The Groff and Friends HOWTO

Dean Allen Provins

University of Calgary
dprovins@ucalgary.ca

1 Introduction

This HOWTO attempts to introduce the document formatting system known as *groff*. It isn't intended to be a reference document; other individuals are working on that. It is aimed at two groups: firstly, the individuals who are new to UNIX, Linux or another UNIX variant, and are discovering that operating system's enormous potential and versatility; and secondly, at those who are considering the use of a document formatting system for producing high quality and consistently formatted papers, articles and possibly books.

groff is not the only document formatting system available to writers. In recent years there have been developed "What You See Is What You Get" or **WYSIWYG** types of programs such as the many that are available under Microsoft's various operating systems. Other similar programs include those available from Sun Microsystems or Applix. In fact, these are not, or at least haven't been, document formatting programs so much as tools to generate shorter works, with their emphasis being on look, rather than content. The only true alternatives (known to this writer) are *troff* and the versatile TEX system with its accompanying macro packages such as LATEX.

As this writer understands the history of text formatting, the predecessor to TEX was *troff*. It was developed at AT&T Bell Labs and was in use from about 1971 according to [Kernighan 1978]. *troff* and related programs were provided as part of the **Documenter's WorkBench**, an optional extra for those purchasing a licence to use UNIX. In the late 1980's, the Free Software Foundation's version of these tools was developed, and was named *groff*. At about the same time (possibly a little earlier), the TEX formatting system was developed.

Like *troff/groff*, TEX and subsequently LATEX, use text formatting commands interspersed with document text to describe how a page is to be formatted by a type setting device.[1] *troff/groff* however, tend to be much more like typical UNIX applications in that the commands used to describe page formatting are terse. These commands are typically 3 characters in length and may be combined in sometimes difficult to read combinations (not unlike the regular expressions used with **sed** or **grep**). The macro packages available for use with *troff/groff* are somewhat more easily read, but are still brief. The advantage offered by TEX is that the commands are much more descriptive or readable. Their disadvantage (in the opinion of this writer) is that they take more time to type. Both however offer writers the opportunity to focus on content, without the distraction of look.

While the early systems were designed to drive specific devices, this is no longer the case. Both systems generate a neutral device-independent page description which subsequent processes format for specific devices. This is often via **PostScript**, which on UNIX or UNIX-clone systems is easily formatted for a wide variety of available printing and plotting devices, but need not be. Today, plain text, TEX format, PCL5 (HP's printer description language) and HTML are available.

Although I have acknowledged the availability of TEX and LATEX, this document will not dwell on them, or compare *groff* to them.[2] For further information on TEX, interested readers are referred to [Lamport 1985]

[1] The original *troff* program was designed to drive a specific machine, a Graphic Systems Incorporated C/A/T (Computer Assisted Typesetter) [Kernighan 1978; Sun Microsystems 1988].

[2] At least not in this revision.

and [Knuth 1986].

This document then will describe how one can profit by the use of *groff* to generate high quality articles, reports and even books (many of which have been written using *groff* or *troff*. See for example [Dougherty 1987; Kernighan 1974; Kernighan 1976] and [Sequin 1975] to name a few).

2 Disclaimer

This document is intended to be one that assists new users by introducing them to *groff* and friends. It is not intended to be a definitive reference manual. Other documents cover these subjects very well, and are available either with the current distribution or at a web site identified in the sequel. *groff* comes with well-written **man pages** that describe the differences between *groff* and *troff*. These with the aforementioned reference documents are all that have been available for some time. Under development however, are complete reference documents for the *groff* suite.

Given the nature of this document, and the availability of other documentation, please do not hold me responsible for any damages that you or your organization may incur as a result of using the *groff* suite or this HOWTO. As with other free software, you use it at your own risk. No one associated with this software, this document, or any others mentioned herein has intended that you incur any damages. All products described here are made available freely, with the hope that they benefit you as much as they have benefited us.

3 Credits and Copyright

The *groff* document preparation suite was written by James J. Clark who re-implemented the *troff* system in some 60,000 lines of C++ code. Since then he has gone on to develop SGML parsers and related processors in both C and C++. All users of *groff* owe considerable thanks to his not inconsiderable efforts.

The current suite of programs are being maintained by Werner Lemberg and Ted Harding, with contributions by many others. As this document is being written, the current release of the suite, with enhancements for HTML and graph rendering, is version 1.17.x.

Some of the current and past contributors to this document, either directly or indirectly, plus contributors to the *groff* suite in general are[3]:

Known Contributors to the *groff* Suite

Nicola Bernardini	discussion on .so and .mso
James J. Clark	groff's author
Jeffrey Copeland	duplex printing
Paul Eggert	various patches including Y2K compliance
Trent Fisher	tex.info format reference manual
Joerg Freudenberger	man page discussion
Thor M. Gil	Windows 95/98/NT binaries for *groff*
Jörgen Hägg	new **mm** macros (version 1.32)
Robert Herrmann	groff gem contributions (posters, business cards, etc.)
Ted Harding	user assistance and smallcaps macros
Steve Izma	how to defer determining page numbers in floating keeps
Larry Jones	Gxditview additions, including grey levels
Susan G. Kleinmann	Additional man pages
Werner Lemberg	release coordination
Eddie Maddox	web site maintainer
Blake McBride	WIN32 port of groff and friends

[3] The following names have been gleaned from the *groff* mail list, and the *groff* documentation. If you do not appear here, but have contributed to the success of *groff* and friends, please let me know. I'll be glad to include you.

Peter Miller	soelim path control
Gaius Mulley	grohtml and tmac.arkup for HTML generation
Bernhard Reiter	mail list manager
Paco Andres Verdu	patches for the Canon capsl printer
Bernd Warken	programming long options for groff, man page discussion and page creation

4 *ms* Macro Overview

As indicated earlier, there are several *macro* packages available for *groff*; but what is a macro package? Think of it as a high level page description, which is written in a low level page description language. An analogy to a macro package might be a procedural language like C, which is relatively easy to read, but which (when compiled) is turned into a low level language like assembler, and ultimately machine instructions. Few people code in assembly language any more,[6] preferring to use higher level instruction sets (which incidentally greatly improves productivity).

In a similar manner, a macro package for *groff* such as the package of *ms* macros[7] is a collection of statements, which when interpreted are translated into low level *troff* instructions. You are welcome to use only the low level instructions in your documents, but you will likely find that the use of a package like *ms* greatly facilitates your productivity. On the other hand, you may find that the macro packages available don't meet your specific formatting needs, and you may wish to alter an existing package, or write your own package.

In this document, you won't see any new macro package development, but later I'll show you how one might customize some of the *ms* commands, or add to that package.

When *groff* invokes the GNU version of *troff*, it causes the replacement of the high level macro commands that you introduce to describe how you have organized your text (into chapters, sections, tables and so on) into low level formatting commands. These are interpreted by the code that actually manipulates your text into an attractive document. If you're familiar with **HTML**, which is a subset of Structured Generalized Markup Language, or **SGML** then you already know about page formatting and markup languages. The *troff* macros (and consequently the *groff* macros) are simply the earliest implementation of machine interpreted markup languages.

The *ms* macros are designed to assist you in formatting letters, proposals, memos, technical papers and reports. I have used them for all of these purposes, and have also created (admittedly short) books. To meet my particular needs, I have added to the package, and also modified the supplied macros. In the sequel, I'll show some of these things[8] so that you are left with a sufficiently comfortable feeling about what you can do, and how you might go about it.

4.1 A Simple Example

Suppose that we need to write a short report. It requires a title, some author information and an abstract to precede several chapters (or sections), some of which may have subsections. We can format it as follows, and we'll see the result in **Figure 1**:

```
.TL
Our Important Report
```

[4] © 1999 - 2001 Dean Allen Provins

[5] For the terms of this licence, see the file COPYING that comes with the *groff* distribution.

[6] Some of us older folks found it rather enjoyable. It sort of set apart the men from the boys so-to-speak.

[7] There are several macro packages available with the *groff* distribution, including *man*, *ms*, *mm* and *me*: see for example, 'man 7 groff_ms' and 'man 7 groff_me' for details.

[8] I won't describe all, as some are embarrassingly poor hacks!

```
.AU
Dean Provins
.AI
Noted Author and Hacker
Student of Life and the University of Calgary
.AB
Noted author and hacker, Dean Provins reveals all in the much heralded
HOWTO for \fIgroff\fP.  In this classic document, Provins describes the
inner workings of one of the earliest machine document processing systems
available.

Readers are encouraged to try any or all the examples provided.
.AE
.NH
Machine Document Processing
.LP
In this chapter, we see that computers have finally come into their own.
No more must workers labour over Gutenberg presses, as they have for
generations.
.NH
Macros
.LP
There are many macro packages available for \fIgroff\fP and friends.  There
are even more available for its predecessor \fItroff\fP.
.ds MS \fIms\fP
.NH 2
The \*[MS] Macro Package
.LP
The \*[MS] macros provide a much needed lift for those toiling with the
basic \fItroff\fP commands.  They (the \*[MS] macros) are now used
extensively by writers the world over.
```

The result looks like the following (I trimmed the bottom of the page off for this display):

Our Important Report

Dean Provins

Noted Author and Hacker
Student of Life and the University of Calgary

ABSTRACT

Noted author and hacker, Dean Provins reveals all in the much heralded HOWTO for *groff*. In this classic document, Provins describes the inner workings of one of the earliest machine document processing systems available.

Readers are encouraged to try any or all the examples provided.

1. Machine Document Processing

In this chapter, we see that computers have finally come into their own. No more must workers labour over Gutenberg presses, as they have for generations.

2. Macros

There are many macro packages available for *groff* and friends. There are even more available for its predecessor *troff*.

2.1. The *ms* Macro Package

The *ms* macros provide a much needed lift for those toiling with the basic *troff* commands. They (the *ms* macros) are now used extensively by writers the world over.

Figure 1: *This page was created from the example text in the HOWTO. This caption, and the figure number was added with a custom ms macro.*

So what happened? The *ms* macros which are introduced by a period, followed by 2 uppercase letters in the example text were interpreted by the *troff* processor as representing *troff* commands. These then controlled the formatting of the final result. With the exception of line 26 (which is a *troff* command to be explained later) all others represented font size changes, centering, emboldening and so on.

We start with the .**TL** command. This is a macro for a **title**. It appears at the beginning of a document, centers the following text, increases the text size and sets it in bold. The author is identified after line 3 (.**AU**) which indicates that an **author name** (or several author's names) follows. This macro sets the name in italics.

The author is further identified by lines of text following the .**AI** macro in line 5. The **abstract** is identified by a .**AB** macro call, and is terminated by a .**AE** which presumably means **abstract end**.

The text of the document follows line 15, which indicates a **chapter** or main **section** via the macro call .**NH**. The section/chapter title follows immediately, and is terminated by .**LP** on line 17. Until the next macro call on line 21, all text is read and after setting in the current font and pitch, with line filling[9] turned on, The text is output to be later formatted as PostScript (or whatever format you wish, and is available).

At line 21, another chapter is identified and text is read in lines 24 and 25.

At line 26, the author thought he might take a short cut, and declared a string expression named **MS** as '\fBms\fP'. From that point on, the author could use the string expansion expression '*[MS]' to represent

[9] If insufficient text is read on an input line, more text from the next and subsequent lines is added to fill the output line. This happens until a blank line or another macro call is encountered.

'\fBms\fP'. Admittedly, this wasn't much of a saving, but it could have been if the string definition was for '\fItroff\fP' which takes more keystrokes than '*[MS]'. The **.ds** definition is an example of a *troff* command inserted into a document to facilitate text entry.

All that remains in the example is a **subsection** definition at line 27. As you can see in the resulting text, sections receive numeric identifiers followed by a period. Subsections received subordinate numeric identifiers. The *ms* macros allow for up to 4 levels of sections/subsections.

For the record, another macro command named .**SH** introduces **unnumbered sections**.

As you can see, using the *ms* macros is quite straightforward. There is no need for you to fill every line of text; in fact, *groff* is quite happy with a single word per line (although you may find it difficult to read). When entering text, use your favourite text editor (**vi** or **emacs**), and type away. If you need to add a few lines, just insert it at the appropriate point. With a text editor, it will be unlikely that you will ever encounter the formatting and overflow problems that have plagued **WYSIWYG** word processing programs on longer documents. They simply don't have the overhead; just one more reason to use a document processing system!

Just a word about **blank lines**. Blank lines terminate the filling of lines. They also indicate the ends of paragraphs. Note that the chapter/section/subsection commands generate blank lines before and after the chapter/section/subsection title.

5 Useful *ms* Macro References

This section lists the *ms* macros available for use with *groff*. They are sorted in two ways: once by function, and once alphabetically.

