have to involve the formatting process. These pre-formatted manual pages are known as *cat* pages. Although cat pages require additional disk storage requirements, they provide a substantial speed increase and their use is recommended.

The automatic support for storing and using cat pages is brought about by simply creating suitable directories for them.

---

[6] see **nroff**(5) for the output device formats available with your **NROFF**

[7] written and maintained by Andries Brouwer <aeb@cwi.nl>.

[8] The ISO 8859-1 and ASCII columns of this table will be identical if this manual was formatted for an ASCII based typewriter display, i.e. using **NROFF** in its native mode.

3.7. Cat page hierarchies

Traditionally, cat pages were stored under the same manual hierarchy as their source manual pages, in *cat<sec>* subdirectories rather than *man<sec>*. This situation is rather limiting in several situations:

- When it is advantageous to mount */usr* as a read-only filesystem. Cat pages cannot be supported in this situation without use of symbolic links to various other areas of the filesystem. This situation is a greater problem if the media itself is read-only, such as CD-ROM.
- When NFS mounting alternate OS's manual page hierarchies. The alternate system may be under someone else's control and they may not want cat pages stored on their system. In fact, it is usually a good idea to export the manual page filesystems read-only, or import them that way. It is possible to avoid the problems, this time with even more symbolic links that may need periodic updating.
- If there is a mixture of normal cat files and stray cats[9], it is very difficult to periodically *trim* the cat space disk usage by removing seldom accessed cat files.

To avoid all of these problems simultaneously, it was decided to support local cat page directory caches.

3.8. Local cat page directory caches

Any location for cat page hierarchy may be specified in the man−db configuration file. The location of the database cache associated with each manual page hierarchy will always be at the root of the cat page hierarchy. By default, the cat page hierarchy shadows the manual page hierarchy. The **FHS** proposes */var/cache/man* as the location for such directories, although man−db allows any directory hierarchy to be used. The **FHS** path transformation rule is as follows:

> */usr/<hierarchy>/share/man/<locale>/man<sec>/page.<sec><ext>*

should be formatted into the cat file

> */var/cache/man/<hierarchy>/<locale>/cat<sec>/page.<sec><ext>*

where the *<locale>* directory component may be missing and *<ext>* may be an empty string.

The suggestion is that stray cats are located in the traditional hierarchy under */usr* whereas re-creatable cat pages are stored under the local writable hierarchy */var/cache/man.* **man** follows strict rules in determining which file is displayed.

As an example, the following route is taken if all three files exist.

(1) Check relative modification time stamps of the manual file and the traditional cat file. If the cat file is up to date (has an equal time stamp), display it.

(2) The traditional cat file is out of date. Check relative time stamps of the manual file and the alternate cat file. If the cat file is up to date, display it.

(3) The alternate cat file is out of date. Format the manual file and display the result in the foreground, while updating the alternate cat file in the background.

When a cat file is created, its time stamp is set to that of the corresponding manual file. Manual files are often stored in **tar** archives, and time stamps may be preserved when these archives are unpacked. Simply checking whether the cat file is newer would sometimes cause **man** to display an out-of-date cat file in this case, when it should have reformatted the manual file instead.

---

[9] cat files that have no source manual page, i.e. they cannot be recreated.

4. Compression

4.1. Compressed manual pages

It is possible to maintain a system of compressed manual pages. This imposes a small overhead on the formatting process, but is nevertheless usually reasonable in order to avoid unnecessary consumption of disk space.

Presently, the compression extension/decompressor pairs must be known at compile time although any number may be defined and used. The following structure is predefined in man−db:

Extension	Decompressor
gz	gzip −dc
z	gzip −dc
Z	compress −dc

It is a relatively easy operation to include further pairs in this structure. See *include/comp_src.h* for details and an example.

Support for compressed manual pages is compiled into the man−db utilities by default. To completely disable this support, edit *config.h* and comment out the following line

#define COMP_SRC 1

This will enable a minor speed increase, but note that support for stray cats with any compression extension other than the default will also be disabled.

