WTFM: Write the Fine Manual

A Gentle Introduction to Manual Pages

G. Branden Robinson

Debian Project

June 27, 2006

Outline

1. Motivation
2. Manpage Basics
3. Manpage Tips and Tricks
4. Meta-Manpage Issues

Why Should We Write Manual Pages ("Manpages")?

- We tell people to read them.
- Sometimes we do this without knowing a manpage on a given subject exists in fact.

```
Of 2253 commands, found manpages
for 2000 (253 missing).
```

- `man-db` is `Priority: important` — unlike Web browsers and PDF viewers.
- Manpages are one of users' last lines of defense against frustration.
- They haven't all been written yet.
- Who really reads `README` files, anyway?

Why Should We Write Manual Pages ("Manpages")?

- We tell people to read them.
- Sometimes we do this without knowing a manpage on a given subject exists in fact.

```
Of 2253 commands, found manpages
for 2000 (253 missing).
```

- `man-db` is `Priority: important` — unlike Web browsers and PDF viewers.
- Manpages are one of users' last lines of defense against frustration.
- They haven't all been written yet.
- Who really reads `README` files, anyway?

Why Should We Write Manual Pages ("Manpages")?

- We tell people to read them.
- Sometimes we do this without knowing a manpage on a given subject exists in fact.

```
Of 2253 commands, found manpages
for 2000 (253 missing).
```

- `man-db` is `Priority: important` — unlike Web browsers and PDF viewers.
- Manpages are one of users' last lines of defense against frustration.
- They haven't all been written yet.
- Who really reads `README` files, anyway?

Why Should We Learn to Write *Good* Manpages?

- Many existing manpages are suboptimal.
- People are unaware of conventions used by high-quality manpages.
- People are ignorant of basic GNU roff (`groff`) syntax and the `an` macro package.
- They don't know lousy `groff`/`an` syntax when they see it.
- They are poorly-equipped to judge the output of engines that generate manpages from another documentation format.
- Lars Wirzenius and I have evaluated several of these; they are mostly horrendous.
- `docbook-to-man` is especially so.

Why Should We Learn to Write *Good* Manpages?

- Many existing manpages are suboptimal.
- People are unaware of conventions used by high-quality manpages.
- People are ignorant of basic GNU roff (`groff`) syntax and the `an` macro package.
- They don't know lousy `groff`/`an` syntax when they see it.
- They are poorly-equipped to judge the output of engines that generate manpages from another documentation format.
- Lars Wirzenius and I have evaluated several of these; they are mostly horrendous.
- `docbook-to-man` is especially so.

What is a manual page?

- A manpage is a text file.
- Manpages are written in the `roff` typesetting language; TEX is another typesetting language.
- Manpages use the `an` and (less commonly) `doc` macro packages, as LATEX is a macro package for TEX.
- A manpage is intermediate in scope between terse user documentation (e.g., usage messages) and book-length manuscripts.
- A manpage is a reference, not a tutorial.

Where are manpages kept?

Sections 1 through 4

- Manpages are kept in subdirectories of `/usr/share/man`.
- Section 1: User Commands (files in `/bin`, `/usr/bin`)
- Section 2: System Interface (e.g., `fork()`, `read()`, `sbrk()`)
- Section 3: Library Interface (e.g., `malloc()`, `printf()`, `XtCreateManagedWidget()`)
- Section 4: Device Interface (files in `/dev`)

Where are manpages kept?

Sections 5 through 8

- Section 5: File Formats (files in `/etc`), e.g., (`fstab`, `xorg.conf`)
- Section 6: Games (files in `/usr/games`)
- Section 7: Miscellaneous (supplementary documentation)
- Section 8: Administrative Commands (files in `/sbin`, `/usr/sbin`)
- Non-English manpages have the same hierarchy in a subdirectory of `/usr/share/man` with a localized name, such as `fr_FR` or `pt_BR`.

