

help2man

A utility for generating simple manual pages

Brendan O'Dea bod@debian.org

Copyright © 1999, 2000, 2001, 2002, 2003, 2004 Free Software Foundation, Inc.

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 permission 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 Foundation.

Short Contents

1	Overview of <code>help2man</code>	1
2	How to Run <code>help2man</code>	2
3	' <code>--help</code> ' Recommendations	4
4	Including Additional Text in the Output	5
5	Using <code>help2man</code> With <code>make</code>	6
6	Producing Native Language Manual Pages.	7
7	Reporting Bugs or Suggestions	8
8	Obtaining <code>help2man</code>	9

Table of Contents

1	Overview of help2man	1
2	How to Run help2man	2
3	‘--help’ Recommendations	4
4	Including Additional Text in the Output	5
5	Using help2man With make	6
6	Producing Native Language Manual Pages.	7
6.1	Changing the Location of Message Catalogs	7
7	Reporting Bugs or Suggestions	8
8	Obtaining help2man	9

1 Overview of `help2man`

`help2man` is a tool for automatically generating simple manual pages from program output.

Although manual pages are optional for GNU programs other projects, such as Debian require them (see [section “Man Pages” in *The GNU Coding Standards*](#))

This program is intended to provide an easy way for software authors to include a manual page in their distribution without having to maintain that document.

Given a program which produces reasonably standard ‘`--help`’ and ‘`--version`’ outputs, `help2man` can re-arrange that output into something which resembles a manual page.

2 How to Run `help2man`

The format for running the `help2man` program is:

```
help2man [option]... executable
```

`help2man` supports the following options:

`'-n string'`

`'--name=string'`

Use *string* as the description for the '`NAME`' paragraph of the manual page.

By default (for want of anything better) this paragraph contains '`manual page for program version`'.

This option overrides an include file '`[name]`' section (see [Chapter 4 \[Including text\]](#), [page 5](#)).

`'-s section'`

`'--section section'`

Use *section* as the section for the man page. The default section is 1.

`'-m manual'`

`'--manual=manual'`

Set the name of the manual section to *section*, used as a centred heading for the manual page. By default '`User Commands`' is used for pages in section 1, '`Games`' for section 6 and '`System Administration Utilities`' for sections 8 and 1M.

`'-S source'`

`'--source=source'`

The program source is used as a page footer, and often contains the name of the organisation or a suite of which the program is part. By default the value is the package name and version.

`'-L locale'`

`'--locale=locale'`

Select output locale (default '`C`'). Both the program and `help2man` must support the given *locale* (see [Chapter 6 \[Localised man pages\]](#), [page 7](#)).

`'-i file'`

`'--include=file'`

Include material from *file* (see [Chapter 4 \[Including text\]](#), [page 5](#)).

`'-I file'`

`'--opt-include=file'`

A variant of '`--include`' for use in Makefile pattern rules which does not require *file* to exist.

`'-o file'`

`'--output=file'`

Send output to *file* rather than `stdout`.

`'-N'`

`'--no-info'`

Suppress inclusion of a '`SEE ALSO`' paragraph directing the reader to the Texinfo documentation.

`'--help'`

`'--version'`

Show help or version information.

By default `help2man` passes the standard `--help` and `--version` options to the executable although alternatives may be specified using:

`'-h option'`

`'--help-option=option'`
help option string

`'-v option'`

`'--version-option=option'`
version option string

3 ‘--help’ Recommendations

Here are some recommendations for what to include in your ‘--help’ output. Including these gives `help2man` the best chance at generating a respectable man page, as well as benefitting users directly.

See section “Command-Line Interfaces” in *standards*, and section “Man Pages” in *standards*, for the official GNU standards relating to ‘--help’ and man pages.

- A synopsis of how to invoke the program. If different usages of the program have different invocations, then list them all. For example (edited for brevity):

```
Usage: cp [OPTION]... SOURCE DEST
      or: cp [OPTION]... SOURCE... DIRECTORY
...

```

Use `argv[0]` for the program name in these synopses, just as it is, with no directory stripping. This is in contrast to the canonical (constant) name of the program which is used in ‘--version’.

- A very brief explanation of what the program does, including default and/or typical behavior. For example, here is `cp`’s:

```
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

```

- A list of options, indented to column 2. If the program supports one-character options, put those first, then the equivalent long option (if any). If the option takes an argument, include that too, giving it a meaningful name. Align the descriptions in a convenient column, if desired. Note that to be correctly recognised by `help2man` the description must be separated from the options by at least two spaces and descriptions continued on subsequent lines must start at the same column.

Here again is an (edited) excerpt from `cp`, showing a short option with an equivalent long option, a long option only, and a short option only:

```
-a, --archive           same as -dpR
      --backup[=CONTROL] make a backup of each ...
-b                     like --backup but ...

```

For programs that take many options, it may be desirable to split the option list into sections such as ‘Global’, ‘Output control’, or whatever makes sense in the particular case. It is usually best to alphabetize (by short option name first, then long) within each section, or the entire list if there are no sections.