In the following lists, the reference to *keep* is a reference to a macro action wherein all formatted text is kept together. The text is added to the output on the current page if there is sufficient room for all of it, or it is placed at the top of the next page if there was not sufficient room.

Alphabetical Listing of *ms* Macros		
Type	**Macro**	**Description**
Page Format	**.1C**	Set single column mode (causes a page break)
Page Format	**.2C**	Set two column mode. Return to single column with .1C
Cover Page	**.AB [no]**	Begin the abstract on the cover page, ending it with .AE ("no" means do not label the abstract)
Cover Page	**.AE**	End the abstract
Cover Page	**.AI**	Print the name of the author's institution, which should appear on the next line. Multiple line entries should be separated by .br in column 1.
Text Format	**.AM**	Create better accent mark definitions
Cover Page	**.AU**	Print the author's name, which should appear on the next line. This macro normally follows .TL, and precedes .AI
Text Format	**.B [text] [trail [precede]]**	Print *text* in bold face. Multiple words should be surrounded by quotes (i.e. .B "some text"). If "text" is omitted, then the font used for rendering subsequent text is bold. Optional trailing and preceding punctuation is suffixed and prefixed to the *text* in the normal format. See also .BI, .I and .C
Text Highlight	**.B1**	Enclose the following text in a box. End with .B2
Text Highlight	**.B2**	End the boxed text which was begun with .B1
Page Format	**.BD**	Commence block display of text. All text is displayed as it is found in the source file, centered on the longest line, until terminated with .DE Compare with .DS B
Text Format	**.BI [text] [trail [precede]]**	Print *text* in bold italic. See also .B, .I and .C
Bibliography	**.BR**	In some version of the *ms* macros, commence the bibliographic format (i.e. precede a bibliographic record). This is not used in *groff*, and so may be defined by the user.
Page Format	**.BT [text]**	Set the bottom title (i.e. at the foot of the page) to "text". This is the date if not otherwise specified.
Text Highlight	**.BX text**	Surround the text (usually a single word) with a box. To surround multiple words, separate them with a digit-width space (i.e. \0).
Page Format	**.CD**	Commence centered display with each line individually centered, until terminated with a .DE Compare with .DS C
Text Format	**.C [text] [trail [precede]]**	Print *text* in constant width font. See also .B, .BI and .I
Page Format	**.DA text**	Print the current date as the center footer of each page. If *text* is specified, use that value in each page footer.
Page Format	**.DE**	End text displayed under the control of .BD, .CD, .ID, .LD or .DS

Alphabetical Listing of *ms* Macros		
Type	**Macro**	**Description**
Page Format	**.DS [type]**	Start displayed text with *keep*, of type block centered (**B**: same as .BD), centered (**C**: same as .CD), indented (**I**: same as .ID), left centered (**L**: same as .LD) or right aligned (**R**: same as .RD). Type **I** is the default. Note that .DS tries to keep all the text on a single page, while the macros similar to .DS do not. Thus if all the text cannot appear on the current page, a page break occurs if .DS is used.
Page Format	**.EF 'left'center'right'**	Place the specified text (left, center or right) in the footers for all evenly numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.
Page Format	**.EH 'left'center'right'**	Place the specified text (left, center or right) in the headers for all evenly numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.
Mathematics	**.EN**	End an equation which was begun with .EQ
Mathematics	**.EQ [x [y]]**	Commence an equation. "x" may be L, I or C, corresponding to left, indented or centered alignment. "y" is an optional equation label (use quotes to enclose multi-word labels).
Page Format	**.FE**	End a footnote.
Page Format	**.FP**	Create a numbered footnote paragraph.
Page Format	**.FS [label]**	Commence a footnote which is terminated by .FE. Optional label "label" will appear as a superscript at the beginning of the footnote. The user should cause the same "label" to appear in his text as a superscript, possibly by enclosing it as *{label*}. An alternative is to use the automatic numbering scheme afforded by the ** macro (place this macro at the appropriate place in your text) and then use .FS, .FE without any label.
Page Format	**.HD [header text]**	Place the specified *header text* below the normal page header.
Text Format	**.I [text] [trail [precede]]**	Print *text* in italic. See also .B, .BI and .C
Page Format	**.ID [indent]**	Begin an indented display without a *keep*, meaning that the display will commence immediately, and carry onto the next page if necessary. Compare with .DS I
Paragraph Format	**.IP [tag [indent]]**	Commence a new paragraph with an optional hanging word (or phrase if surrounded by quote marks. The amount of indentation may also be specified (such as .IP "Label 1" 1i which will add the text *Label 1* to the output, and indent all following lines by 1 inch).

Alphabetical Listing of *ms* Macros		
Type	**Macro**	**Description**
Index	**.IX [a [b [c [d]]]**	Place up to 4 index words or phrases (if surrounded by quote marks) in the error output file after appending the arguments to .IX with the current page number. It is up to another program or script to format these additional data for inclusion into the final document. See elsewhere in this HOWTO for a possible script.
Page Format	**.KE**	End a collection of lines of text which together will be kept together as a *keep*. See .KF and .KS for a description of the types of *keep*s available.
Page Format	**.KF**	Begin a collection of lines of text which will be kept together as a *keep*. This *keep* is known as a floating *keep* in that if there is insufficient room on the current page for the entire block, then it will be placed at the top of the next page. See also .KS
Page Format	**.KS**	Begin a collection of lines of text which will be kept together as a *keep*. This *keep* is known as a fixed keep. IF there is at least one line of room available for output text on the present page, then this collection will be added to the current page.
Page Format	**.LD**	Begin a left justified display without a *keep*. Compare with .DS L
Text Format	**.LG**	Increase the point size of the text by 2.
Paragraph Format	**.LP**	Commence a left justified paragraph.
Page Format	**.MC width**	Begin outputting multiple columns of width *width*. The width parameter should be specified in units like inches (such as .MC 2i).
Page Format	**.ND text**	Prevent the placing of the date in each page footer (see .DA). Place *text* on the cover page.
Section Header	**.NH [level [start level]]**	Place a numbered chapter or section header in the output (the header text follows on the next line or lines, until a .LP or other paragraph macro is encountered). A *level* of zero or one (or if none is specified) causes a section header of the highest level to be created. Levels 2 through 4 are subordinate (i.e. level 0 would appear as "**1. Title**", and level 2 would appears as "**1.1 Section**"). If *start level* is specified and *level* equals **S** then the current section number is set to that value (such as .NH S 3 would cause the next section level to be something like "**3. Section Title**", even though the previous section was (say) 17.
Text Format	**.NL**	Set point size to the default (10 pts, if not specified via the number register PS at the beginning of the document).
Page Format	**.OF 'left'center'right'**	Place the specified text (left, center or right) in the footers of all odd numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.

Alphabetical Listing of *ms* Macros		
Type	**Macro**	**Description**
Page Format	**.OH 'left'center'right'**	Place the specified text (left, center or right) in the headers of all odd numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.
Page Format	**.P1**	May be used to print the header on the first page, if macro .TM is defined. In *groff*, it is defined to set the first page to page 1.
Paragraph Format	**.PP**	Commence a paragraph with the first line indented by the amount given in number register PI.
Page Format	**.PT [text]**	Specify the page title, placed at the head of each page via *text*. If not specified, the page title defaults to "-%-", where % is the current page number.
Table of Contents	**.PX [no]**	Print the table of contents. If *no* is used as the argument, then no title is added. Normally, this is invoked by the .TC macro.
Paragraph Format	**.QP**	Commence a paragraph that is fully indented on both the left and right hand sides. This is often used for quotations.
Text Format	**.R**	Return to Roman font. Note that there is no argument for this macro. Compare to .B, .BI and .C, all of which accept arguments.
Page Format	**.RD**	Begin a right justified display without a *keep*. Compare with .DS R
Text Format	**.RE**	Retreat one level of relative indentation. See also .RS
Document Format	**.RP [no]**	Format the document in *released paper* format. If *no* is used as the argument, then suppress the title on the first page.
Text Format	**.RS**	Indent by one level of relative indentation. The amount of indentation may be altered by use of the *troff* command .in (default *5n*) within a .RS/.RE pair (such as .RS, .in 10n, ... text..., .RS)
Section Header	**.SH**	Create a new section using the line (or lines following, up to another paragraph command like .NH) as a bold title. Unlike .NH, this section will not be numbered.
Text Format	**.SM**	Decrease the size of following text by 2 points. Compare to .LG and .NL
Paragraph Format	**.TA**	Set the tab setting to to *5n*. This macro may be redefined (say as .de TA, .ta T 2i, .fc :, .., which would cause text separated by colons to appear to separated by 2 inch tabs).
Table of Contents	**.TC [no]**	Print a table of contents page with the first page having page number one, set in Roman numerals. Compare to .PX which does not alter the page numbers in any way. If *no* is specified, then the title (*Table of Contents*) is not printed.
Table Format	**.TE**	End a table which was commenced by the macro .TS, and which will be formatted by the *tbl* processor.
Table Format	**.TH**	End a table header which will be printed at the head of each page if the table exceeds a single page. This requires the use of option *H* on the .TS invocation.

Alphabetical Listing of *ms* Macros		
Type	**Macro**	**Description**
Cover Page	**.TL**	Set the title, which is all following text up to the next macro command (generally .AU) in bold text, two points larger than the default font size.
Document Format	**.TM**	This macro is not implemented in *groff*, but was used to set documents in the University of California thesis format. It may be defined by users for their own purpose.
Table Format	**.TS [H]**	Begin a table to be formatted by the *tbl* processor (generally invoked with the *-t* option to *groff*). If *H* is specified, then the table will have a multi-page header (see .TH).
Text Format	**.UL [text]**	Underline the text on the macro command line. Multiple words may be underlined if enclosed in double quotes. Unlike commands such as .B, following lines of text are not processed by the .UL macro.
Text Format	**.UX [text]**	Place the word `UNIX®` in the text the first time the macro is used, but only the word `UNIX` on subsequent invocations. If *text* (multiple words allows if enclosed in quotes) appears on the macro command, the text will be appended to the word `UNIX`. This is useful for adding punctuation.
Table of Contents	**.XA [no [indent]]**	Add an entry to the table of contents. If *no* appears as the first argument, then do not include the page number. The *indent* determines the relative horizontal position of the entry in the generated table.
Table of Contents	**.XE**	Terminate the addition of a single (or multiple) entry to the table of contents. Compare to .XS and .XA
Paragraph Format	**.XP**	Commence a paragraph with the first line having no indent, but with subsequent lines having an indent equal to the value of number register PI (default is *5n*). Subsequent paragraphs will all be indented.
Table of Contents	**.XS [page | no [indent]]**	Add an entry to the table of contents. if *no* appears, then the page number will not be included. If *page* (i.e. a page number) appears, then that page number will be used for the entry. The *indent* determines the relative horizontal position of the entry in the generated table.

The following is a listing sorted by macro type.

Functional Listing of *ms* Macros		
Type	**Macro**	**Description**
Bibliography	**.BR**	In some version of the *ms* macros, commence the bibliographic format (i.e. precede a bibliographic record). This is not used in *groff*, and so may be defined by the user.
Cover Page	**.AB [no]**	Begin the abstract on the cover page, ending it with .AE ("no" means do not label the abstract).
Cover Page	**.AE**	End the abstract.
Cover Page	**.AI**	Print the name of the author's institution which should appear on the next line. Multiple line entries should be separated by .br in column 1.
Cover Page	**.AU**	Print the author's name, which should appear on the next line. This macro normally follows .TL, and precedes .AI
Cover Page	**.TL**	Set the title, which is all following text up to the next macro command (generally . AU) in bold text, two points larger than the default font size.
Document Format	**.RP [no]**	Format the document in *released paper* format. If *no* is used as the argument, then suppress the title on the first page.
Document Format	**.TM**	This macro is not implemented in *groff*, but was used to set documents in the University of California thesis format. It may be defined by users for their own purpose.
Index	**.IX [a [b [c [d]]]**	Place up to 4 index words or phrases (if surrounded by quote marks) in the error output file after appending the arguments to .IX with the current page number. It is up to another program or script to format these additional data for inclusion into the final document. See elsewhere in this HOWTO for a possible script.
Mathematics	**.EN**	End an equation which was begun with .EQ
Mathematics	**.EQ [x [y]]**	Commence an equation. "x" may be L, I or C, corresponding to left, indented or centered alignment. "y" is an optional equation label (use quotes to enclose multi-word labels).
Page Format	**.1C**	Set single column mode (causes a page break).
Page Format	**.2C**	Set two column mode. Return to single column with .1C
Page Format	**.BD**	Commence block display of text. All text is displayed as it is found in the source file, centered on the longest line, without a *keep*, until terminated with .DE Compare with .DS B
Page Format	**.BT [text]**	Set the bottom title (i.e. at the foot of the page) to "text". This is the date if not otherwise specified.
Page Format	**.CD**	Commence centered display with each line individually centered, until terminated with a .DE Compare with .DS C

