How to Write a Manual Page

Henry Spencer

Zoology Computer Systems
University of Toronto

Introduction

This document is an attempt to outline how to write a UNIX® manual page, using the U of T Zoology variant of the UNIX Version 7 manual-page macros. Of necessity, every last detail cannot be covered; judgement and good taste are still necessary for writing readable documentation, especially when striving for clarity in the extremely concise format encouraged by UNIX.

Manual pages are stored as sources suitable for input to *troff*; *nroff* is compatible enough with *troff* that good *troff* formatting is generally good *nroff* formatting. The ‘‘manual page’’ is stored as a single file even if the printed form of it is several pages long. The formatting macros used are summarized in *man*(7); a lot of this document is just a more detailed explanation of them.

Title Heading Information

A normal manual page consists of a title heading, several subject headings, and indented text paragraphs. (*Beware*: some of the details of title heading format are local to U of T, and may not be the same in printed manuals from Bell.)

Normally the very first thing in a manual page is the `.TH` macro, with three arguments: the *name* of the manual page, the *chapter* in which it is to be filed, and the *date* it was last modified. For example:

```
.TH MAN 7 "8 March 1990"
```

Name and *chapter* appear in the upper left and right corners of all manual pages, with *chapter* parenthesized. By convention, *name* is in capital letters. *Chapter* is usually a number, possibly followed by a (capital) letter.

Date must be given in the format illustrated above, *day month year*, and must be enclosed in double quotes so it is taken as a single argument. This should be the date when the last significant change was made to the page; correction of typographical errors etc. doesn’t count. The date is printed at the bottom center of the manual pages, with a ‘‘+’’ appended if the source file has been changed more recently.

The third field can be *commentary* instead of a date, and is in fact taken as commentary if it does not seem to be a date in the prescribed format. If the *commentary* form of the third field of `.TH` is used, there should then be a `.DA` macro giving the *date* of the manual page. `.DA` takes three arguments, respectively day, month, and year. Some old manual pages do things this way, but this form is not recommended for new pages.

The top centre of a manual page is occupied by the name of the *manual* to which it belongs; this is usually the *UNIX Programmer’s Manual*, which is therefore the default. If you are writing a manual page for some other manual, you must specify the name of the manual *before* you give the `.TH` title header. To do this, use the `.MA` macro:

`.MA` *manual* [*commentary*...]

The optional *commentary* has the same meaning as for `.TH`; it need not be given there if it is given in `.MA`. Remember to put *manual* in double quotes if it contains blanks. An example:

```
.MA "Our Own Strange UNIX Manual"
.TH FNS 9 "8 March 1990"
```

Section Headings

The major sections of a manual page are set off by section headings, such as NAME, SYNOPSIS, DESCRIPTION, FILES, SEE ALSO, BUGS, etc. etc. Some fairly specific conventions are followed* as to what sections are present and what information they contain; see later for details. By convention, subject headers are written in capital letters.

By convention, if a section is empty its section header does not appear (Bell Labs does not consistently follow this convention, but we do).

A section header is given, and a section begun, by a `.SH` macro with the subject header as its argument. The header is printed at the left margin; the section is indented a short distance. Thus:

```
.SH NAME
text lines
   .
   .
   .
```

produces:

NAME
text lines
.
.
.

Paragraphs

Sections consisting of text are normally a sequence of paragraphs. Simple paragraphs are separated by `.PP` (synonym: `.LP`), which outputs a small vertical space, checks that enough paper remains on the page for a few more lines, and resets indents and the like (notably, the ''prevailing indent'', a mystical number whose significance will appear soon, is set to 0.5 in.).

The `.HP` macro is like `.PP` except that it produces a ''hanging'' paragraph, with all lines after the first indented farther than the first. The amount of extra indent is taken from the ''prevailing indent''; if `.HP` is given an argument, the prevailing indent is first set to the value of the argument (the value is taken to be in characters, *troff* ens).

There are two ways to produce a ''tagged'' paragraph: a paragraph whose text is indented an extra amount and whose first line is preceded, in the space of the extra indent, by a *tag*, a short piece of text. Such paragraphs are used whenever it is desirable to discuss a list of items (e.g. command options, with each tag an option).

The `.TP` macro begins a tagged paragraph; the next text line is the tag. The text thereafter is indented by the prevailing indent. If `.TP` is given an argument, the prevailing indent is first set to the value of the argument. The `.IP` macro is similar, except that its first argument is the tag and its second (optional) argument is the prevailing-indent setting.