- Any useful additional information about program behavior, such as influential environment variables, further explanation of options, etc. For example, `cp` discusses `VERSION_CONTROL` and sparse files.
- A few examples of typical usage, at your discretion. One good example is usually worth a thousand words of description, so this is highly recommended.
- In closing, a line stating how to email bug reports. Typically, *mailing-address* will be ‘`bug-program@gnu.org`’; please use this form for GNU programs whenever possible. It’s also good to mention the home page of the program, other mailing lists, etc.

The `argp` and `popt` programming interfaces let you specify option descriptions for ‘--help’ in the same structure as the rest of the option definition; you may wish to consider using these routines for option parsing instead of `getopt`.

4 Including Additional Text in the Output

Additional static text may be included in the generated manual page by using the ‘`--include`’ and ‘`--opt-include`’ options (see [Chapter 2 \[Invoking help2man\], page 2](#)).

The format for files included with these option is simple:

```
[section]
text

/pattern/
text
```

Blocks of verbatim `*roff` text are inserted into the output either at the start of the given ‘`[section]`’ (case insensitive), or after a paragraph matching ‘`/pattern/`’.

Patterns use the Perl regular expression syntax and may be followed by the ‘`i`’, ‘`s`’ or ‘`m`’ modifiers (see [section “perlre\(1\)” in *The perlre\(1\) manual page*](#))

Lines before the first section or pattern which begin with ‘`-`’ are processed as options. Anything else is silently ignored and may be used for comments, RCS keywords and the like.

The section output order is:

```
NAME
SYNOPSIS
DESCRIPTION
OPTIONS
EXAMPLES
other
AUTHOR
REPORTING BUGS
COPYRIGHT
SEE ALSO
```

Any ‘`[name]`’ or ‘`[synopsis]`’ sections appearing in the include file will replace what would have automatically been produced (although you can still override the former with ‘`--name`’ if required).

Other sections are prepended to the automatically produced output for the standard sections given above, or included at *other* (above) in the order they were encountered in the include file.

5 Using `help2man` With `make`

A suggested use of `help2man` in Makefiles is to have the manual page depend not on the binary, but on the source file(s) in which the ‘`--help`’ and ‘`--version`’ output are defined.

This usage allows a manual page to be generated by the maintainer and included in the distribution without requiring the end-user to have `help2man` installed.

An example rule for the program `prog` could be:

```
prog.1: $(srcdir)/main.c
        -$(HELP2MAN) --output=$@ --name='an example program' ./prog
```

The value of `HELP2MAN` may be set in `configure.in` using either of:

```
AM_MISSING_PROG(HELP2MAN, help2man, $missing_dir)
```

for `automake`, or something like:

```
AC_PATH_PROG(HELP2MAN, help2man, false // No help2man //)
```

for `autoconf` alone.

6 Producing Native Language Manual Pages.

Manual pages may be produced for any locale supported by both the program and `help2man`¹ with the ‘`--locale`’ (‘`-L`’) option.

```
help2man -L fr_FR@euro -o cp.fr.1 cp
```

6.1 Changing the Location of Message Catalogs

When creating localised manual pages from a program’s build directory it is probable that the translations installed in the standard location will not be (if installed at all) correct for the version of the program being built.

A preloadable library is provided with `help2man` which will intercept `open(2)` calls for message catalogs for the domain given by `$TEXTDOMAIN` and re-map the requests to the appropriate file under `./po` (or `$LOCALEDIR` if given).

So for example:

```
LD_PRELOAD="hacklocaledir.so preloadable_libintl.so" TEXTDOMAIN=prog \
  help2man -L fr_FR@euro -i prog.fr.h2m -o prog.fr.1 prog
```

will translate requests by `prog` for ‘`/usr/share/locale/fr/LC_MESSAGES/PROG.mo`’ to ‘`po/fr.gmo`’.

Notes:

- The generalisation of ‘`fr_FR@euro`’ to ‘`fr`’ in the example above is done by `gettext`, if a more specific match were available it would also have been re-mapped.
- The inclusion of `preloadable_libintl.so` in `$LD_PRELOAD` is required only for cases (such as `glibc`) where `gettext` is built into `libc` (where `__open` would otherwise be satisfied internally).
- This preload hack has only been tested against `glibc` 2.3.1 and `gettext` 2.3.1 on a GNU/Linux system; let me know if it does (or doesn’t) work for you (see [Chapter 7 \[Reports\]](#), page 8).

¹ `help2man` currently supports ‘`fr_FR`’ and ‘`pl_PL`’ (see [Chapter 7 \[Reports\]](#), page 8 for how to submit other translations).

7 Reporting Bugs or Suggestions

If you find problems or have suggestions about this program or manual, please report them to bug-help2man@gnu.org.

Note to translators: when submitting new translations for `po/help2man.pot` please additionally translate `help2man.h2m` (used to augment the manual pages for `help2man`).

8 Obtaining `help2man`

The latest version of this distribution is available on-line from:

<ftp://ftp.gnu.org/gnu/help2man/>