Creating Your Own MAN Page Version 1.0

contributed by Harold Rodriguez (X_console)

So you have finally finished your long awaited program! After months of coding in some cryptic programming language, version 1.0 is ready to be distributed to the masses. Problem is... what does your program do? How does it work? What options does it have? Got a manual page? No? How will you ever tell all those people who email you with questions like, "How do I use program foo???" with "RTFM!"? Ah... you need a manual page. This tutorial will give you a quick introduction into writing manual pages. It is not hard. With a little patience, everything will be fine. But first, why bother with a manual page? It is after all, much easier to simply just create a plain old ASCII text README file. But it is also much simpler to just type man command, rather than searching for your README file which could very well be located anywhere. And so, manuals make this easier. No need to search for anything at all. Just type the command, and you have it.

SECTIONS

Manual pages have a particular format. When you run the manual page for any program, you will normally find some very common sections. These sections are in bold and capital letters. They are, in order of appearance:

NAME
SYNOPSIS
DESCRIPTION
OPTIONS
BUGS
AUTHOR
SEE ALSO

These sections appear 90% of the time in any descent manual page. You will no doubt find other sections, though they are not the very common ones, and are added for extra information for the program. Now for a description on what each section is used for:

NAME

This is the name of the program, followed by a short (read: short) description of what it does or what the name stands for if it is in an acronym. This will be read by apropos, man -k, makewhatis and whatis.

SYNOPSIS

Basically, this is the syntax used to run your program from the command line. An example would be like: foo [-d] [-e] [-f filename]

DESCRIPTION

This is where you describe what the program does. Because this is what the user is most likely going to look at first, it is important that this section is clear. If the user does not understand what is written here, you can expect your inbox to be filled with emails.

OPTIONS

If your program uses options, as in the above SYNOPSIS section, this is where you explain what each option does. For instance, what does [-d] do when it is given as an argument to program foo? Explain here.

BUGS
If there are any bugs in your program, or anything that does not work the way you want it to, put it here so the user knows.

AUTHOR
Your name followed by your email address. Your email address is important here for a couple of reasons. Firstly, people can give you bug reports so you can fix up your program. Secondly, if you get stupid emails, you can reply with RTFM.

SEE ALSO
If your program is related in a way to another program, specify the program's name here, followed by its section number. For instance, program foo is a text editor. You might want to make references to say, the vi text editor as: vi(1)

Now that you have a much better understanding on the format of the manual pages, it is time to learn how the manual page is printed to the screen when the man command is given.

HOW IT WORKS

When you run the command man foo, what actually happens is that man runs the groff command. Suffice it to say that groff is a text formatter that reads special macros in a file and outputs a formatted file, depending upon the macros used. Read the manual page for groff for a more informative explanation on how to use it. We will be using groff macros to create our manual page. These macros always start with a dot: "." followed by the macro name, and its parameters. Now that you know how it basically works, we can (finally) begin the actual coding of the manual page. I will be showing portions of source code, but I will not be showing the output of the manual page. You are expected to code it into your text editor, and run it through the required groff command, shown later on.

TITLE HEADER

We begin with the .TH macro. This macro expects at least five parameters in the order of:

```
.TH [name of program] [section number] [center footer] [left footer] [center
header]
```

Now for the explanation:

```
[name of program]
```

This is obviously the name of your program. It will be on the left and right header of each page.

```
[section number]
```

Manual pages are kept in sections. If you check /usr/man you will find up to nine manual directories. These are the sections. Each section holds a specific type of manual page:

* Section 1: user commands.

* Section 2: system calls.
* Section 3: subroutines.
* Section 4: devices.
* Section 5: file formats.
* Section 6: games.
* Section 7: miscellaneous.
* Section 8: system administration.
* Section n: new.

  So if your manual page is for a game, then you will use section 6. If it is a system administration program, then you will use section 8. The section number will appear beside the name of the program in brackets: foo(1)

```
[center footer]
```

You can write anything that you like here and it will be displayed at the center of the footer of every page. Normally you put the date here.

```
[left footer]
```

You can write anything that you like here and it will be displayed at the left footer of every page. Normally you put the version number of your program here.

```
[center header]
```

You can write anything that you like here and it will be displayed at the center header of every page. Most manual pages have this omitted.

Here is an example for the title of our program foo:

.TH foo 1 "14 May 1999" "version 1.0"

As you can see, we have ommited [center header]. You can actually omit anything you like, but it is best to have the first four in the manual page. Pay attention to the use of quotations. If you need to have whitespace in a particular section, use quotations to keep the macro from getting confused. This applies to all macros.

SECTION HEADER

As I explained earlier, the manual pages are divided into sections. These sections are defined with the .SH macro. For instance, the first section is always NAME. .SH requires just one parameter:

```
.SH [section name]
```

.SH will always have [section name] converted to bold lettering. Text written below .SH will be indented. So let us take a look at our current manual page with the .TH and the .SH macros:

```
.TH foo 1 "14 May 1999" "version 1.0"
.SH NAME
foo - my own text editor
```

Note that the - is required to make the dash distinct from hyphens. Type all that into your text editor, and save it as foo.1. To view it as a manual page, type:

xconsole$ groff -man -Tascii ./foo.1 | less

Optionally, you may use the man command itself to view the manual page you have created:

xconsole$ man ./foo.1

You have just created your first manual page. If you scroll all the way down, you will find the version number, the date, of your program, and the page number. Believe it or not, this is all you need to know to write a manual page. Of course, this will be a very simple manual page... but a manual page nonetheless. You will need to add more sections using the .SH macro until you are satisfied.