It is not necessary to explicitly begin a new paragraph after a section header.

Indenting

The `.RS` and `.RE` macros begin and end *relative indents*. These are pieces of text indented beyond the indent of the material above and below.

`.RS` begins a relative indent; if an argument is given, this is the amount to indent (default: prevailing indent). It saves the old value of the prevailing indent and sets the prevailing indent to 0.5 in.

`.RE` ends a relative indent, or possibly more than one. It restores the previous indenting level and the previous prevailing indent. (If it is given an argument, this indicates the relative indent to be ended, where indents are numbered as they are nested, starting at 0 for no relative indent at all.)

* In the Unix Programmer's Manual, that is; other manuals may have different rules.

Tabs and Tables

The default tab stops are every half inch. It may occasionally prove useful to change them with *troff*'s `.ta`, to facilitate construction of things like multicolumn tables. The `.DT` macro resets the tab stops to the defaults.

Usually, a better way to include tables is to use *tbl*(1). Our *man*(1) command's auxiliaries automatically invoke *tbl* if it is used in the manual page. Tables are introduced by `.TS` and concluded by `.TE` as usual with *tbl*. The `.TS H` form for multi-page tables is not supported at present.

Typographical Requests

Manual pages make fairly heavy use of **bold** and *italic* type fonts. To simplify writing manual pages, there are a number of macros which do font changes for you. All of them change the font of a small piece of text; the text is either the arguments to the macro (up to 6 words or strings in quotes) or, if no argument is given, the next text line. None of these macros cause a break, so they may be used anywhere in text.

`.B` and `.I` provide bold and italic text respectively. Situations often arise where it is necessary to have one part of a word in one font and another part in another; examples:

X's, **Y**'s
troff(1)
−w*nnn*

For such cases, there are several macros which merge the words from their text input into a single word, alternating from one font to another from word to word. `.BI` alternates between bold and italic. `.BR`, `.IB`, `.IR`, `.RB`, and `.RI` exhaust the remaining combinations of Roman, Bold, and Italic. The above examples could have been produced by:

```
.BR X "'s," Y "'s"
.IR troff (1)
.BI \-w nnn
```

There is also provision for asking for a slightly smaller point size, as in the usual typeset way of writing ''UNIX''. The `.SM` macro puts its text input out one point size smaller than usual. (This is of course invisible in *nroff*.) If you want (say) small bold letters, do it like this:

```
.SM
.B
letters
```

If you want to do anything really tricky with point sizes and fonts, or it you simply wish to change fonts in the middle of a line without breaking it up to use the font-change macros, it is quite all right to use the *troff* `\f` etc. sequences. The new-paragraph macros reset font and point size; the predefined *troff* string `\*S` resets the point size.

Typing Conventions

Quotation marks are written using pairs of left and right quotes (‘‘) (’’) rather than the double quotes ("). There are three reasons for this. First, the double quotes are often used to enclose macro arguments; there is no way to put them inside such arguments. Second, output on devices like phototypesetters looks much better that way. Third, the paired-quotes convention *is* the more correct English usage (it is unfortunate that most typewriters do not recognize this, and that terminal and character-set designers have not put the matter right).

Remember that the hyphen, the dash, and the minus sign are three different characters—even though most computer-terminal keyboards only have one key for all three. For *nroff* and *troff* input, the hyphen is just ‘‘-’’, the dash is ‘‘\(em’’, and the minus sign is ‘‘\-’’. Be careful to get these right.

More generally, read and remember the *nroff/troff* escape sequences for special characters (Table II at the end of the *NROFF/TROFF User's Manual*); use them rather than trying to construct the character by hand. Hand-constructed characters won't come out right on things like phototypesetters, while *nroff* and

troff will do their best to print escape-sequence characters properly on any device.

Unix Programmer's Manual Conventions

The Unix Programmer's Manual (UPM) is the major document using the manual macros. In addition to what has been described about the use of the macros, the UPM defines a number of conventions which are usually followed when writing manual pages for it. Other manuals may have different conventions.

The *name* argument to `.TH` is usually the name of the entity (program, subroutine, etc.) that is being described. Sometimes a manual page describes several entities; in such cases, the one used in `.TH` is the first one in the NAME list (to be discussed in a moment). A manual page which is not associated with any particular program, subroutine, etc., is named by what it describes: preferably one short word (e.g. *crash*(8)). Occasionally this convention is also applied to a manual page describing a large number of related programs (etc.); for example, *string*(3) describes a bunch of string-handling routines.