What does a manpage look like?
Control Lines, Data Lines, and Escape Sequences

- Lines beginning with a period/full stop (`.`) are control lines, also called requests.

```
.SH SYNOPSIS
One two
.br
three.
```

- In the above, “Synopsis” is an argument to the `SH` request.
- Everything that isn’t a control line is a text line.
- Character sequences beginning with a backslash (\) are escape sequences. These often consist of a left parenthesis followed by two characters.

```
Copyright \(co 2005 Quux \fIItalics\fP Inc.
```

- The `groff(7)` manual page contains detailed discussions of control lines and escape sequences.

What does a manpage look like?
Request Processing

- Some requests take no arguments, some take one, and others take more than one.
- Arguments to requests are separated by whitespace.
- If you give a request fewer arguments than it expects, groff will usually try to muddle through.
- Arguments to a request in excess of those required are ignored.
- Use double-quotes (") to include whitespace in an argument.

What does a manpage look like?
Escaping Literals

- To begin a line with a literal period, use the zero-width non-printing escape sequence `\&` to get the period away from the beginning of the line, which is the only place it is treated specially:

```
\&. This line begins with a dot.
```

- To input a literal backlash, use the escape sequence defined for this purpose:

```
\(rsn often indicates a newline.
```

What does a manpage look like?
Basic Structure

- As with HTML, newlines are not treated literally.
- Put a newline at the end of every sentence.
- This tells groff to use inter-sentence spacing, which is important for proportionally-spaced output formats.
- It also makes life easier for translators (reading a diff is easier for them and you).
- Don't put blank lines in your manpage (except under special circumstances).
- Use the null request — a lone dot — to visually pad your manpage for editing in source form.

```
...long sentence ends.
.
Long sentence begins...
```

What does a manpage look like?
Comments

- Groff supports comments, as any civilized language does.
- The `\"` escape sequence begins a comment that runs to the end of the line. `\" XXX: Fix grammar.`
- `.\" This is a comment on a line by itself.`
- Groff processes the newline in the former example, and ignores it in the latter. Can you guess why? Hint: What does the lone dot request (“.”) do?

What does a manpage look like?
Comments

- Groff supports comments, as any civilized language does.
- The `\"` escape sequence begins a comment that runs to the end of the line. `\" XXX: Fix grammar.`
- `.\" This is a comment on a line by itself.`
- Groff processes the newline in the former example, and ignores it in the latter. Can you guess why? Hint: What does the lone dot request (“.”) do?

What does a manpage look like?

The Line-Break Request

- By convention, requests that are part of the roff language itself are in lowercase, such as `.de`, `.if`, or `.rn`.
- Macros defined by macro packages such as `an`, by contrast, are in capital letters; e.g., `.BR`, `.IB`, and `.PP`.
- There is only one non-macro roff request that you are likely to need: the `.br` request causes a line break without paragraph separation.

The Most Commonly-Used Macros

Paragraphs and Indentation

- `.PP`: new paragraph
- `.TP`: new paragraph with “tag”

```
.TP
.B \-\-help
displays a help message and exits.
```

- `.RS`: begin indented area
- `.RE`: end indented area

```
Examples:
.RS
\(bu one
\(bu two
.RE
```

The Most Commonly-Used Macros

Changing Fonts

- `.B`: bold all arguments
- `.I`: italicize all arguments
- `.BR`: alternate bold and normal (“roman”)
- `.IR`: alternate italics and normal
- `.BI`: alternate bold and italics
- `.RB, .RI, .IB`: inverse order of the above three
- I recommend using these macros in preference to font escapes wherever possible, as they are more readable.

Font Escapes

- `\fB`: switch to bold font
- `\fI`: switch to italic font
- `\fR`: switch to normal (“roman”) font
- `\fP`: switch back to previous font

Dashes, Hyphens, and Quotation Marks

- `-`: hyphenation dash
- `\(en`: en-dash
- `\(em`: em-dash
- `\-`: literal hyphen-minus (codepoint 45)
- `\(oq`: left single quotation mark
- `\(cq`: right single quotation mark
- `\(aq`: apostrophe-quote (codepoint 39)
- `\(dq` or `"`: vertical double quotation mark (codepoint 34)
- `\(lq`: left double quotation mark
- `\(rq`: right double quotation mark

Critical Manpage Macros

Overview

- There are two macros that are *essential* in all manpages, because `mandb` uses them for indexing.
- If you don't get them right, users may not be able to find your manpage with tools like `apropos`.