4.2. Compressed cat pages

man−db compresses cat files by default. During configuration, **configure** will try to find **gzip** and, if found, all cat files produced by **man** will be compressed with

gzip −7c

and have a **.gz** extension appended. If **gzip** is not found,

compress −c

is used as the compressor and the extension **.Z** is appended.

To store cat files in an uncompressed state and to disable compressed extension processing completely, edit *config.h* and comment out the following line

#define COMP_CAT 1

4.2.1. Stray cats

Normally, **man** will only look for cat files with the default compression extension. The default compression extension is dependent on the default compressor and may be an empty string if the support for compressed cats is disabled.

It is possible for a system to be supplied with stray cat files located in the traditional cat page hierarchy. To make matters worse, they may have compression extensions other than the default and reside on read-only media. In such circumstances, stray cat files will be accepted with any compression extension that is also supported for manual pages.

This special treatment of stray cat pages is removed if support for compressed manual pages is turned off or not available.

5. Formatting

As already pointed out in the introduction, there are two primary formatters common to **UNIX**: **NROFF** and **TROFF**.

In the following sections, I will use the term **TROFF** to describe the typesetter formatter and **NROFF** to describe the typewriter formatter. The term ROFF will be used to describe a generic formatter.

5.1. GROFF

If using the **GROFF** package, there is a further choice, **GROFF** itself. Essentially, **GROFF** forms a pipeline of processors including **TROFF** and an output processor which translates the ditroff produced by **TROFF** into the appropriate output format. The default output format, or device, for **GROFF** is PostScript. Anything else must be specified using the device argument. To illustrate **GROFF**, the command

```
groff −Tdvi /dev/null
```

will form the following pipeline

```
troff −Tdvi /dev/null | grodvi
```

If **GROFF** is tied to **man**'s **−T** option, it is still possible for **man** to produce ditroff via use of the **−Z** option.

In **GROFF** 1.09, **NROFF** is bundled as a shell script that calls **GROFF**, which in turn calls **TROFF** with the default options **−Wall −mtty-char −Tascii**, passing the result through **grotty** before it finally reaches the screen.

It is imperative that the script does not pass pre-processing options to **GROFF**'s command line as **man** takes care of this separately.

5.2. Devices

Both **NROFF** and **GROFF** may allow output device selection. As mentioned previously, classic **NROFF** produces output suitable for a typewriter device, classic **TROFF** produces output suitable for a **C/A/T** and **GROFF** produces output suitable for a PostScript interpreting device by default.

5.3. Macros

There are several ROFF macro sets in existence that are suitable for manual pages. Unfortunately, they tend to be incompatible with each other.

During configuration, **configure** will attempt to determine a suitable macro set for the local system's manual page collection. It attempts to use **NROFF** with the following three macro packages:

macro package	macro filename	nroff command
andoc	tmac.andoc or andoc.tmac	nroff −mandoc
an	tmac.an or an.tmac	nroff −man
doc	tmac.doc or doc.tmac	nroff −mdoc

The first that succeeds is used. The **andoc** macro set is suitable for manual pages written using either **an** or **doc** macro commands, but not a combination of both.

5.4. Pre-format processors (pre-processors)

Manual pages may require pre-processing by any of the following

Program	ID	Pre-processes
eqn	e	equations
tbl	t	tables
grap	g	graphs
pic	p	pictures
refer	r	A bibliography
vgrind	v	program listings

It is possible to assign a default pre-processor list that all manual pages will be passed through prior to the primary formatter. By default, this is empty. To define a default list, edit *include/manconfig.h* and un-comment the following line

/* #define DEFAULT_MANROFFSEQ "t" */

which will enable **tbl** processing by default. To change the list, replace the **t** with a suitable string of processor ID's.

Pre-process options may be provided at run time in various forms, but in general the pre-processors required by each manual page is indicated in the first line of the manual page itself. See **man**(1) for details.

If a manual page does not contain a pre-processor string in its first line, it will be scanned for well-known ROFF requests used to pass input to certain pre-processors. Thus, the pre-processor string is often unnecessary for correct output, but should nevertheless be included for efficiency.

5.5. Format scripts

It is very likely that alternate systems manual pages may require non-standard macro packages or possibly even special pre-processors. To tackle such problems, special format scripts may be created on a per manual hierarchy basis.