Functional Listing of *ms* Macros		
Type	**Macro**	**Description**
Page Format	**.DA text**	Print the current date as the center footer of each page. If *text* is specified, use that value in each page footer.
Page Format	**.DE**	End text displayed under the control of .BD, .CD, .ID, .LD or .DS
Page Format	**.DS [type]**	Start displayed text with *keep*, of type block centered (**B**: same as .BD), centered (**C**: same as .CD), indented (**I**: same as .ID), left centered (**L**: same as .LD) or right aligned (**R**: same as .RD). Type **I** is the default. Note that .DS tries to keep all the text on a single page, while the macros similar to .DS do not. Thus if all the text cannot appear on the current page, a page break occurs if .DS is used.
Page Format	**.EF 'left'center'right'**	Place the specified text (left, center or right) in the footers for all evenly numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.
Page Format	**.EH 'left'center'right'**	Place the specified text (left, center or right) in the headers for all evenly numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.
Page Format	**.FE**	End a footnote.
Page Format	**.FP**	Create a numbered footnote paragraph.
Page Format	**.FS [label]**	Commence a footnote which is terminated by .FE. Optional label "label" will appear as a superscript at the beginning of the footnote. The user should cause the same "label" to appear in his text as a superscript, possibly by enclosing it as *{label*}. An alternative is to use the automatic numbering scheme afforded by the ** macro (place this macro at the appropriate place in your text) and then use .FS, .FE without any label.
Page Format	**.HD [header text]**	Place the specified *header text* below the normal page header.
Page Format	**.ID [indent]**	Begin an indented display without a *keep*, meaning that the display will commence immediately, and carry onto the next page if necessary. Compare with .DS I
Page Format	**.KE**	End a collection of lines of text which together will be kept together as a *keep*. See .KF and .KS for a description of the types of *keep*s available.
Page Format	**.KF**	Begin a collection of lines of text which will be kept together as a *keep*. This *keep* is known as a floating *keep* in that if there is insufficient room on the current page for the entire block, then it will be placed at the top of the next page. See also .KS
Page Format	**.KS**	Begin a collection of lines of text which will be kept together as a *keep*. This *keep* is known as a fixed keep. IF there is at least one line of room available for output text on the present page, then this collection will be added to the current page.

Functional Listing of *ms* Macros		
Type	**Macro**	**Description**
Page Format	**.LD**	Begin a left justified display without a *keep*. Compare with .DS L
Page Format	**.MC width**	Begin outputting multiple columns of width *width*. The width parameter should be specified in units like inches (such as .MC 2i).
Page Format	**.ND text**	Prevent the placing of the date in each page footer (see .DA). Place *text* on the cover page.
Page Format	**.OF 'left'center'right'**	Place the specified text (left, center or right) in the footers of all odd numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.
Page Format	**.OH 'left'center'right'**	Place the specified text (left, center or right) in the headers of all odd numbered pages. One or more of the text strings may be omitted, but all single quote marks must be given.
Page Format	**.P1**	May be used to print the header on the first page, if macro .TM is defined. In *groff*, it is defined to set the first page to page 1.
Page Format	**.PT [text]**	Specify the page title, placed at the head of each page via *text*. If not specified, the page title defaults to "-%-", where % is the current page number.
Page Format	**.RD**	Begin a right justified display without a *keep*. Compare with .DS R
Paragraph Format	**.IP [tag [indent]]**	Commence a new paragraph with an optional hanging word (or phrase if surrounded by quote marks. The amount of indentation may also be specified (such as .IP "Label 1" 1i which will add the text *Label 1* to the output, and indent all following lines by 1 inch).
Paragraph Format	**.LP**	Commence a left justified paragraph.
Paragraph Format	**.PP**	Commence a paragraph with the first line indented by the amount given in number register PI.
Paragraph Format	**.QP**	Commence a paragraph that is fully indented on both the left and right hand sides. This is often used for quotations.
Paragraph Format	**.TA**	Set the tab setting to to *5n*. This macro may be redefined (say as .de TA, .ta T 2i, .fc :, .., which would cause text separated by colons to appear to separated by 2 inch tabs).
Paragraph Format	**.XP**	Commence a paragraph with the first line having no indent, but with subsequent lines having an indent equal to the value of number register PI (default is *5n*). Subsequent paragraphs will all be indented.

Functional Listing of *ms* Macros		
Type	**Macro**	**Description**
Section Header	**.NH [level [start level]]**	Place a numbered chapter or section header in the output (the header text follows on the next line or lines, until a .LP or other paragraph macro is encountered). A *level* of zero or one (or if none is specified) causes a section header of the highest level to be created. Levels 2 through 4 are subordinate (i.e. level 0 would appear as "**1. Title**", and level 2 would appears as "**1.1 Section**"). If *start level* is specified and *level* equals **S** then the current section number is set to that value (such as .NH S 3 would cause the next section level to be something like "**3. Section Title**", even though the previous section was (say) 17.
Section Header	**.SH**	Create a new section using the line (or lines following, up to another paragraph command like .NH) as a bold title. Unlike .NH, this section will not be numbered.
Table Format	**.TE**	End a table which was commenced by the macro .TS, and which will be formatted by the *tbl* processor.
Table Format	**.TH**	End a table header which will be printed at the head of each page if the table exceeds a single page. This requires the use of option *H* on the .TS invocation.
Table Format	**.TS [H]**	Begin a table to be formatted by the *tbl* processor (generally invoked with the *-t* option to *groff*). If *H* is specified, then the table will have a multi-page header (see .TH).
Table of Contents	**.PX [no]**	Print the table of contents. If *no* is used as the argument, then no title is added. Normally, this is invoked by the .TC macro.
Table of Contents	**.TC [no]**	Print a table of contents page with the first page having page number one, set in Roman numerals. Compare to .PX which does not alter the page numbers in any way. If *no* is specified, then the title (*Table of Contents*) is not printed.
Table of Contents	**.XA [no [indent]]**	Add an entry to the table of contents. If *no* appears as the first argument, then do not include the page number. The *indent* determines the relative horizontal position of the entry in the generated table.
Table of Contents	**.XE**	Terminate the addition of a single (or multiple) entry to the table of contents. Compare to .XS and .XA
Table of Contents	**.XS [page | no [indent]]**	Add an entry to the table of contents. if *no* appears, then the page number will not be included. If *page* (i.e. a page number) appears, then that page number will be used for the entry. The *indent* determines the relative horizontal position of the entry in the generated table.
Text Format	**.AM**	Create better accent mark definitions

Functional Listing of *ms* Macros		
Type	**Macro**	**Description**
Text Format	**.B [text] [trail [precede]]**	Print *text* in bold face. Multiple words should be surrounded by quotes (i.e. .B "some text"). If "text" is omitted, then the font used for rendering subsequent text is bold. Optional trailing and preceding punctuation is suffixed and prefixed to the *text* in the normal format. See also .BI, .I and .C
Text Format	**.BI [text] [trail [precede]]**	Print *text* in bold italic. See also .B, .I and .C
Text Format	**.C [text] [trail [precede]]**	Print *text* in constant width font. See also .B, .BI and .I
Text Format	**.I [text] [trail [precede]]**	Print *text* in italic. See also .B, .BI and .C
Text Format	**.LG**	Increase the point size of the text by 2.
Text Format	**.NL**	Set point size to the default (10 pts, if not specified via the number register PS at the beginning of the document).
Text Format	**.R**	Return to Roman font. Note that there is no argument for this macro. Compare to .B, .BI and .C, all of which accept arguments.
Text Format	**.RE**	Retreat one level of relative indentation. See also .RS
Text Format	**.RS**	Indent by one level of relative indentation. The amount of indentation may be altered by use of the *troff* command .in (default *5n*) within a .RS/.RE pair (such as .RS, .in 10n, ... text..., .RS)
Text Format	**.SM**	Decrease the size of following text by 2 points. Compare to .LG and .NL
Text Format	**.UL [text]**	Underline the text on the macro command line. Multiple words may be underlined if enclosed in double quotes. Unlike commands such as .B, following lines of text are not processed by the .UL macro.
Text Format	**.UX [text]**	Place the word `UNIX®` in the text the first time the macro is used, but only the word `UNIX` on subsequent invocations. If *text* (multiple words allows if enclosed in quotes) appears on the macro command, the text will be appended to the word `UNIX`. This is useful for adding punctuation.
Text Highlight	**.B1**	Enclose the following text in a box. End with .B2
Text Highlight	**.B2**	End the boxed text which was begun with .B1
Text Highlight	**.BX text**	Surround the text (usually a single word) with a box. To surround multiple words, separate them with a digit-width space (i.e. \0).

5.1 Registers used by the *ms* Macros

5.1.1 Number Registers

So those are the *ms* macros. To use them effectively, you should also know about the registers that are set when you use the *ms* macros. These are in the following tables[10]:

ms **Macro Number Registers Sorted Alphabetically**			
Name	**What it Controls**	**When it Takes Effect**	**Default Value**
DD	distance to display	displays	0.5v
FF	footnote format	next .FS	0, 1, 2 or 3
FI	footnote indent	next .FS	2n
FL	footnote length	next .FS	5.5i
FM	footer margin	next page	1i
HM	header margin	next page	1i
LL	length of a line	paragraph	6i
LT	title length	next page	6i
PD	inter paragraph distance	paragraph	0.3v
PI	paragraph indent	paragraph	5n
PO	left margin	next page	1i
PS	text size (in points)	paragraph	10
QI	quoted paragraph indent	next .QP	5n
VS	spacing between lines	paragraph	12

ms **Macro Number Registers Sorted By Function**			
Name	**What it Controls**	**When it Takes Effect**	**Default Value**
DD	distance to display	displays	0.5v
FM	footer margin	next page	1i
HM	header margin	next page	1i
LL	length of a line	paragraph	6i
PO	left margin	next page	1i
FF	footnote format	next .FS	0, 1, 2 or 3
FI	footnote indent	next .FS	2n
FL	footnote length	next .FS	5.5i
PD	inter paragraph distance	paragraph	0.3v
PI	paragraph indent	paragraph	5n
QI	quoted paragraph indent	next .QP	5n
VS	spacing between lines	paragraph	12
PS	text size (in points)	paragraph	10
LT	title length	next page	6i

Note that the **footnote format** has more than one possible value. If set to 1, footnote superscripting is suppressed. If set to 2, indentation of the first line is also suppressed. If set to 3, the footnote resembles an .IP-like footnote paragraph.

Why might you require the number registers? If you were a writer, or perhaps a student that wanted to format some document in a particular manner, say for their thesis, then these registers would allow you control the look of the document. I believe that **WYSIWYG** call the look of a document, the **document style**. For

[10] In these tables, I show only the default values for *groff* (i.e. the equivalent to *troff*, not *nroff*. See the man page for details.

example, you might want unequal left and right margins. Perhaps the left should be 2 inches, and the right just 1. This can be effected by altering the **left margin** and **length of line** registers at the beginning of the document. You would do this as follows:

```
.nr PO 2i
.nr LL 5.5i
```

If you also wanted to set 12 point type, with a corresponding increase in line spacing, you would use:

```
.nr PS 12
.nr VS 10*1200/1000
```

Why, you ask, did I use such a complicated formula for setting the line spacing? Well, *groff* doesn't support floating point arithmetic, and to get a line spacing which is 20% larger than the point size, I simply multiplied by 1.2 times the *sizescale* parameter[11] and then rescaled down the result to points (by dividing by the *sizescale* parameter).

5.1.2 String Registers

There are also **registers**string available for use in your documents. These are tabulated as follows:

ms **Macro String Registers sorted Alphabetically**	
Register Name	**Function**
*Q	leading quote mark
*U	trailing quote mark
*_	dash
*(MO	current month of the year
*(DA	current day of the month
**	automatically numbered footnote
*'	acute accent (place before letter)
*`	grave accent (place before letter)
*^	circumflex accent (place before letter)
*,	cedilla accent (place before letter)
*:	umlaut accent (place before letter)
*~	tilde accent (place before letter)

Note that if you employ .AM to obtain enhanced mark definitions, the strings should appear after the letter, not before.

To use these registers, say to place a circumflex over the letters *e* in the word *example*, you code as follows:

```
\*^exampl\*^e
```

This results in êxamplê. if the .AM macro has not been invoked. If it has, then the coding is:

```
e\*^xample\*^
```

Let me add an aside on string registers.

You can create and use your own string registers with the *troff* command **.ds**. Suppose that you are writing this HOWTO, and find that you have to type \fIgroff\fP many times (as I have done in this document), or perhaps you want to use the phrase *groff is* **great** many times. You could code as follows, where you want the two letter code **GG** to represent the longer string (not a great example, but you get the idea):

```
.ds GG \fIgroff is\fP \fBgreat\fP
```

You can now use *[GG], or *(GG (either will do, see the *gtroff* man page) to produce: *groff is* **great**!