Critical Manpage Macros

The TH Macro

- The first is `TH` ("text heading"), which declares most of the basic information: name, section, date of last modification, and version.

```
.TH gizmo 1 2005-07-09 "Gizmo 1.0"
```

- In practice, the only mandatory arguments are the name and section.
- Most manpages capitalize the command name. Don't do this — that's a presentation decision (which `man` itself can make), and we use a case-sensitive OS.
- Manpages with the same name can be disambiguated by the section name (e.g., `3perl, 3X11`).
- The section name should match the extension of the manpage's filename as installed.

Critical Manpage Macros

More on the TH Macro

- The date and version arguments to `TH` are inconsistently used in practice.
- I recommend using an ISO 8601 date string and the name and release version number of the (upstream) source package.
- `TH` actually accepts a fifth argument, the “manual title”, but this is inconsistently used, and often duplicates a manual name that is so general that `mandb` should already know it (e.g., “User Commands”).

Critical Manpage Macros

The .SH macro with the NAME Argument

- The “name” section is the other essential part.

```
.SH NAME
gizmo \- GNU frob bletcher
```

- This is the “section heading” macro, which takes one string argument.
- Note the `\-`. Don’t use a plain hyphen-minus.
- Use correct casing for manpage’s name here even if you don’t in the `TH` request.

Critical Manpage Macros

The .SH macro with the NAME Argument

- The “name” section is the other essential part.

```
.SH NAME
gizmo \- GNU frob bletcher
```

- This is the “section heading” macro, which takes one string argument.
- Note the `\-`. Don’t use a plain hyphen-minus.
- Use correct casing for manpage’s name here even if you don’t in the `TH` request.

How to Write a Command Synopsis

Command Name, Options, and Operands

- A command synopsis should document the command's name, operands, and options.

```
.B update\-gizmo
[
.I options
]
.I operand
```

- Again, note the escaped hyphen.
- Put the command name, and anything to be typed literally, in bold.
- Square brackets indicate optional arguments.

How to Write a Command Synopsis

Command Name, Options, and Operands

- A command synopsis should document the command's name, operands, and options.

```
.B update\-gizmo
[
.I options
]
.I operand
```

- Again, note the escaped hyphen.
- Put the command name, and anything to be typed literally, in bold.
- Square brackets indicate optional arguments.

How to Write a Command Synopsis

Repeated and Exclusive Arguments

- Use an ellipsis to note repeated arguments.

```
.IR file " ..."
```

- Use braces (curly brackets) and vertical lines (pipes) to denote that one of an exclusive set of arguments should be specified.

```
.BR "tar " { " create " | " list " |
.BR "extract " }
```

How to Write a Command Synopsis

Incompatible Invocations

- If a command can be invoked in multiple incompatible ways, multiple lines can be used for the synopsis.

```
.BR "gizmo " [ " \-ABCabc " ]
.RB { " see " | " hear " | " smell " }
.br
.BR "gizmo " { " \-\-help " |
.BR " \-\-version " }
```

Headings for Manpages in Sections 1, 6, or 8

- NAME
- SYNOPSIS
- DESCRIPTION
- OPTIONS
- EXAMPLES
- EXIT STATUS
- INPUT FILES (if used)
- OUTPUT FILES (if used)
- DIAGNOSTICS (if any)
- ASYNCHRONOUS EVENTS (if handlers defined)
- CONSEQUENCES OF ERRORS (if unusual)
- AUTHOR or AUTHORS
- SEE ALSO

Headings for Manpages in Sections 2 or 3

- NAME
- SYNOPSIS
- DESCRIPTION
- EXAMPLES
- RETURN VALUE
- CONFORMING TO
- AUTHOR or AUTHORS
- SEE ALSO

Headings for Manpages in Sections 4, 5, 7

- NAME
- DESCRIPTION
- EXAMPLES (if applicable)
- AUTHOR or AUTHORS
- SEE ALSO

How to Portably Include URLs

Recipe

Include this recipe near the beginning of your manpage. I recommend placing it before the `TH` macro.

```
.de URL
\\$2 \(laURL: \\$1 \(ra\\$3
..
.if \n[.g] .mso www.tmac
```

The `.de` request defines a macro called `URL`; the definition is on the next line, and `..` ends it.