If the file

<manual_hierarchy>/mandb_nfmt

exists and is executable, it is expected to be able to correctly format a manual page originating from *<manual_hierarchy>* to its standard output. It will be supplied with either two or three arguments:

- manual page filename
- pre-processor string
- output device (optional)

Similarly, if the option **−T**<*device*> or **−t** was supplied to **man** and the file

<manual_hierarchy>/mandb_tfmt

exists and is executable, it will be used in the same way.

An example of such a script, supplied by Markus Armbruster <armbru@pond.sub.org>, who provided support for external formatter scripts, can be found as *tools/mandb_fmt−script*

The script can be used as both an **NROFF** and **TROFF/GROFF** format script and can be installed as *mandb_nfmt* and hard linked to *mandb_tfmt* after modification appropriate for your particular site.

6. The index database caches

As mentioned in the introduction, man−db uses database lookups to search for manual page locations and information. When performing a manual page lookup or a basic **whatis** search, the databases are searched in

key → *content*

mode and are as fast as the underlying databases can be. When performing **apropos** or special **whatis** searches, the databases are searched in a linear way, which, although far more expensive than *keyed* lookup, is no worse than traditional text based file searching.

6.1. index database location

The databases are always located at the root of the cat page hierarchy, whether this is the same as the manual page hierarchy or not. As file locking mechanisms are employed to ensure that concurrent processes do not update a database simultaneously, it is almost imperative that the databases reside on a local filesystem since file locking across NFS filesystems may be unavailable or flaky. To avoid such problems, **man** can be compiled without database maintenance support. See the section titled "Modes of operation" for details.

6.1.1. Manual hierarchies with no index database

It is possible for the man−db utilities to operate without aid from an index database. Under such circumstances, search methods will use only file globbing and whatis type searches are performed on any traditional whatis text databases that may exist. Only the traditional cat hierarchy is searched for cat files.

6.1.2. User manual page hierarchies

A user may have any number of personal manual page hierarchies listed in their $**MANPATH**. By default, **man** will maintain **mandb** created databases at the root of user manual page hierarchies. The definition of a user manual hierarchy is that it does not have an entry in the man−db configuration file. See **manpath**(5) for details.

6.2. Contents of an index database

There are four kinds of entry in an index database.

(1) A direct entry regarding a particular manual page. Manual pages that are unique in terms of name use just a single entry in the database and can be looked up by simply using the name as the key.

(2) A common name index entry that lists the extensions of all of the manual pages sharing the common index entry name. Manual pages that share common names but have differing extensions each have a single database entry, but this time they are looked up with a key comprised of their name and their extension. The entire set of common named pages also has an common name index entry that informs of the extensions available.

(3) An indirect entry that has a pointer to the real entry. Manual pages that are whatis references to a particular page do not physically exist so they have a pointer to the entry containing the location of the real manual page.

(4) Special identification entries. There is one special key name, "$version$" that identifies the database storage scheme version.

In order to support looking up manual pages in a case-insensitive fashion, keys are stored in lower case. If the name of the page was not already in lower case, its true case is also stored in the common name index entry.

In the following entries, the character "|" will be used to separate the fields. In reality a tab is used. Direct and indirect entries takes the form:

<name> → *<realname>*|*<ext>*|*<sec>*|*<mtime.sec>*|*<mtime.nsec>*|*<ID>*|*<ref>*|*<filter>*|*<comp>*|
<whatis>

Common name index entries take the form:

> *<name>* → |*<realname1>*|*<ext1>*|*<realname2>*|*<ext2>*|*<realname3>*|*<ext3>*| . . . *<realnamen>*| *<extn>*

and common name direct or indirect entries take the form:

> *<name>*|*<ext>* → *<realname>*|*<ext>*|*<sec>*|*<mtime.sec>*|*<mtime.nsec>*|*<ID>*|*<ref>*|*<filter>*| *<comp>*|*<whatis>*

where in each case the filename being represented is formed as

> *<manual_hierarchy>/man<sec>/<name>.<ext>.<comp>*

in the case of a manual page, or

> *<cat_hierarchy>/cat<sec>/<name>.<ext>.<comp>*

in the case of a stray cat.