[11] See the man page for *gtroff* for a description of fractional point sizes.

6 Tables

Periodically, you may find that you need to set out some text in a tabular form. Perhaps it's a list of names and numbers, or perhaps it is much more complicated. The table capabilities of the *groff* suite are handled by a processor known as **tbl**.

To use **tbl**, one needs to describe the general format (it needn't be fixed for the entire table), and then enter the text in a manner that indicates which text belongs in which column. While I describe text as appearing in tables, I know that equations can also be entered. Whether figures can be or not, is not something I have ever experimented with.

The only limitation that I have ever encountered with tables is the use to footnotes. As I recall, automatically numbered footnotes are incorrectly numbered; but we'll see if that still holds in the following example.

Suppose that we have some names and telephone numbers of key individuals, and that we would like to organize them in a table to be contained in a document. The following might do the job for an individual who uses *groff* to keep his diary:[12]

```
.SH
Another Day in the Life of Dean P.
.LP
Lately, my life has become more complicated.  Now the boss wants me to
remember the names and telephone numbers of key suppliers, and selected
contacts.  As if I don't already have enough to do.  Well my memory used
to be up to the task, but in these declining years, if I want to hang on
to this job, I better work smarter.  I think I'll use that new-fangled
groff program.  I hear it's great!
.TS
center box tab (:);
cb cb cb cb
r | n | l | l.
Telephone Number:Serial Number:Contact:Supplier
_
123-3456:123.:Jim:ABS Supplies
123-5437:.456::Genco Paints
557-234-3333:123.456:Suzy:HAL Computer Systems
.TE
Well that should do it.  As long as I have this diary, I'll never forget
again.
.SH
The Next Day
.LP
Wow!  I sure impressed the boss today.  I got ABS Supplies contact name
with just one small glance at my diary.  Now I'm in his good books!
```

And what is the result of this? Here is **Figure 2**.

[12] Let us assume that we don't need numbered chapters.

Another Day in the Life of Dean P.

Lately, my life has become more complicated. Now the boss wants me to remember the names and telephone numbers of key suppliers, and selected contacts. As if I don't already have enough to do. Well my memory used to be up to the task, but in these declining years, if I want to hang on to this job, I better work smarter. I think I'll use that new-fangled *groff* program. I hear it's great!

Telephone Number	Serial Number	Contact	Supplier
123-3456	123.	Jim	ABS Supplies
123-5437	.456		Genco Paints
557-234-3333	123.456	Suzy	HAL Computer Systems

Well that should do it. As long as I have this diary, I'll never forget again.

The Next Day

Wow! I sure impressed the boss today. I got ABS Supplies contact name with just one small glance at my diary. Now I'm in his good books!

Figure 2: *This is the page that corresponds to the un-numbered chapter titled, "Another Day in the Life of Dean P."*

In this example, we used the section command .**SH** to provide unnumbered section titles (lines 1-3). Line 10 saw the beginning of our table, which went to line 19. As you can see, tables begin with a .**TS** command, and end with a .**TE** command.

The first line after .**TS** (up to the semi-colon) describes some general parameters. I have indicated that the table should be centered, enclosed in a box, and that the text information will be divided into columns by a colon (the **tab** command). You can use any character here, including a real TAB. Note that this line must be terminated by the semicolon.

The next line applies to the first line of the table data. In this case it says that there will be four columns, and that the text for each column (if given) is to be centered and set in bold type. The final line (up to the period on line 13) describes the subsequent lines of text which make up the table. Here I indicated that the first column was to be right justified (**r**) and followed by a vertical line (**|**). The next column will be treated as numeric data and the decimal points will be aligned. The last two columns will be left justified. Vertical lines will separate all columns. The period at the end of this line indicates that the formatting information is complete.

The next line is presumed to be the column headers (corresponding to the first line of format description). Then comes a horizontal line, followed by the remaining lines of text. Notice the colons separating each column, and notice that line 17 lacks one column of data. The **tbl** program leaves space for missing data, if necessary. As I said earlier, the **tab** character (a colon here) can be any character not in the text.

This is a very simple table example, but is useful to illustrate how easy it is to generate one. A very slightly more complicated table might look like,

A Table Containing Equations	
Equation 1	$a = \sqrt{b^2 + \frac{c}{d^3}}$
Equation 2	$a_k = \sum_{i=1}^{N} C_i \int_0^{\pi} x_k^2 dx$
Don't you wish there were **no more**	Equations?

It was generated by means of

```
 1   .TS
 2   center doublebox delim (@#) tab (   );
 3   cb s
 4   l l.
 5   A Table Containing Equations
 6   _
 7   Equation 1     @a = sqrt {b sup 2 + c over {d sup 3}}#

 8   Equation 2     @a sub k = sum from i=1 to N C sub i  int from 0 to pi x sub k sup 2 dx#
 9   .T&
10   r l.
11   T{
12   Don't you wish there were
13   \fBno more\fP
14   T}   T{

15   .br
16   Equations
17   T}
18   .TE
```

In this example, I have illustrated inline equation definitions. As you can see, they are textual, which makes them reasonably easy to read (and key in).

I also asked for a **doublebox** centered, with equation delimiters of @ and #,[13] and a TAB character as the separator between the columns.[14]

Between lines 7 and 8 there is a blank line (unnumbered), and at line 9, I have given a **tbl** command (**.T&**) which says that I wish to restate the format of the columns (which I define on the next line, ending it in a period).

Just for fun, and to illustrate how one can enter long strings of text (although mine is very short), I have illustrated the use of the text block entry commands **T{** and **T}**. These must appear as follows:

- Enter the **T{** as the first two characters immediately following the **tab**, or the first two characters if the text block belongs in the first column.
- Place the **T}** on the line immediately following the last line of text in the block. If another column follows, enter the **tab** and continue with that column's text.

The **tbl** program will wrap blocks of text into a single column for you. As you can see, **tbl** wraps the text where it likes (not necessarily where you would like); nevertheless, text wrap, especially in the last column can be very useful.

7 Equations

groff has excellent support for mathematics. Virtually all the math symbols that you might like to use are available. You can code sums, integrals, matrices, simple or complex algebraic statements in near-English. For example, a matrix is identified by the same word: **matrix**. Columns of the matrix are identified as left,

---

[13] In my documents, for historical reasons, I set these as the equation delimiters. More typically, you would use two @ symbols, or any character that you would never use in your equations. You know that the @ character is also used in internet mail addresses. Unfortunately, the **eqn** processor doesn't know the difference between the start of an inline equation and an address. If you need to include addresses in your document, precede the @ by a backslash (e.g. dprovins\@ucalgary.ca). Perhaps a better choice would be a dollar sign (i.e. $). Then when you wanted to use that symbol, you could either hide it with a backslash (i.e. \$), or use a Postscript dollar symbol \(Do which looks just the same (i.e. $).

[14] A real TAB entered at this location on the page causes the parentheses to appear separated by this amount. At another location, the amount might be different.

center and right-justified by the terms **lcol**, **ccol** and **rcol**.

Limits of sums and integrals are easy to define with terms like **from** and **to**.

We've already seen how equations can be included in tables, lets look at some more. The only rule that you need to remember (and which I frequently fail to observe because of my poor (make that sloppy) typing ability), is the need to separate terms by at least one space. I suppose one other is that curly brackets (i.e. "{" and "}") are used to group expressions which are to be treated as a unit, as in defining the numerator of a complicated fraction.

Here's an expression for a Legendre function:

$$P_{n,m}(\cos\theta) = \frac{1}{2^n n!}(t - t^2)^{m/2} \frac{d^{n+m}}{dt^{n+m}}(t^2 - 1)^n$$

I coded it as

```
.EQ
P sub n,m ( cos theta ) =
1 over {2 sup n n!} (t - t sup 2 ) sup m/2 {d sup n+m} over
{dt sup n+m} (t sup 2 - 1 ) sup n
.EN
```

Here is another expression, a discrete Fourier transform:

$$F(\frac{k}{N}\Delta t) = \sum_{j=0}^{N-1} f(j\Delta t)e^{-\frac{i2\pi jk}{N}} \quad n = 0, 1, \cdots, N-1$$

It was coded as,

```
.EQ
F ( k over N DELTA t ) = sum from j=0 to N-1 f( j DELTA t ) e sup {- {i 2 pi
jk} ov N}~~~n = 0, 1, ... , N-1
.EN
```

As you can see, an English-like syntax, using tokens that (generally) represent the mathematical symbols allow you to describe fairly complex expressions. Each token is separated from all others by white space (i.e. space bar entries, tab key entries or newline entries). Four other special characters are used as follows:

eqn Token Separators	
Separator	**Use**
Braces: { }	Braces, or curly brackets, are used to group terms so that they may be acted on as a unit. An example would be to take the square root of $a + b$. Without braces (i.e. sqrt a + b), you get $\sqrt{a} + b$, but with braces (i.e. sqrt {a + b}), you get $\sqrt{a + b}$. Note that the braces do not require white space around them.
double quotes: ""	Double quotes are used to separate text from an equation when used between a .EQ and .EN pair.
tilde: ~	A tilde represents a full space in the output. That is, it ensures that the math symbols are separated when printed (normally **eqn** removes unnecessary white space).
Circumflex: ^	A circumflex provides half the space of a tilde.

7.1 Key Words for Equation Setup

Other key words are described in the following table:

Key Words for Describing Equations	
Term	**Description**
sub, sup	These terms indicate subscripts and superscripts. Order is important as A sub i sup j which yields A_i^j, is different from A sup j sub i, which is A^{j_i}. You might also want A^{j_i} which results from A sup {j sub i}.
over	This is used to create fractions such as a over b, which creates $\frac{a}{b}$, and {a sin theta cos alpha sup 2} over {sqrt {b sup 2 - 4 a c}} which creates $\frac{a \sin\theta \cos\alpha^2}{\sqrt{b^2 - 4ac}}$.
sum, int, from, to, inf	These terms are used to create sums and integrals with optional limits. One can have sum from {i = 0} to inf a sup 2 sin theta int from 0 to pi cos alpha d alpha, which yields $\sum_{i=0}^{\infty} a^2 \sin\theta \int_0^{\pi} \cos\alpha d\alpha$
left, right	These two terms allow one to bracket expressions with round or square parenthesis, such as left [x sup 2 + left (y sup 2 over alpha right) sin theta right], which yields $\left[x^2 + \left(\frac{y^2}{\alpha} \right) \sin\theta \right]$. Note that the **right** clause is optional, but the **left** must appear if a **right** is required. Use a dummy **left** such as left "", which means no symbol. One particular variation are the phrases *left floor*, *left ceiling*, *right floor* and *right ceiling*. When used between .EQ and .EN, these produce hooked bracket pairs.
lpile, cpile rpile, pile	Piles produce vertically aligned equations, such as lpile {a = b + c above sqrt {c + d}}, which yields $\begin{array}{l} a = b + c \\ \sqrt{c + d} \end{array}$. **l**, **c** and **r** represent left, center and right alignment of the expressions. Notice that braces are required to identify all the terms in the pile.
matrix, lcol, ccol, rcol	These are used to construct matrices, such as left [matrix {lcol {a above b+c} ccol {d+e above f} rcol {"eqn 1" above "eqn 2"}} right] which yields $\left[\begin{array}{lcr} a & d+e & \mathit{eqn\ 1} \\ b+c & f & \mathit{eqn\ 2} \end{array} \right]$. Notice that braces are required to identify all the terms in both the matrix and each of the columns.

Key Words for Describing Equations	
Term	**Description**
dot, dotdot, hat, tilde, bar, vec, dyad, under	These terms, which follow the expression to which they apply provide diacritical marks. For example, a dot ˜ b under ˜ c vec d dyad looks like $\dot{a}$ $\underline{b}$ $\vec{c}$ $\overleftrightarrow{d}$.
mark, lineup	If **mark** is used in an equation, then the position at which the next mathematical symbol in the expression is remembered. Subsequent equations which include **lineup** will cause that equation to be aligned with the previous equation. If not used, and .EQ and .EN have no alignment specification on the macro invocation, then each equation will be centered.
define	Keywords may be aliased by means of a **define** statement within a .EQ and .EN pair. For example, one could tire of writing "cos (theta)" many times, so one could say define ct "{cos (theta)}". Subsequent uses of the term **ct** would be replace by {cos (theta)}. Notice that I included braces (as well as surrounding the aliased expression with quotes. The braces allow be to say ct over a+b to get $\frac{\cos(\mathit{theta})}{a+b}$, instead of $\cos(\theta \frac{)}{a+b}$, which is not likely to be what I want. The quote marks simply identify the complete expression being aliased.
Greek letters etc.	Lower case Greek letters like alpha, beta and so on produce $\alpha\beta$, and upper case PHI, GAMMA yield $\Phi\Gamma$ as you might expect.