The *chapter* in `.TH` is the chapter number. The precise meaning of the chapter numbers has changed somewhat from one version of UNIX to another; the set for UNIX version 7 is:

1 Commands (i.e. programs)
2 System calls
3 Subroutines
4 Special files (i.e. devices)
5 File formats and conventions
6 Games
7 Macro packages, language conventions, misc.
8 Maintenance procedures and other esoterica

A more detailed explanation of these chapters can be found in the introduction to the UPM.

Chapter numbers are sometimes suffixed by one or more letters; these mark the manual page as being part of some special subset of pages. Examples: commands in chapter 1 which are related to system maintenance have chapter number ''1M''; the subroutines of chapter 3 are subdivided into a number of categories depending on their use.

The order and contents of the various sections of a manual page are quite stereotyped in the UPM. The sections will be discussed individually in the order they generally occur.

The first section, present in all entries, is NAME. This lists the exact names of the things discussed in the entry, and says a few words about their purpose. If there is more than one name, the first should be the one which best evokes visions of the whole list, since that will also be the name of the manual entry as a whole. (It is agreed that this criterion is a bit vague.)

The name(s) (separated by commas) are followed by a minus sign (*not* a hyphen, *not* a dash—see *Typing Conventions*), which is followed by the purpose. Keep it brief, on one line if humanly possible; avoid font changes, special symbols, and cryptic buzzwords. (The NAME section is used by other programs, such as the one which prepares the on-line index for the manual—hence the insistence on brevity and the rigid format.) Examples (as written, not as printed):

```
split \- split a file into pieces
grep, egrep, fgrep \- search a file for a pattern
tar \- tape archiver
awk \- pattern scanning and processing language
```

Multiple lines may appear in this section if necessary, separated by `.br`.

Next, on most manual pages, is the SYNOPSIS section. This is absent only in manual entries which are not discussing identifiable programs, subroutines, etc., but rather general concepts like what to do when the system crashes. The rule is: if there is any conceivable way to *type* it, the synopsis section should specify how.

Particularly for commands, a few conventions are used to indicate what is going on:

Boldface things are literals, to be typed just as they appear.

Non-bold things are placeholders, indicating a place where a specific sort of thing is to be typed (e.g. a number).

Square brackets [ ] around something mean that it is optional.

Braces { } around several things indicate that one and only one should be chosen.

An ellipsis ‘‘...’’ means that the previous thing can be repeated.

An argument given as ‘‘name’’ generally means a filename.

One thing to remember when writing the synopsis for command options: the minus that is often used to indicate an option is a *minus*, not a hyphen or a dash. Some simple examples, each followed by its printed appearance:

```
\fBawk\fR [ \fB\-F\fIc\fR ] [ prog ] [ file ]
```

awk [**−F***c*] [prog] [file]

```
.B df
[
filesystem
]
```

df [filesystem]

```
.B split
[
.BI \- n
] [ file [ name ] ]
```

split [−*n*] [file [name]]

As you can see, either the font-change macros or the `\f` escapes can be used. There is not a great deal to choose between them; opinions vary as to which is more readable when editing the manual page.

The above examples, taken from existing manual pages, tend to leave spaces between brackets and the bracketed items. This is a historical practice which should perhaps be discouraged because it increases the bulk of the SYNOPSIS section without significantly improving readability.

The same sort of conventions apply to SYNOPSIS sections for things other than commands, although such sections tend to use boldface almost exclusively, since there is seldom much choice about how to call a subroutine. If a manual entry describes more than one program, subroutine, etc., the synopses are separated by `.br` or `.PP` depending on whether space between the items is desirable (usually not if the items are one line each, usually so if they are multiple lines).

The next section is the big one, present in all manual sections: DESCRIPTION. This is typically several paragraphs of narrative text describing the details of what goes on. It is helpful if the first paragraph is a capsule summary of what the program (subroutine, etc.) does and what its inputs and outputs are.

Within narrative text in a manual entry, the basic rules are those of good English: clarity and conciseness. Paragraphs should be short. Tables, lists, etc. should be used whenever they make something clearer.

Frequently a narrative has cause to name programs, variables, macros, etc., and to reproduce pieces of the SYNOPSIS section. There are some simple typographical rules for helping the reader tell what’s what.