If any of the fields would be empty, a single "−" is stored in its place. *<comp>* represents the compression extension, *<mtime.sec>* is an integer representing the seconds part of the last modification time of the manual page, *<mtime.nsec>* is an integer representing the nanoseconds part of the last modification time of the manual page, *<ref>* points to the entry containing the location of the real page, *<ID>* is one of the following identification letters, and *<filter>* represents any preprocessors that are needed to display the page.

ID	#define	Description
A	ULT_MAN	ultimate manual page, the full source nroff file
B	SO_MAN	manual page containing a .so request to an ULT_MAN
C	WHATIS_MAN	virtual whatis referenced page pointing to an ULT_MAN
D	STRAY_CAT	cat page with no source manual page
E	WHATIS_CAT	virtual whatis referenced page pointing to a STRAY_CAT

The *ID* illustrates the precedence. Some types of manual page can be referenced by several means, e.g. .so requested and whatis referred. In such a case, only one reference must be stored in the database, the precedence level decides which.

6.2.1. Favouring stray cats

With the above rules of precedence, it is possible for a valid stray cat page to be replaced by a whatis referred page sharing identical name-space.

If you would like to see the stray cat page **kill**(1) instead of the **bash_builtins**(1) page referenced by **kill**(1), edit *include/manconfig.h* and un-comment the following line

/* #define FAVOUR_STRAYCATS */

6.2.2. Accessdb

A simple program, **accessdb** is included with man−db. It will output the data contained within a man−db database in a human readable form. By default, it will *dump* the data from */var/cache/man/index.<db-type>*, where *<db-type>* is dependent on the database library in use.

Supplying an argument to **accessdb** will override this default. Tabs are replaced in the output by a tilde "~" in the *key* field and a single space in the *content* field.

6.2.3. Example database

As an example of both **accessdb** and the database storage method, the output of src/accessdb man/index.bt

after first running

src/mandb man

from the top level build directory is included below.

$version$ -> "2.5.0"
accessdb -> "- 8 8 1410381979 324541691 A - - - dumps the content of a man-db database in a human readable format"
apropos -> "- 1 1 1410381979 268541692 A - - - search the manual page names and descriptions"
catman -> "- 8 8 1410381979 328541691 A - - - create or update the pre-formatted manual pages"
lexgrog -> "- 1 1 1410381979 268541692 A - - - parse header information in man pages"
man -> "- 1 1 1410381979 280541692 A - t - an interface to the on-line reference manuals"
manconv -> "- 1 1 1410381979 272541692 A - - - convert manual page from one encoding to another"
mandb -> "- 8 8 1410381979 324541691 A - t - create or update the manual page index caches"
manpath -> " manpath 5 manpath 1"
manpath~1 -> "- 1 1 1410381979 300541691 A - - - determine search path for manual pages"
manpath~5 -> "- 5 5 1410381979 304541691 A - - - format of the /etc/manpath.config file"
whatis -> "- 1 1 1410381979 300541691 A - - - display one-line manual page descriptions"
zsoelim -> "- 1 1 1410381979 304541691 A - - - satisfy .so requests in roff input"

6.3. Database types

man−db has support for various low level database libraries commonly in use today. The interfaces to the libraries are known as

- ndbm (**UNIX**)
- gdbm (**GNU**)
- btree (Berkeley DB)

man−db currently does not hold more than one database open at any time, so

- dbm (**UNIX**)

support could be added in the future.

6.4. Limitations

The general differences and limitations are best compared in a table.

Name	Type	File name	Content memory		Concurrent access	Shareable
			type	limit		
ndbm	hash	index[10]	static	1Kb	none	no
gdbm	hash	index.db	dynamic	–	file locking	no
btree	binary tree	index.bt	static	–	none	yes

Those types that have no built in concurrent access strategy are provided with **flock**(2) based file locking by man−db.

Berkeley DB initializes its databases very quickly, so **btree** may have some performance advantages when doing **man** searches. However, it is quite heavyweight and its library SONAME and on-disk formats have changed a number of times to provide features considerably beyond what man−db needs, so the preferred library interface is now **gdbm**. **configure** will look for **gdbm**, **btree** and then finally **ndbm** routines when

[10] ndbm databases are physically represented by two files, *index.dir* and *index.pag*, but are referred to simply as *index* by the interface routines.

configuring man−db.