7.2 Words with Special Meaning in Equations

Certain words are interpreted by **eqn** to mean certain mathematical symbols. While all available symbols are described in the next section, there is a short list of commonly used symbols for which easily remembered names have been ascribed. The following table lists them.

Special Words in Equations	
Input	**Output**
>=	$\geq$
<=	$\leq$
==	$\equiv$
!=	$\neq$
+-	$\pm$
->	$\rightarrow$
<-	$\leftarrow$
<<	$\ll$
>>	$\gg$
inf	∞
partial	∂
prime	$\prime$
approx	$\approx$

Special Words in Equations	
Input	**Output**
nothing	
cdot	$\cdot$
times	$\times$
del	∇
grad	∇
...	$\cdots$
,...,	$,\ldots,$
prod	$\prod$
union	$\bigcup$
inter	$\bigcap$

Selected mathematical words are printed in Roman font when used. These are:

sin	sinh	arc	log	Re	and
cos	cosh	max	ln	Im	if
tan	tanh	min	exp	det	for

7.3 Other Symbols for Use in Equations

There are many mathematical symbols available for your use, far too many for me to list here. If you do a *man groff_char*, you will see what characters and math symbols you can use.

This *man* page lists input codes from 33 to 255 (with a few omitted). To employ any of these (other than those that you can type directly on the keyboard: numbers 33 to 96, and 123 to 126 inclusive), simply code them as \N'num', where *num* is the numeric input code. For example, to print **ydieresis**, number 255, code it as \N'255' to yield ÿ.

Symbols that lack numeric input codes (all after number 255) have textual codes like \(-D, whose PostScript name is **Eth** (the symbol immediately following code number 255). Using the given coding, an **Eth** would look like Ð.

7.4 Equations and Comments

You should know that the equation processor doesn't seem to recognize comments. That is, if you place the sequence .\" before some text that you wish to comment out, then *eqn* will happily interpret it if inside an ".EQ - .EN" pair. Furthermore, the pair ".ig - .." doesn't work within an ".EQ - .EN" pair. Either will prevent the equation processor from interpreting inline equations though.

8 Figures

Figures may be used in *groff* documents, with limited positioning capability. The default position of a figure is in the center of the page (assuming a single column of text). One may also place a figure as a left justified or right justified object.

Unless you take care to use certain *troff* macros (which I will explain in a moment), no text can appear, nor can multiple images appear at the same vertical place as a figure. This is not overly restrictive as centered figures stand out well on a printed page. Both the figures of example printed pages seen earlier are typical.

groff accepts **encapsulated PostScript** as the only figure format when using the *ms* macros. When using the **HTML** macro packages, both **GIF** and **PNG** format images are acceptable.

Encapsulated PostScript is simply a PostScript file that contains a **BoundingBox** comment near the beginning of the file. This *box* describes the **x** and **y** limits of all ink which is to be placed on the page.

To introduce a figure to a document, the **PSPIC**. macro is employed. It takes several arguments, as described in the *man* page for **grops**. To summarize, the macro command has the format:

```
.PSPIC [-L | -R | -I n] file [ width [ height ]]
```

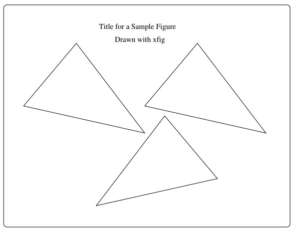
The graphic is indicated by *file* (e.g. my_pict.eps), and the width and height that you wish the illustration to have is indicated by *width* and *height*, which may have scale indicators attached. The default scale is inches (**i**). Other units include **points** (**p**) and **centimeters** (**cm**). For the complete list, please see the *troff* documentation, referenced earlier.

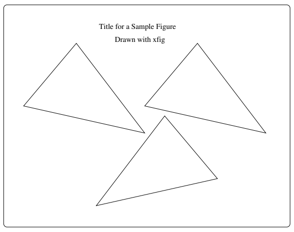
The **PSPIC** macro scales the width and height uniformly so that it is no more than *width* wide and *height* high. If either **-L** or **-R** are specified, the graphic is positioned as a left-aligned or right-aligned illustration. The **-I** option allows the graphic to be indented by *n* units (of the indicated type).

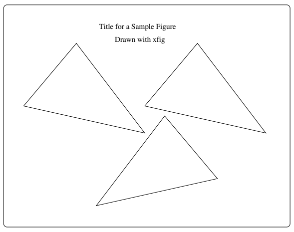
If you wish several graphics to be placed at the same vertical position, say one left-aligned, and the other right-aligned, then you must use certain *troff* commands. Basically, you must **mark** your current position on the page with the **mk**. command, and after drawing the first graphic, **return** to that position with either the **rt**. or **sp**. commands. As an example, let us draw the same (rather simple figure) three ways. We'll position it on both edges of the page, and will place it between the other two. The macro call to display the graphic *sample.eps* is:

```
.br                          \" flush all text out first
.nr p                        \" define number register 'p'
.mk p                        \" note the present vertical place on the page
.PSPIC -L sample.eps 1.5     \" left justified image, 2 inches maximum dimension
.sp |0u                    \" return to that place
.mk                          \" demonstrate the other mark method
.PSPIC    sample.eps 1.5     \" centered figure
.rt                          \" and how we can return to the other mark
.PSPIC -R sample.eps 1.5     \" right aligned figure
```

The result is







A limitation of illustration inclusion is that there is no caption capability included with the *ms* macro package. I handled this initially by using quoted paragraphs on centered illustrations. For example, using the same figure as before, I would code

```
.br
.PSPIC    sample.eps 2
.QP
Figure 5.3: This is the caption for my illustration.  Notice that I
used a program called xfig to create it.
.LP
```

This results in the following:

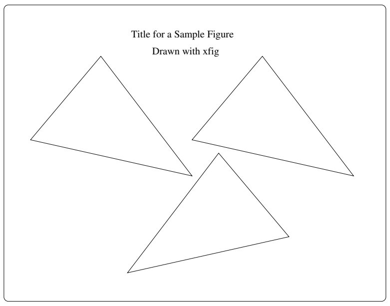


Figure 5.3: This is the caption for my illustration. Notice that I used a program called **xfig** to create it.

While this was satisfactory initially, I eventually developed some macro add-ons for *ms* to handle captions for figures and tables, and to allow me to reference these elsewhere in the text. I'll describe these later.

I should add that to surround a figure with a box, such as we saw in the first two illustrations (see for example, **Figure 1**). you simply surround the figure with the macro pair **B1**. and **B2**. Using the same figure as last time, the code would look like,

```
.br
.B1
.PSPIC    sample.eps 2
.QP
Figure 5.3: This is the caption for my illustration.  Notice that I
used a program called xfig to create it.
.LP
.B2
```

This is displayed as,

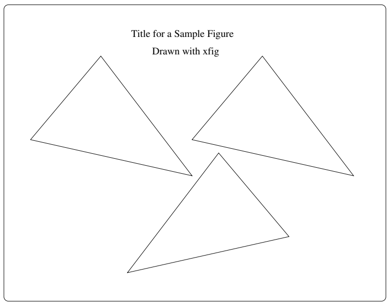


Figure 5.3: This is the caption for my illustration. Notice that I used a program called **xfig** to create it.

Lastly, you will want to be sure that your figure and caption remain together. This is called **keep**ing the text together. The macro pair to use is either, **KS**. and ,**KE**. or **KF**. and **KE**. The first pair will determine the size of the figure and caption, and place it on the same page if sufficient room remains, else it will move it to the next page, and leave a blank space on the current page. The second pair doe the same, except that it allows text to be placed on the current page, if there is room for it.

The coding looks like the following. Notice that I dropped the .**br** command as I knew that .**KS** would issue a break.

```
.KS              \" Start the keep.  You can use .KF here if desired.
.B1
```

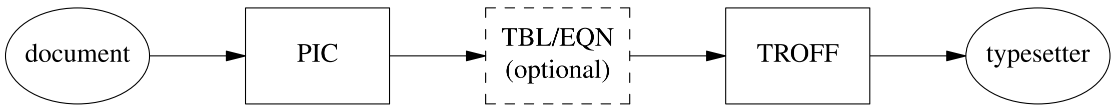
```
.PSPIC    sample.eps 2
.QP
Figure 5.3: This is the caption for my illustration.  Notice that I
used a program called xfig to create it.
.LP
.B2
.KE               \" This ends the keep
```

A last comment on figures. I mentioned that **encapsulated PostScript** was the format to use. This is not entirely true, and *groff* and its predecessor *troff* also accept a format called **PIC**. **PIC** is a graphics language which was developed and in use at Bell Labs at least early as 1984. It is a descriptive language, and uses English words to describe a picture.

An example, taken from [Kernighan 1991] is the following:

```
.PS
ellipse "document"
arrow
box "PIC"
arrow
box "TBL/EQN" "(optional)" dashed
arrow
box "TROFF"
arrow
ellipse "typesetter"
.PE
```

This somewhat innocuous text produces the following image,



The previous reference gives a full description of **PIC**'s capabilities. Those not interested in learning the **PIC** language, but still wishing to use it in their documents are referred to **xfig** which will produce **PIC** on request.

9 References

If you are writing articles that will be published, you may want to include references. Certainly, for any academic journal, this will be a requirement. Fortunately *groff* has a complete bibliographic package, called **refer**.

refer is a program that copies a *groff* document from standard input to standard output, while watching for lines between **.[** and **.]**. When these are encountered, *refer* replaces the citation found between them with an appropriate text sequence. This sequence is usually just the first (if there are several) author and a date, contained between a matched pair of parenthesis. As well, *refer* adds a complete citation at an appropriate place in the document (often at the end).

refer refers to a text-based bibliographic database in order to perform its actions. To recognize a reference, you must place key words that uniquely identify the reference between the **.[** and **.]** symbols.

For example, the sequence,

```
.[
[ ] Kernighan 1991
```

```
.]
```

generates the reference [Kernighan 1991] in the text (using my particular *refer* specifications).

The bibliographic database is composed of fields which define different parts of a reference. Each begins with a letter which defines the kind of information contained, and ends with a newline. The letter is preceded by an '%' symbol. Each complete entry is separated from all other entries in the database by one or more blank lines. Typical fields are[15]

Bibliographic Database Identifiers	
Letter Code	**Description**
A	An author's name in the order first name followed by last name. If an author has several names, or initials, as in Joe B. Smith, or J. B. Smith, then all of the first names or initials should be linked together by means of the *troff* code \0. Thus you would code Joe B. Smith as Joe\\0B. Smith, and this would be placed exactly one space after %A.
B	The title of a book that contains the article to which you are referring. See also **E**.
D	The date of publication of the book or article.
E	For an article in a book, the name of the book's editor. See also **B**.
I	The name of the publisher.
J	The name of the journal containing the article.
N	The journal issue.
P	The page numbers of an article.
T	The title of the article or the book.
V	The volume number of the journal.

A few sample entries in a bibliographic database follow.

```
%A Brian\0W. Kernighan
%C Murray Hill, New Jersey 07974, USA
%D May, 1991
%I AT&T Bell Laboratories
%R Computing Science Technical Report No. 116
%T PIC - A Graphics Language for Typesetting User Manual

%A James\0W. Cooley
%A John\0W. Tukey
%D 1965
%J Mathematics of Computation
%N 90
%P 297-301
%T An Algorithm for the Machine Calculation of Complex Fourier Series
%V 19

%A Weikko A. Heiskanen
%A Helmut Moritz
%C Graz, Austria
%D 1966
%I Institute of Physical Geodesy, Technical University
%K geomatics geodesy
%L H & M, 1993
%T Physical Geodesy
```

[15] For the complete list, do 'man refer'.

The first two entries are for a report, and for a journal article. The third is a book.

10 Customizing and Creating *ms* Macros

When I started to use *groff* several years ago (spring of 1997 to be exact), I had a copy of [Dougherty 1987] to work with. This is not a book that I recommend, not because it is incomplete as it covers everything, but because I could not easily locate the information I wanted to accomplish some specific formatting requirement. The book wasn't written to be a reference (at least in my opinion)[16].

I was influenced by some comments about the *ms* package though. I was left with the impression that if I wanted to modify any macro set, then I should consider that package specifically, it being the easiest to play with. So I jumped in and made several modifications to *ms*, and added several other macro instructions.

For example, I added a macro called **AN** which altered the way that figures, tables and equations were numbered. Of course, these weren't automatically numbered, so as you'll see shortly, I altered the **NH** macro to accommodate my numbering requirements (as well as how chapters were to be positioned on pages).