How to Portably Include URLs

Using the URL Macro

Here's how to use it:

```
.URL "http://www.debian.org" "Debian" "*"
```

The first argument is the URL, the second is the text to be hyperlinked, and the third (optional) argument is any text that needs to immediately trail the hyperlink without intervening whitespace.

How to Portably Include URLs

Portability

```
.if \n[.g] .mso www.tmac
```

GNU roff defines a `URL` macro; what the above does is test for the presence of GNU roff, and source the `www.tmac` macro definition file (which itself also defines `URL`) if it is — this overrides the definition just made, but leaves it intact for non-GNU roff implementations.

How to License Your Manpage

Recommendations

- If you're writing a manpage for an existing piece of software copyrighted by someone else, it's polite to place the manual page under the same license as the software it documents.
- I recommend this even for GNU software where the TeXinfo manuals may be under the GNU FDL whereas the program code is under the GNU GPL.

How to License Your Manpage

Caveats and Experiences

- The GNU GPL and FDL licenses are incompatible in both directions, so you cannot just take material from the software and put it in the manual page, or vice versa. Only the copyright holder can do this without a license.
- Another possibility in this situation would be to dual-license the manual page under the GNU GPL and the GNU FDL.
- I personally have used the MIT/X11 and GNU GPL licenses for manpages.

Shipping Manpages in Debian Packages

The Simple Case

- Installing manpages that don't require any alternatives handling is not terribly difficult: `dh_installman`

Shipping Manpages in Debian Packages

The Simple Case

- Installing manpages that don't require any alternatives handling is not terribly difficult: `dh_installman`

Shipping Manpages in Debian Packages

The Challenge of Alternatives

- Manpages using the alternatives system require more care.
- They are generally slave links controlled by the resource they document, which is directly handled by `update-alternatives`.
- You need to register the manpages in the `postinst` script of your package.
- You need to unregister the manpages in the `prerm` script of your package, for the `remove` and `deconfigure` arguments *only*.

Shipping Manpages in Debian Packages
Recipe for Handling Alternatives

- Example of `postinst` code snippet:

```
update-alternatives --install /usr/bin/x-terminal-emulator \
 x-terminal-emulator /usr/X11R6/bin/xterm 20 --slave \
 /usr/share/man/man1/x-terminal-emulator.1.gz x-terminal-emulator.1.gz \
 /usr/X11R6/man/man1/xterm.1x.gz
```

- Example of `prerm` code snippet:

```
if [ "$1" = "remove" ] || [ "$1" = "deconfigure" ]; then
  update-alternatives --remove x-terminal-emulator /usr/X11R6/bin/xterm
fi
```

Shipping Manpages in Debian Packages

Recipe for Handling Alternatives

- Example of `postinst` code snippet:

```
update-alternatives --install /usr/bin/x-terminal-emulator \
 x-terminal-emulator /usr/X11R6/bin/xterm 20 --slave \
 /usr/share/man/man1/x-terminal-emulator.1.gz x-terminal-emulator.1.gz \
 /usr/X11R6/man/man1/xterm.1x.gz
```

- Example of `prerm` code snippet:

```
if [ "$1" = "remove" ] || [ "$1" = "deconfigure" ]; then
  update-alternatives --remove x-terminal-emulator /usr/X11R6/bin/xterm
fi
```

Further Reading

- `man(7)`
- `groff(7)`
- `groff_char(7)`

History

This presentation was originally given on 9 July 2005 in Helsinki, Finland at DebConf 5, the 6th[1] annual conference of Debian Developers.

`$Id: wtfm.tex 70 2006-06-27 15:34:18Z branden $`

[1] Yes, there was a DebConf 0.

Acknowledgements

- Suggestions
  - Lars Wirzenius
  - Jimmy Kaplowitz
- Corrections
  - Donnie Berkholz
  - Francesco Poli
  - Glen Smith

Copyright and Licensing