6.5. Sharing databases in a heterogeneous environment

It may be necessary or advantageous to share databases across platforms, regardless of the potential file locking problems.

An example would be a user having a personal manual page hierarchy in an NFS based home directory environment, whereby the home directory is held on and mounted from a single machine in a heterogeneous network.

In this context, the database cache will have the same name and reside in the same place on all machines. There are at least two ways to deal with this problem.

- Hack the *include/manconfig.h* file on each platform to provide a unique database name for each system. No databases will be shared.
- Install and use the Berkeley DB database interface library on each platform. These databases can be shared across big-endian/little-endian platforms although a database created on a big-endian platform will suffer a small access penalty when used by a litle-endian machine and vice-versa.

7. Miscellaneous

7.1. Modes of operation

The man−db utilities can operate in many different modes, allowing varying degrees of freedom, functionality and security. No mode requires that the manual page hierarchies be writable.

(1) Default mode

man is setuid to the user MAN_OWNER which is 'man' by default and is changeable via options to **configure**. **mandb**, if run by the superuser or MAN_OWNER, creates globally accessible index databases owned by MAN_OWNER. Once the databases are created, **man** will update entries in them if it finds newly installed manual pages (if the **−−update** flag is used) or delete entries if manual pages are removed. In this mode it is possible for a malicious **man** user to deliberately lock a database as a writer, thus denying read access to other users.

If cat directories exist and have the correct permissions, **man** will take care of producing cat files. These will be owned by MAN_OWNER. The default permissions of both cat files and databases are 0644.

(2) No man database updates

This mode also requires **man** to be setuid, but is favoured where databases must be shared in an environment unfriendly to kernel locking procedures, eg. NFS. It also prevents possible "denial of service" attacks by malicious **man** users as **man** never opens the databases as a writer in this mode. To replace the functionality lost by disallowing **man** write access to the databases, **mandb** should be rerun whenever new manual pages are installed. Otherwise, **man** will not be able to use the database to find and display the newly added manual pages, and will have to use the filesystem instead. Each index database may be owned by an arbitrary user who will have subsequent write access to the database. Cat files are created in the same way as for mode (1) above.

To use the man−db utilities in this mode, give the option '−−disable−automatic−update' to **configure**.

(3) No man database updates or cat production

man is installed not setuid. This mode of operation probably offers the highest level of security but it requires higher levels of maintenance than other modes due to the restrictions imposed upon **man**. Each database is owned by an arbitrary user as in mode (2). Each cat hierarchy is also owned by an arbitrary user who is responsible for creating cat files using **catman** whenever new manual files are installed. **man** will be completely passive in its action, i.e. no index databases will be written to and no cat files are ever produced.

To use the man−db utilities in this mode, supply the options '−−disable−cache−owner −−disable−setuid −−disable−automatic−update −−disable−cats' to **configure**, or build man−db as in mode (1) and install the binaries without the setuid bit set.

(4) Wide open

man is installed not setuid. This mode is similar in operation to the majority of vendor supplied, non setuid, cat file supporting manual pager suites. It is not recommended. The databases are owned by an arbitrary user who maintains them using **mandb**. **man** does not update the databases. Cat files are produced and stored in world writable cat directories and have world write access themselves.

To use the man−db utilities in this mode, supply the options '−−disable−cache−owner −−disable−setuid −−disable−automatic−update' to **configure**, edit *include/manconfig.h* and change the definition of CATMODE from 0644 to 0666.

Other variations can also be used. In fact it is possible for **man** to actually create index databases, usually the job of **mandb**, for users' private manual page hierarchies. This is enabled by giving the option '−−enable−automatic−create' to **configure**.