Another macro that I added was one to place not just bullets (i.e. • symbols) before indented paragraphs, but optionally numbers of letters. This one I called **BU** and I use it frequently.

For something as simple as double spacing, I created **DB**, and for a draft version of a document, I created **DM**, which calls **DB**. It also prevents all images from appearing, and consequently renders faster on my screen.

To specify headers and footers easily, I added **DO** which takes as its first argument, your specification of which of 3 headers or footers you wish to set, and as its subsequent arguments, the text you wish to place there.

If you write mathematical text, then you'll soon want equation numbering to be automatic, and for that matter, you'll probably also want your tables and figures to be numbered. For these, I created created table and figure numbering macros, and altered the equation macro (.**EQ**) to automatically number all my equations.

The following illustrate most of my macros. They aren't elegant; I can't even guarantee that they are bug free[17], but they work for me (most of the time). If you use them and run into difficulties, please feel free to contact me, and I'll try to assist. If you have a suggested fix, so much the better.

10.1 The Bullet Macro: BU

The macro creates bulleted indented paragraphs by default. The symbol can be changed at will. Given one of 6 option letters, plus a number (i.e. two arguments), numbered lists can be created. The syntax is:

```
.BU [[n|N|i|I|a|A num] | symbol]
```

The list must be terminated by a .**LP** command (a section command works just as well, and furthermore, it resets the list to bulleted). For example, I might write

- first line.
- second line

and so on. The same 'pretend list' with numbering would look like

1 first line.

2 second line

and so on. The text to create these two was:

[16] Hopefully, you'll find this HOWTO to be a useful reference, at least while you're getting started.

[17] While writing this HOWTO, I noticed that a reference to figure 3 was preceded by a comma - which I did not include. This is likely a bug in my macro.

```
.BU
first line.
.BU
second line
.LP
.BU n
first line.
.BU
second line
.LP
```

The macro looks like[18],

```
.\"  ***********************My BU macro**********************************
.\"
.\"  Predefine some registers and strings.  These are also reset with
.\"  every .NH [i] invocation.
.nr bu 0
.ds symbol \\(bu
.ds numeric
.de BU
.\"
.\"  If there is an argument, its either a symbol, or a numeric format & value
.\"
.if !’\\$1’’ \{\
.  ds symbol
.  ds numeric \$1
.  ie ’\\$1’n’ .af bu 1        \" set format to 1 ...
.  el \{\
.    ie ’\\$1’N’ .af bu 001    \" set format to 001 ...
.    el \{\
.      ie ’\\$1’i’ .af bu i    \" set format to lower case roman numerals
.      el \{\
.        ie ’\\$1’I’ .af bu I  \" set format to upper case roman numerals
.        el \{\
.          ie ’\\$1’a’ .af bu a\" set format to lower case alphabetic
.          el \{\
.            ie ’\\$1’A’ .af bu A    \" set format to upper case alphabetic
.            el \{\
.              ds symbol \\$1
.              ds numeric
.            \}
.          \}
.        \}
.      \}
.    \}
.  \}
.\"
.\"  If there is a second argument, it must be the start value
.\"
.  if !’\\*[numeric]’’ \{\
.    ie ’\\$2’’ \{\
.      nr bu 1
```

[18] Note that interpolated values (arguments to macros, strings, number registers) and special characters must be preceded by **two** backslash characters when used in a macro definition.

```
.    \}
.    el \{\
.      nr bu \\$2
.    \}
.  \}
.\}
.\"
.ie !'\\*[numeric]'' \{\
.  IP \\n(bu
.  af bu 1
.  nr bu \\n(bu+1
.  ie '\\*[numeric]'n' .af bu 1
.  el \{\
.    ie '\\*[numeric]'N' .af bu 001
.    el .af bu \\*[numeric]
.  \}
.\}
.el \{\
.  IP \\*[symbol]
.\}
..
```

10.2 The AN Macro

This is a very simple macro. If invoked, it sets a number register to a non-zero value. This number register is tested in my modified **NH** macro (illustrated next) and if found to be non-zero, resets the numbering for equations, figures and tables from zero for each chapter.

I define the macro as[19]:

```
.nr an 0
.de AN
.nr an +1  .\" count the invocations of .AN (out of interest)
..
```

10.3 Modified NH Macro

The **NH** is an important one. It sets section headings and performs resets of selected internal items. I wanted it to do more, and so copied the whole to my macro definition file and added a few lines (which I deliberately noted as mine, so as not to muck up the original). It depends on the existence of certain string and number registers, plus another macro that does paging on request.

```
.\"  **********************strings and registers*********************
.\"
.\"  Number registers 'eq' (equation), 'tb' (table #), 'fg' (figure #)
.\"  These are updated by invocations of .EQ, .TB and .FI
.\"  with 'eq', 'tb' and 'fg' preset to the next value for use prior to the
.\"  .EQ, .TB and .FI commands
.\"
.\"  If macro TB (or FG) is invoked, then number register tB (or fG)
.\"  is incremented so that other macros can determine if they were
.\"  every invoked.
.\"
.\"  string Ch is set (as required) in the NH macro following
```

[19] But have yet to use the count for anything.

```
.nr eq 1
.nr tb 1
.nr tB 0
.nr fg 1
.nr fG 0
.ds Ch
.ds Eq (\\*[Ch]\\n[eq])
.ds Tb  \\*[Ch]\\n[tb]
.ds Fg  \\*[Ch]\\n[fg]
.\"
.\" ****************************PG my page eject macro*******************
.\"
.ds PAGE This page intentionally blank.
.de PG
.\"            This macro tests the 'an' register.
.\"            If not > 0, it does nothing
.if \\n[an]>0 \{\
.  fl
.\"
.\"            For first level .NH invocations, ensure the heading
.\"            and text starts on an odd page number
.\"
.  ds dummy_string \\&
.  if !\\n[nl]u=\\n[HM] \{\
.    sp |\\n[.p]u       \" NOT at top of page, so eject
.    while !\\n[nl]u=\\n[HM] \{\
\\*[dummy_string]
.      fl               \" needed to add something to the page to allow an eject
.      sp |\\n[.p]u     \" STILL not at top of page (probably a diversion), so eject
.    \}
.  \}
.\"
.\"       Test:if odd page continue, else test Thesis mode
.\"            if Thesis mode continue, else print heading and eject
.\"
.  if e \{\
.    if !'\\*[thesis]'1' \{\
.      ce 1               \" place our message at the top of the page
.      ft R
\\*[PAGE]
.      ft P
.      sp |\\n[.p]u       \" go to next page, then reset footers
.      if '\\*[RF]'References' .ds RF
.      if '\\*[RF]'Index'      .ds RF
.      OF '\\*[LF]'\\*[CF]'\\*[RF]'
.      EF '\\*[RF]'\\*[CF]'\\*[LF]'
.    \}                   \" if !'\0'1'
.  \}                \
.\"
.\"                 Now on correct page, change footers
.\"
.  OF '\\*[LF]'\\*[CF]'\\*[RF]'
.  EF '\\*[RF]'\\*[CF]'\\*[LF]'
.\}                 \" if \\n[an]>0
```

```
..
.\" *****************************NH macro (modified)********************
.\"
.\"  In macro package 'ms', add chapter stuff to macro 'NH'
.\"
.\"  Most of the following is from the supplied tmac.gs macro package
.\"
.de @NH
.ie '\\$1'S' \{\
.    shift
.    nr nh*hl 0
.    while \\n[.$] \{\
.         nr nh*hl +1
.         nr H\n[nh*hl] 0\
.         shift
.    \}
.    if !\\n[nh*hl] \{\
.         nr H1 1
.         nr nh*hl 1
.         @error missing arguments to .NH S
.    \}
.\}
.el \{\
.    nr nh*ohl \\n[nh*hl]
.    ie \\n[.$] \{\
.         nr nh*hl 0\\$1
.         ie \\n[nh*hl]<=0 \{\
.              nr nh*ohl 0
.              nr nh*hl 1
.         \}
.         el \{\
.              if \\n[nh*hl]-\\n[nh*ohl]>1 \
.                   @warning .NH \\n[nh*ohl] followed by .NH \\n[nh*hl]
.         \}
.    \}
.    el .nr nh*hl 1
.    while \\n[nh*hl]>\\n[nh*ohl] \{\
.         nr nh*ohl +1
.         nr H\\n[nh*ohl] 0
.    \}
.    nr H\\n[nh*hl] +1
.\}
.\"  ***********************Dean's additions*****************************
.\"
.\"  With every .NH [i] invocation, reset the bullet stuff
.\"
.nr bu 0
.ds symbol \\(bu
.ds numeric
.\"
.\"  For first level .NH invocations, reset the equation, table and
.\"  figure numbering to 1, if the '.AN' macro was invoked
.\"
.if \\n[an] \{\
```

```
.  if \\n[nh*hl]=1 \{\
.    ds Ch \n[H1]-
.    nr eq 1
.    nr tb 1
.    nr fg 1
.    ds Fg  \*[Ch]\n[fg]
.    ds Tb  \*[Ch]\n[tb]
.    ds Eq (\*[Ch]\n[eq])
.\"
.\"  Users MUST put '.bp 1' after their abstract (.AB ... .AE)
.\"  to get the article text to start on page 1
.\"
.\"  Note that (oddly) this does not place the first chapter on page 1,
.\"  as NH/PG doesn't seem to recognize that it is at the top of page 1.
.\"  The solution (for now) is to put '.bp 0' after the abstract.  The
.\"  first page of chapter 1 then appears where it belongs (on the third
.\"  page, but numbered as 1).  Note that this applies when .AN has been
.\'  invoked.
.\"
.\"  For first level .NH invocations, ensure the heading
.\"  and text starts on an odd page number
.\"
.    ds RF Chapter \n[H1]
.    PG   \
.  \}                \" if \n[nh*hl]=1
.\}                  \" if \n[an]
.\"  *******************Back to the original NH stuff******************
.ds SN
.nr nh*i 0
.while \\n[nh*i]<\\n[nh*hl] \{\
.    nr nh*i +1
.    as SN \\n[H\\n[nh*i]].
.\}
.SH
\*[SN]
..
```

10.4 Figure Caption and Numbering: FI

The second last macro that I'll illustrate is my figure captioning and numbering macro. It permits me to automatically number and add captions to figures, as well as refer back to figures by number. An alternative scheme is used for equation references, and I'm considering modifying this macro to be more like the other. I'll show you that macro next.

The figure macro will place a figure number in the text (if the **N** option is invoked) or expect a following caption if it isn't. A short caption (about one to one and one half lines) requires nothing more than you type it in, and follow it with my modified **LP** macro (also shown here). For longer captions, you must use a **P** option (at least until I make that the default - feel free to add this yourself).

The comments in the macro should give you some insight into how it should be used.

```
.\"
.\" ********************************FI macro*****************************
.\"
.\"  The FI macro:  if invoked with an N argument, print the word and
```

```
.\"  do nothing else.  If a second argument, assume it is punctuation.
.\"  If no argument, treat as a figure caption.  All text following
.\"  will be in italics.  Follow with paragraph command (e.g. .PP) or .LP
.\"
.\"  Recommended coding:
.\"
.\"       .KF (or.KS)
.\"       -------a single blank line--------
.\"       .B1
.\"       .PSPIC file.eps size (or .PS ...PIC codes... .PE)
.\"       .B2
.\"       .FI P
.\"       first (of possibly several) line(s) of descriptive text.
.\"       .LP
.\"       -------a single blank line--------
.\"       .KE
.\"
.ds FIGURE Figure
.nr nmbr 0
.de FI
.nr nmbr 0
.ie '\\$1'N' \{\
.  NUM \\$2             \" test second arg for inc/decrement
.  ie !'\\*[num]'' \{\
.    nr nmbr \\$2       \" make a note of the value
.    shift 2         \" discard N and inc/decrement
.  \}
.  el .shift 1
.  nr fg +\\n[nmbr]
.  ft B
\\$2\\*[FIGURE] \\*[Ch]\\n[fg]\\fP\\$1
.  nr fg -\\n[nmbr]
.
.\"  Added option for .FI P linelength linestart, recommend
.\"  .FI P 5.25 5n   to make it look like a .QP entry
.el \{\
.  ie  '\\$1'P' \{\
.    ie '\\$2'' \{\
.      tm Warning: Figure \\*[Fg] caption length missing near line \\n[.c]
.      nr nbr 10n \" \n[.s]*10
.      ll (u;\n[.l]-\\n[nbr])
.      nr nbr 5n \" \\n[.s]*5
.      ds indent \\n[nbr]
.    \}
.    el \{\
.      ll \\$2
.      if !'\\$3'' .ds indent \\$3
.    \}
.    nr fin 2
.    br
.    di fig
.    ft B
\\*[FIGURE] \\*[Fg]:
.    ft I
```

```
.    nr fg +1
.    ds Fg \\*[Ch]\\n[fg]
.    nr fG +1
.  \}
.  el \{\
.    br               \" new method is to capture text up to .LP and decide then
.    di fig           \" by processing the diversion 'fig'.  Old method was .QP
.    nr fin 1
.    ft B
\\*[FIGURE] \\*[Fg]:
.    ft I
.    nr fg +1
.    ds Fg \\*[Ch]\\n[fg]
.    nr fG +1
.  \}
.\}
..
.\"
.\" ****************************@LP: my re-defined LP macro*******************
.\"
.de @LP
.\"
.\"            Where Dean's changes occur...
.\"            Assumes all figures centered on page/column, or if
.\"            2-3 side by side, then there is only one caption for all
.\"
.if !\\n[fin]=0 \{\
.  br
.  di
.  ie \\n[fin]=1 \{\
.    ie \\n[dn]<=\\n[.v] .ce
.    el .QP
.    fig
.  \}
.  el \{\
.    nf
.    in (u;\\*[indent])   \" QP indent is 5n, ll is 5.25i
.    ds indent
.    fig
.    fi
.    ll
.    in
.  \}
.  nr fin 0
.\}
.\"
.\"            Where Dean's changes end...
.\"
.par*start 0 0
.nr \\n[.ev]:ai \\n[\n[.ev]:PI]
..
```

10.5 Equation Numbering and References: QN

This last macro sets me up for referencing historical (i.e. those we've already formatted) and future equation references. To get the latter to work, you have to run *groff* twice. The first time, it complains and inserts a note into the text for equations that it couldn't find. As equations are found however, their references are added to a text file which is read by the second *groff* run (automatically). Thus the second time through, the macro locates the future reference and inserts the correct equation number in the text.