Pieces of the SYNOPSIS are reproduced as they occurred, complete with font changes etc.; the same applies to any place where a similar notation is useful in expanding on what is meant by something mentioned in the synopsis.

There is one exception to this: names of programs, subroutines, files, and variables, even the ones described in the synopsis, are treated like foreign words: they are written in italics. Such names are capitalized if they fall at the beginning of a sentence, although it is perhaps better to try to avoid such situations. The ''italics'' rule applies even to name-and-chapter references in text; within a DESCRIPTION section, the proper way to refer to the manual entry for, say, the mail program, is *mail*(1). (Note that this rule does *not* apply to non-narrative material, such as tables.)

Names of constants, *troff* macros, and shell environment variables are generally written in boldface or literal. There is much less consistency about this than about the italicized names.

Many manual entries have a FILES section, giving the names of the files which are built into the program. The names are generally given one to a line, with a comment following indicating what the file's significance is. This is a place where non-default tab stops are useful, because such a description is much easier to read if everything is in two neat columns. The preferred technique is to set the tab stop, not to an absolute number, but to slightly more than the width of the longest entry, using *troff*'s width-finding `\w` primitive:

```
.ta \w'the longest entry'u+2n
```

Many manual entries have a SEE ALSO section giving pointers to related information: sometimes references to various documents, often just a list of other manual entries. Document names are italicized, as in any proper bibliography; names of manual entries are usually written without italics, although this is inconsistent with the italics used when referring to them in text.

A DIAGNOSTICS section explains any diagnostics which are not thought to be self-explanatory. When in doubt, explain. If all diagnostics are considered self-explanatory, there is no need for a DIAGNOSTICS section at all: a section which just says ''Intended to be self-explanatory'' is useless clutter.

A variety of other sections sometimes sneak in at this point when the author feels like it: EXAMPLES, WARNINGS, RANGES AND LIMITATIONS, CAVEAT, etc. Avoid verbosity: one of the major virtues of the UNIX manual style is its compactness.

Next, in manual entries for things not taken straight from Bell, comes the HISTORY section, whose purpose is to record authorship (for local products), modification history (for Bell products), and origin (for non-Bell products). This section is *mandatory* for any Bell program which has been changed locally, even for user-invisible bug fixes. Local improvements to Bell documentation should also be mentioned here.

Finally... If the program should (horrors) have any bugs, or any non-obvious deficiencies, or even anything which ''should be done better'', there is a BUGS section. This is a sequence of (generally one-line) narrative paragraphs, more-or-less one to each bug/quirk. To encourage full reporting, it is policy that mentioning something in a BUGS section does not imply a commitment to fix it (or even the intent to; deficiencies like limits on the size of data are often more trouble to fix than is justified). This is the place to mention things which are unsatisfactory or tricky about the program, even if it is not clear that they are bugs.

In general, if in doubt as to how to format something, it is better to look for an existing manual page and imitate it rather than inventing a new and unique style; standardization of style is a strong aid to readability, and is not usually a barrier to communication.

Invocation

The manual-page macros can be invoked with the **−man** option to *nroff*, *troff*, *ms*, or *typeset*. The auxiliaries for *man*(1) do some slightly more involved trickery to pick up things like Sun comment lines, but *ms* output is a good approximation to the final form of the manual page.

Template

It is usually simpler to start writing a manual page based on a ''boilerplate'' template, which can be added to or altered as needed, rather than typing it in from scratch. Such a template can be found in */usr/pub/template/man*; as of May 1990, it looks about like this:

```
.TH NAME CHAPTER "dd month yyyy"
.SH NAME
name(s) \- what it does
.SH SYNOPSIS
.SH DESCRIPTION
The first sentence should summarize what it does.
The rest of the first paragraph should elaborate.
.PP
Discuss whatever else needs discussing in further paragraphs.
.SH FILES
.SH SEE ALSO
.SH DIAGNOSTICS
This section appears ONLY if there is something unobvious
and important about the diagnostics or the general behavior
in case of error.
.SH HISTORY
Origin (if non-Bell), history of local work and changes.
.SH BUGS
Including things which are less than perfect but not worth
fixing.
```

Style and Tone

The UNIX manual pages, by intent, are a reference manual, not a tutorial. About the only thing really wrong with the classical manual pages is that they could do with more examples in tricky areas. Avoid long explanations aimed at beginners.

Be concise. Complex packages should be summarized tersely in a manual page and documented fully in a separate document. Use cross-references rather than duplicating material from other manual pages, unless the material in question is central to your explanation.