In summary, *include/manconfig.h* contains definitions for

- CATMODE
- DBMODE

the setuid installation and operation of **man** is modified by supplying either of the following options to **configure**:

- −−enable−setuid
- −−disable−setuid

and other aspects of **man**'s behaviour are controlled by the following options to **configure**:

- −−enable−automatic−create
- −−disable−automatic−update
- −−disable−cats

7.2. NFS root squash

If **man** is installed setuid to an arbitrary user and is run by root, instead of gaining the effective user id of the setuid user, **man** is run with both uid and euid as root. This is neccesary due to infelicities with the **POSIX** setuid() function call: All users except root may change to and from the effective (setuid) user, however once root has setuid(user), there is no way back.

A side effect of this is that **NFS** mounted cat hierarchies or databases will be unwritable if the following conditions exist:

- man/catman/mandb is run by root
- The NFS mount has the root squash flag set

To get around this problem, the root user must first attain the ID of the cat hierarchy or database owner before running **man/catman/mandb** whenever the databases need updating or cat files are to be produced.

7.3. NLS message catalogues

man−db has built in support for native language message catalogues. That is, it can issue messages in the locale of the user's choice. This will only occur if the locale's translation has been written. Before undertaking a translation, please contact the Translation Project (http://translationproject.org/) who are coordinating such activities.

7.4. Credits

The authors would like to thank the following people for their time, effort, support, ideas and code which went into man−db:

Markus Armbruster <armbru@pond.sub.org>
Lionel Cons & colleages <cons@dxcern.cern.ch>
Carl Edman <cedman@princeton.edu>
Caleb Epstein <epstein_caleb@jpmorgan.com>
Lars Fenneberg <lf@gimli.comlink.de>
Zoltan Hidvegi <hzoli@cs.elte.hu>
Nils Magnus <magnus@unix-ag.uni-kl.de>
Daniel Quinlan <quinlan@yggdrasil.com>
Fabrizio Polacco <fpolacco@debian.org>
Gordon Sadler <gbsadler1@lcisp.com>
Colin Phipps <cph@cph.demon.co.uk>
Paul Slootman <paul@wurtel.net>
Jose Rodriguez <boriel@airtel.net>
Eirik Fuller <eirik@hackrat.com>
Matej Vela <vela@debian.org>
Clint Adams <schizo@debian.org>
Jeremy C. Reed <reed@reedmedia.net>
Erik Andersen <andersen@codepoet.org>
Giuseppe Sacco <eppesuig@debian.org>

David Weinehall <tao@debian.org>
Ralph Corderoy <ralph@inputplus.co.uk>
Yuri Kozlov <kozlov.y@gmail.com>
Henning Makholm <henning@makholm.net>
Lars Wirzenius <liw@iki.fi>
Nicolas François <nicolas.francois@centraliens.net>
Ivan Shmakov <oneingray@gmail.com>
Peter Breitenlohner <peb@mppmu.mpg.de>
Robert Luberda <robert@debian.org>
Chusslove Illich <caslav.ilic@gmx.net>

and all those translators listed in the **man/THANKS** file.

Glossary

manual page
: A file containing descriptions related to the use of a function or program or the structure of a file. The name of the file is formed from the title of the manual page followed by a period followed by the name of the section that it resides in, optionally followed by an extension. The format of the file is **NROFF** and may be compressed, having a suitable compression extension appended.

cat page
: A formatted manual page suitable for viewing on a vt100-type terminal.

stray cat page
: A cat page that does not have a relative manual page on the system, i.e. only the cat page was supplied or the manual page was removed after the cat page had been created.

section
: Each manual page or cat page hierarchy is divided into sections, each section having its own directory. Manual page hierarchy section names begin with 'man' and cat page sections with 'cat'.

extension
: A package may provide manual pages with filenames ending in a package-specific extension name. This allows manual pages with the same title to coexist in the same manual page hierarchy and section without sharing the same filename. It also provides a further mechanism for man to select the correct manual page.

manual page hierarchy
: A directory tree divided into manual page sections, each containing a collection of manual pages.

cat page hierarchy
: A directory tree divided into cat page sections, each containing a collection of cat pages.

traditional cat page hierarchy
: The same location as the manual page hierarchy.

alternate cat page hierarchy
: A separate location to that of the traditional cat page hierarchy.

traditional cat page
: A cat page located in a traditional cat page hierarchy.

alternate cat page
: A cat page located in an alternate cat page hierarchy.

Contents