This is a little more sophisticated than the figure macro, and someday soon, I'll probably make **FI** work more like **QN**.

As an illustration, I might code an equation and give it a label (with the **QN** macro). Then I can refer to it later, as follows:

$$a = b + c \tag{1}$$

In equation 1 we saw that we could add two variables to assign a value to the third.

This was coded as

```
.QN A sample
.EQ
a = b + c
.EN
In
.QN R sample
we saw that we could add two variables to assign a value to the third.
```

Shown here are my modified **EQ** macro and my **QN** macro:

```
.\" ******************************EQ macro (modified)********************
.\"
.\"  In macro package 'ms', add automatic equation numbering to EQ
.\"
.\"  Most of the following is from the supplied tmac.gs macro package
.\"
.de @EQ
.br
.ds eqn*num "\\$2
.ie '\\$1'L' .nr eqn*type 0
.el \{\
.    ie '\\$1'I' .nr eqn*type 1
.    el \{\
.         nr eqn*type 2
.         if !'\\$1'C' .ds eqn*num "\\$1
.    \}
.\}
.\"  ************************Dean's additions*****************************
.if '\\*[eqn*num]'' \{\
. ds eqn*num \\*[Eq]
. nr eq +1
. ds Eq (\\*[Ch]\\n[eq])
.\}
.\"  *****************Back to the original EQ stuff**********************
.di eqn*div
.in 0
.nf
```

```
.
..
.\"  *****************************QN macro *****************************
.\"
.\"  This macro simply prints 'equation #' in the text, with possibly
.\"  trailing punctuation.  It also allows for referencing an equation.
.\"
.\"  To assign a "label" reference the current equation, use
.\"
.\"       .QN A label (single word)
.\"
.\"  To refer to an equation by "label" reference, use
.\"
.\"       .QN R label [punct]
.\"
.\"  To refer to an equation by increment, use
.\"
.\"       .QN N +-offset [punct]
.\"
.ds EQUATION equation
.ds output doc.ref
.ds references doc.ref
.de QN
.if !'\\*[references]'' \{\
.  sy touch \\*[output]
.  so \\*[output]
.  ds references
.  sy rm -f \\*[output]
.\}
.ie '\\$1'A' \{\
.  ds QN.\$2 \\*[Ch]\\n[eq]
.  sy echo .ds QN.\\$2 \\*[Ch]\\n[eq] >> \\*[output]
.\}
.el \{\
.  ie '\\$1'R' \{\
.    ie d QN.\\$2 \{\
\\$4\\*[EQUATION] \\*[QN.\\$2]\\$3
.    \}
.    el \{\
.      tm  ******* NO LABEL named QN.\\$2 was defined *******
\\$4\\fB*\\$2 UNDEFINED*\\fP\\$3
.    \}
.  \}
.  el \{\
.    ie '\\$1'N' \{\
.      nr eq +\\$2
\\$4\\*[EQUATION] \\*[Ch]\\n[eq]\\$3
.      nr eq -\\$2
.      ds Eq (\\*[Ch]\\n[eq])
.    \}
.    el \{\
\\$2\\*[EQUATION] \\*[Ch]\\n[eq]\\$1
.    \}
.  \}
```

```
.\}
..
.de DO               \" macro to redefine headers and footers
.if '\\$1'LF' \{\
.  shift
.  ds LF \\$*
.\}
.if '\\$1'CF' \{\
.  shift
.  ds CF \\$*
.\}
.if '\\$1'RF' \{\
.  shift
.  ds RF \\$*
.\}
.OF '\\*[LF]'\\*[CF]'\\*[RF]'
.EF '\\*[RF]'\\*[CF]'\\*[LF]'
.ds do FALSE
.if '\\$1'LH' \{\
.  shift
.  ds LH \\$*
.  ds do TRUE
.\}
.if '\\$1'CH' \{\
.  shift
.  ds CH \\$*
.  ds do TRUE
.\}
.if '\\$1'RH' \{\
.  shift
.  ds RH \\$*
.  ds do TRUE
.\}
.\"
.\"  Need the following test as PN is not defined if this is onvked too soon
.\"
.if 'do'TRUE' \{\
.  OH '\\*[LH]'\\*[CH]'\\*[RH]'
.  EH '\\*[RH]'\\*[CH]'\\*[LH]'
.\}
..
```

10.6 Small Caps

Ted Harding provided a macro to create small capitals in a recent Email message to the *groff* mail list. The following is a direct quote from his Email:

> Since I mentioned the small-caps issue in a previous mail, some of you may be interested in the following macros which do a primitive implementation of small-capitals.
>
> I think the main point to be careful of is: Don't change the point size within a smallcaps block (you're unlikely to need to do so anyway; but if you do it won't work right). If you need to do that, you'd better do the smallcaps by hand (or maybe make more than one block of it).
>
> The two macros **.smallcaps** (to turn it on) and **./smallcaps** (to turn it off) follow, with a small

example at the end.

His macros and a short example follow:

```
.de smallcaps
.nr .sc.ps (\\n[.s]*75/100)
.nr .cap.PS \\n[.s]
.char a \s[\\n[.sc.ps]]A\s[\\n[.cap.PS]]
.char b \s[\\n[.sc.ps]]B\s[\\n[.cap.PS]]
.char c \s[\\n[.sc.ps]]C\s[\\n[.cap.PS]]
.char d \s[\\n[.sc.ps]]D\s[\\n[.cap.PS]]
.char e \s[\\n[.sc.ps]]E\s[\\n[.cap.PS]]
.char f \s[\\n[.sc.ps]]F\s[\\n[.cap.PS]]
.char g \s[\\n[.sc.ps]]G\s[\\n[.cap.PS]]
.char h \s[\\n[.sc.ps]]H\s[\\n[.cap.PS]]
.char i \s[\\n[.sc.ps]]I\s[\\n[.cap.PS]]
.char j \s[\\n[.sc.ps]]J\s[\\n[.cap.PS]]
.char k \s[\\n[.sc.ps]]K\s[\\n[.cap.PS]]
.char l \s[\\n[.sc.ps]]L\s[\\n[.cap.PS]]
.char m \s[\\n[.sc.ps]]M\s[\\n[.cap.PS]]
.char n \s[\\n[.sc.ps]]N\s[\\n[.cap.PS]]
.char o \s[\\n[.sc.ps]]O\s[\\n[.cap.PS]]
.char p \s[\\n[.sc.ps]]P\s[\\n[.cap.PS]]
.char q \s[\\n[.sc.ps]]Q\s[\\n[.cap.PS]]
.char r \s[\\n[.sc.ps]]R\s[\\n[.cap.PS]]
.char s \s[\\n[.sc.ps]]S\s[\\n[.cap.PS]]
.char t \s[\\n[.sc.ps]]T\s[\\n[.cap.PS]]
.char u \s[\\n[.sc.ps]]U\s[\\n[.cap.PS]]
.char v \s[\\n[.sc.ps]]V\s[\\n[.cap.PS]]
.char w \s[\\n[.sc.ps]]W\s[\\n[.cap.PS]]
.char x \s[\\n[.sc.ps]]X\s[\\n[.cap.PS]]
.char y \s[\\n[.sc.ps]]Y\s[\\n[.cap.PS]]
.char z \s[\\n[.sc.ps]]Z\s[\\n[.cap.PS]]
..
.de /smallcaps
.rchar a b c d e f g h i j k l m n o p q r s t u v w x y z
..
.LP
This Line Will Be In Ordinary Capitals And Lowercase Letters
.LP
.smallcaps
This Line Will Be In Small Caps Instead Of
Ordinary Capitals And Lowercase Letters
./smallcaps
.LP
This Line Will Be In Ordinary Capitals And Lowercase Letters
```

This Line Will Be In Ordinary Capitals And Lowercase Letters

THIS LINE WILL BE IN SMALL CAPS INSTEAD OF ORDINARY CAPITALS AND LOWERCASE LETTERS

This Line Will Be In Ordinary Capitals And Lowercase Letters

Ted added in a subsequent Email that in most cases these macros (while set up only in terms of the ASCII characters A-Z) work for many cases of accented characters as well.

11 Customizing Equations and References

11.1 Equation Customization

As I learned more about *groff*, and used it for some course notes I was making, I soon discovered that it would be much easier to use abbreviations for some terms in the many equations that I wanted to format.

The **eqn** package provides such a facility. For the bulk of my 'writing' (such as it is), I invoke my *standard* abbreviation file at the beginning of each chapter. I do this as I compose each chapter,and I store them in separate files. In this way, I can format a chapter and scan it for syntax, spelling and all manner of other composition errors by piping the output to **gv**.

Since an equation definition always seems to insert the space for an equation, even if it contains nothing but abbreviations, I use a script (see the section titled **Useful Scripts**) to strip out the redundant copies.

The abbreviations that I employ are listed below. Note that the first token following the word **define** is the abbreviation subsequently available for use in an equation. The **eqn** processor replaces the token with the text following the token. I enclose all the following text in double quotes, and where I believe that the double quotes must be preserved, I enclose the entire expression (with its encompassing double quotes) by additional single quotes. This may not be necessary, but it doesn't seem to hurt.

Note that if you are examining the source that generated this HOWTO, you will find that I had to use additional backslash characters for formatting purposes. In other words, to make a comment appear (comments start with .\"), I had to type it as .\\".

```
.\"  *****************Define some useful equation stuff********************
.\"
.\"  MUST be manually added to the text (i.e. the .s file), but WILL
.\"  leave space for an equation.  Thus, may be worthwhile to include it
.\"  in the FIRST equation of each chapter.
.\"
.\"  MUST appear in the order
.\"
.\"
.\"
.\"  Note that after a paragraph reset (.LP, .IP, .PP, .QP, .AE etc.)
.\"  a following ".EQ, delim @@, .EN" will insert a blank, numbered
.\"  equation in the text.
.\"
.EQ " "
delim @#
define (( "left ("
define )) "right )"
define [[ "left ["
define ]] "right ]"
define ’’ ’left ""’
define  "left {"
define  "right }"
define ab "above"
define ov "over"
define fr "from"
define pa "partial"
define −inf "{- inf}"
define lc ’"\fR{\fP"’
define rc ’"\fR}\fP"’
```

```
define Fourier '"\s+4\f[ZCMI]F\fP\s0"'
define Laplace '"\s+4\f[ZCMI]L\fP\s0"'
define Comp '"\s+4\f[ZCMI]C\fP\s0"'
define l| "left |"
define r| "right |"
define prime "\(aa"
define ` "sup"
define _ "sub"
define prop "\(pt"
define iv "i vec"
define jv "j vec"
define kv "k vec"
define perp "\(pp"
define eval '~"\s+8\fR|\fP\s0"'
.EN
```

Occasionally, you might want some ad hoc expression to be abbreviated. Say for example, you expect to type **sin theta** many times, and it isn't in your default list (it isn't in mine). Simply define it at the beginning of the first equation in which it appears (or any equation prior to that first appearance), and then use the abbreviation thereafter. For example, you might define it and use it immediately as,

```
.EQ
define st "{sin theta}"
x = sqrt st ov x sup 2
.EN
```

which would result in

$$x = \frac{\sqrt{\sin\theta}}{x^2} \qquad (2)$$

Notice that I enclosed the **sin theta** expression in curly brackets. They ensure that I can use **st** as the numerator in a fraction (without having to remember to enclose it in the brackets). This saves additional keystrokes. Using additional braces does not affect the operation of **EQ**.