FONT ATTRIBUTES

We continue with font attributes. The simplest ones are bold and italics. The macro for bold is .B and the macro for italics is .I. Depending on your system, italic fonts may appear as underlined text instead of actually italicized text. Normal font (no bold or italic), is called Roman. Let us further modify our manual page to look like as follows:

```
.TH foo 1 "14 May 1999" "version 1.0"
.SH NAME
foo - my own text editor
.SH SYNOPSIS
.B [-d] [-e] [-f
.I filename
.B ]
```

Run it through groff again and study the result. It is important to remember that each macro only affects the parameters that are passed to it. That is why in order to italicize filename, it has to be put on a new line with the .I macro prefixing it. This is how we mix bold and italics. Then to have the closing brace in bold, it has to be put on its own line prefixed with .B. However, it is possible to have bold and italics alternating on each parameter. That is:

bold italic bold italic...

Let us say that you want to have bold, followed by italics over and over. The macro for it would be .BI Try it with the following line:

```
.BI This is the foo text editor.
```

If you run this, you will find that you have some strange output. For one thing, there does not appear to be any white space. In order to have white space, you need to use quotations:

```
.BI "This " "is " "the " "foo " "text " "editor."
```

Notice that the quotations have whitespace trapped inside. This is how the whitespace is created. Now it runs correctly. This is useful in speeding things up sometimes. All the font macros can be mixed in this way. Here is a list of font macros:

```
.B = bold
.BI = bold alternating italic
.BR = bold alternating Roman
.I = italic
.IB = italic alternating bold
.IR = italic alternating Roman
.RB = Roman alternating bold
.RI = Roman alternating italic
```

COMMENTING

As you can see, creating a manual page has suddenly become a little more complicated. Fortunately, like in any programming language, one can comment the source code for the manual page. Comments are prefixed with ", as in:

```
." This is a comment
```

It is a good idea to comment your code so if anyone wants to modify it, they will know what it is they are modifying.

PARAGRAPHING

Simple paragraphing is done with the .PP macro. Take for instance the following in our manual page:

```
.SH DESCRIPTION
.B foo
is a text editor that I wrote. It is extremely simple to use,
yet powerful enough to compete with other editors like
.BR vi "(1) and " emacs "(1). Instead of making use of the CTRL and ESC keys, "
.BR foo " makes use of the F1-F12 keys, thus making it simple."
." next paragraph
.PP
Note however that
.B foo "(1) is still in Beta testing mode, and may not work as expected. "
If you have any problems with it, please feel free to email me and let me know.
If you wish to join the development team, check out our website at
.B http://foo.bar.org
```

When you run this through groff, you will see that you have two paragraphs now.

Another form of paragraphing is relative indent paragraphing. This is called as such because it indents all following paragraphs by 0.5 inches to the right. This is normally used in the FILES section (if you have it), or in other ways. There are two macros for this. The first is for relative indent start: RS, and the second is relative indent end: RE. Here is an example:

```
.SH FILES
.I /etc/foorc
.RS
Global system wide configuration file.
.RE
.I $HOME/.foorc
.RS
Local system configuration file.
.RE
```

And that is how relative indenting is done. Another paragraphing macro to add to your collection is IP. This is great for the OPTIONS section. What it does is takes one parameter. Anything on the next line will be indented and tabbed to the right. As always, here is an example you are expected to try out:

```
.SH OPTIONS
.IP -d
disable syntax highlighting
.IP -e
enable syntax highlighting
.IP "-f filename"
Specifies the file you want to edit. This option must be given with the
.BR " -e " "or the " "-d " option.
```

.IP has the options in Roman font. An alternative to .IP is .TP, which is the tag paragraph. It gives you more control in the sense that you may want the option, say, -e to be bold:

```
.SH OPTION
.TP
.B -d
disable syntax highlighting
.TP
.B -e
enable syntax highlighting
.TP
.BI -f " filename"
Specifies the file you want to edit. This option must be given with the
.BR " -e " "or the " "-d " option.
```

Here we see that we have made all options bold, with filename italicized. More control.

TRYING IT OUT

By now, you should have the knowledge to create a manual page. The manual page that we created here is incomplete of course. It is just an example after all. Now it is time for you to write your own manual page. When you have completed it, you will need to compress it with gzip, and then copy it to the respective manual page directory. Finally, run makewhatis to add your manual page to the whatis database:

```
root# gzip foo.1
root# cp foo.1.gz /usr/man/man1
root# makewhatis
root# whatis foo
```

foo (1) - my own text editor

Now run the command man foo and you should be presented with the manual page for foo. You will also want to try it out with the apropos and the man -k command to make sure everything works the way it is supposed to.

CONCLUSION

Just some pointers here before we close off. When writing your manual page, try to keep it standardized, and similar to other manual pages. For instance, do not use FLAGS instead of OPTIONS. No need to confuse people. The manual page is supposed to be clear. Have it proof read to make sure there are no errors. Make sure the SYNOPSIS is correct, otherwise people are going to wonder why certain options do not work. In short, do it right.

Harold Rodriguez .:. X_console
Email Address .:. xconsole at it.yorku.ca
World Wide Web .:. http://it.yorku.ca/moonfrog