11.2 Reference Customization

In a manner similar to customizing equations, you can set up your references by means of a definition file. I use the following, and find it quite adequate, although recently an individual indicated that it was non-standard. You be the judge. The definitions that I use (and show here) were used to format this document.

```
.\"
.\" ********How I like my bibliographic references formatted***********
.\"
.\"
.R1
accumulate
bracket-label " [" "]" "; "
et-al " et al." 1 2 # 3
# date-as-label D.+yD.y%a*D.-y # qualifies the date in the reference text
# no-label-in-reference       # removes label in reference list
label "@ ' ' D.y%a*"          # adds disambiguating letter in text and label
sort A+D+                      # sort by first author surname, then by date
move-punctuation
.R2
```

12 Useful Scripts

If you examine my **makefile** in detail, you'll see that I invoke several scripts. These are not required for normal processing with *groff* and friends. I use them to generate an index, and to remove duplicate references to my reference and equation customizations (see the previous section).

For the sake of completeness (and not necessarily because they are good) I include them here, with some short explanation. I have no doubt that long time users of *groff* have even methods of accompishing their goals, and I dully expect to be able to include their implementations in future versions of this document.

If you have some (preferably) robust methods of customizing *groff*, please contact me so that I can include them (and give you the credit that you deserve). In the mean time, here are mine.

12.1 Scripts to Strip Redundant Customizations

The first three I call refer.script, eqn.script and eqn2.script. They are used to generate PostScript output without a table of contents or index being included.

Here is refer.script,

```
#n

#    A 'sed' script to remove all references to 'ref.s' or
#    'references.s' after the first

/ref\.s/{
x
s/ref\.s//
t already_there
x
h
p

:already_there
d
}

p
```

Here is eqn.script,

```
#n

#    A 'sed' script to remove all references to 'eqn.s' after the first

/eqn\.s/{
x
s/eqn\.s//
t already_there
x
h
p

:already_there
d
}
```

```
#    Line must be OK, print it

p
```

Here is eqn2.script,

```
#n

#    A 'sed' script to remove all references to second and
#    subsequent .EQ delim @@ .EN invocations

/^\. *EQ/{
N
N
s/\(delim  *..\n\. *EN\)/\1/
t got_it
p
b end

:got_it
x
s/\(delim  *..\n\. *EN\)/\1/
t end
x
p
x
b end
}

p
:end
```

12.2 Creating the Index

I create table of contents entries and index entries in a rather convoluted manner. The process involves 4 scripts and 3 other invocations of **sed** plus **uniq** and **sort**. These last 5 processes are shown in the makefile, so I will not repeat them here. The other four scripts follow.

The first is a shell script that adds table of contents entries to the source file. It looks for numbered and unnumbered chapter and section headings, and adds the appropriate source lines.

Here is toc.script,

```
#n
#
#    toc.script
#
#    A 'sed' script to add Table of Contents entries to a 'groff' file intended
#    for use with the 'gs' (Gnu's version of 'ms') macro package.
#
#    The 'n' on the first comment line inhibits printing.

#    This script also searches for BOLD and ITALIC invocations (.B, .I)
#    and uses these to create INDEX entries.

#    Search for the Numbered Heading command .NH [n]
```

```
/^. *NH\>/{
p
n
P
s/^/\*[SN] /
i\
.XS\
.if \n\[nh\*hl\]<2            \{\
.  ps \n[.s]+4-\n\[nh\*hl\]      \
.\}\
.ti (m;\n\[nh\*hl\]-1)
p
a\
.ps\
.XE
}

#    Search for the Section Heading command .SH
#    Note that the dummy substitution is needed for the final test
#    to avoid printing the section header line twice.

/^. *SH\>/{
p
n
P
s/\(^.*$\)/\1/
i\
.XS\
.ti (n;\n\[nh\*hl\])
p
a\
.XE
}

#    Watch for 'refer' invocations: save the last invocation in the hold space

/^\. *\[/,/^\. *\]/{
h
}

#    Append the commands to read the index file formatting in
#    (i.e. /var/tmp/x) and then
#    add the Table of Contents to the output file

#    If we encountered citations, we'll have '.]' in the hold space

${
p
x
s/^\. *\].*//
t got_refer

a\
.PG\
```

```
.so /var/tmp/x\
.DO LF \
.DO RF \
.TC
b no_refer

:got_refer
a\
.PG\
.XS\
.ps \n[.s]+2\
.sp\
References\
.ps\
.XE\
.DO LF \
.DO RF References\
.[\
$LIST$\
.]\
.PG\
.so /var/tmp/x\
.DO RF \
.TC

:no_refer
}

#    Find all BOLD or ITALICIZED text (.B single_word OR .B "multiple words")
#    and use it to create index entries

/^\. *[BI][     ]/{

#    If .IX or .IP encountered, skip the line

/^\. *IP\>/{
b not_this
}

/^\. *IX\>/{
b not_this
}

p

#    Delete the initial .B, .I or .BI

s/^\. *[BI]  *//
s/^\. *BI  *//

#    Remove trailing punctuation
#    do twice, in case have .B "several words" ) ( to generate
#    (several words) in the output
```

```
s/  *.[^ "@]* *$//
s/  *.[^ "@]* *$//

#    Strip @'s if they commence a line

/^@/{
s/@\([^@][^@]*\)@/\1/g
}

s/^/.IX /
p

:not_this
}

#    Strip the $ signs from the RCS stuff

/^\.EF\>/{
s/\$//gp
}

/^\.OF\>/{
s/\$//gp
}

#    Print everything else

t end
p
:end
```

Two of the other scripts (index.script and format.script) are **awk** scripts. The third (title.script) is another shell script. The three finish the process of setting up the index.

The index.script

```
#    index.script

#    Awk script to process index output from tmac.s macros

#    This is the first of 2 'awk' scripts.  It precedes 'format.script' in
#    a pipeline.  The 'title.script' (a 'sed' script) follows last:

#         index.script | format.script | title.script

#    In the Makefile, this sequence is preceded by a 'sed' command to replace
#    the ... sequence in the index entries in the error file by a TAB
#    This is output is sorted and made unique.

BEGIN {
  FS = "\t"
  start=1
  indent_space = "   "         # 3 space indent
  for (i = 1; i < 6; i++)
    last [i] = ""
```

```
}

#    Our indenting function

function indent (count) {
  while (--count) {
    printf ("%s", indent_space)
  }
}

#    Mainline
{
  n = split ($0, entry)        # store up to 5 entries (5th is line number

#    Strip out warning lines like   "indent cannot be negative", and
#                         "can't break line"

  if (match ($0, "warning: ")) {
    print "warning: encountered and ignored" > /tmp/groff_errors
    next             # try the next line of input
  }
  if (match ($0, "error: ")) {
    print "error: encountered by index.script:  line was:" > /tmp/groff_errors
    print "" > /tmp/groff_errors
    print "     " $0 > /tmp/groff_errors
    print "" > /tmp/groff_errors
    print "run terminated" > /tmp/groff_errors
    exit
  }

#    Find the first different entry

  for (i = 1; i < 6; i++) {
    if (entry [i] != last [i]) {
      j = i
      break
    }
  }

#    If the first entry is different, output a line feed

    if (start == 0) {
      if (j == 1) {
     start = 0
     printf ("\n")
      }
    }
    else
      start = 0

#    If only the line number is different, just add it

    if (j == 5)
      printf (", %s", entry [j])
```

```
    else {
      for (i = j; i < 6; i++) {
        if (i == 1)             # first word different: just print it
       printf ("%s", entry [i])

        else if (i < 5) {       # not the page number:
          if (length (entry [i]) > 0)     # non-blank word: go to next line
         printf ("\n");

          if (length (entry [i]) != 0) {
         indent(i)              # indent the word sufficiently
         printf ("%s", entry [i])
       }
        }

        else
       printf (", %s", entry [i])         # add the page number after a COMMA
      }
    }

  for (i = 1; i < 6; i++)             # copy current line to last line
    last [i] = entry [i]

  next                                # get next line of input
}
```

The format.script

```
#     format.script

#     Awk script to apply groff formatting to index output from GS macro

#     This script follows the 'index.script' invocation in a pipeline
#     of three:

#          index.script | format.script | title.script

BEGIN {
FS = "\t"
OFS = ""
lower = "abcdefghijklmnopqrstuvwxyz"
upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
}

#     Mainline

NF > 0 {
  printf ("%s\n", ".DO RF Index")
  printf ("%s\n", ".DO LF ")

  if ($0 !~ /^ .*/) {           # ignore lines beginning with spaces
    n = 1
    while ((newchar = substr ($1, n, 1)) !~ /[A-Za-z]/) {
      n++
```

```
      if (n == 25) {      # bad line
        newchar = oldchar
        break
      }
    }

    if (newchar ~ /[a-z]/) {              # convert first char to uppercase
      for (i = 1; i <= 26; ++i) {
     if (newchar == substr (lower, i, 1)) {
       newchar = substr (upper, i, 1)
       break
     }
      }
    }

    if (substr ($1, 1, 1) ~ /[0-9]/)
      newchar = ""

    if (newchar != oldchar) {       # next alphabetic character - add title
      printf ("\n\n%s\n", ".ne 4")
      printf ("%s\n", ".ti -2n")
      printf ("%s%s%s\n", "\fB", newchar, "\fR")
      printf ("%s\n", ".br")
      oldchar = newchar
    }
    printf ("%s\n", ".ne 2")
  }

  printf ("%s\n", ".ti -4n")
  printf ("%s\n", $0)
  printf ("%s\n", ".br")
}
```

The title.script

```
#    title.script

#    This is the third of three scripts (2 'awk', 1 'sed' to process
#    the index file created by 'groff'.  The three scripts are piped together
#    with several other UNIX commands.  The order is:

#         index.script | format.script | title.script

#    If this is run under 'gsed', then multiple comments are available.
#    If not then delete all comments.

#    Before the FIRST line, insert the following format codes

1i\
.bp\
.XS\
.ps \n[.s]+2\
.sp\
Index\
.ps\
```

```
.XE\
.SH \
Index\
.br\
\l'\n(.lu-\n(.iu'\
.br\
.2C\
.in +4n\
.na
s/   /  /

#    After the LAST line, reset the footer codes (RF is set in format.script)

$a\
.ds RF
```

13 Document Organization and a Sample Makefile

For shorter documents, like papers I might publish, or perhaps just a nicely formatted letter, I do the whole thing as a single unit. Longer documents I organize by chapters. That way I can work on a single chapter, with periodic reviews without having to wait on the processing of the entire work.

I simplify the organization by storing documents on a particular subject in a subdirectory of the directory where all my documents reside. On my system, it is **/home/provinsd/docs/s**. The **docs** subdirectory, has documents that are not *groff* formatted works; the **s** subdirectory just happens to contain everything that uses the *ms* macros. Pictorially, it looks like **Figure 3**.

I use **RCS** to track revisions of each document, or chapter. I even use RCS to track revisions of figures generated by **xfig** (which I used to create **Figure 3**).

Notice that **RCS** is included in each directory. This ensures that if I use the same name for a figure (say), that I don't clobber one with the same name in another subdirectory which corresponds to another document (and I've done that, unfortunately).

Notice that I have used a single **makefile**. I do this because then I can track everything I've done, and I don't need to duplicate a fairly complex set of dependencies.

Another file that is common for all *groff* documents (using the *ms* macros) is **tmac.ds**. This file contains all of my personal macros, and my customizations of the macro set called **tmac.s** which is found in **/usr/share/groff/tmac**.

The directories in the illustration (odds_ends, and approximation) contain shorter documents like this **HOWTO** which is in 'odds_ends', and longer works like the notes for an approximation course that I took recently.

Notice the naming structure for the chapters in 'approximation'. The first is **app.00.s**, and it corresponds to the cover page (i.e. the title, author and abstract). Chapter one is in **app.01.s**, and succeeding chapters have higher numbers. For running through my Makefile, this makes sorting easy.

For figures, I usually use **xfig**, but there are other programs that you may wish to use. **gnuplot** comes to mind for numerical data. Anything that can generate **encapsulated Postscript** files is suitable (except if you plan to create an **html** file with *groff*. This requires a **PNG** or **GIF** file). I create the former (when needed for html output) with the **convert** program that comes with the **ImageMagick** package. It does a satisfactory job, and also has a screen capture program **import** that can be useful at times.

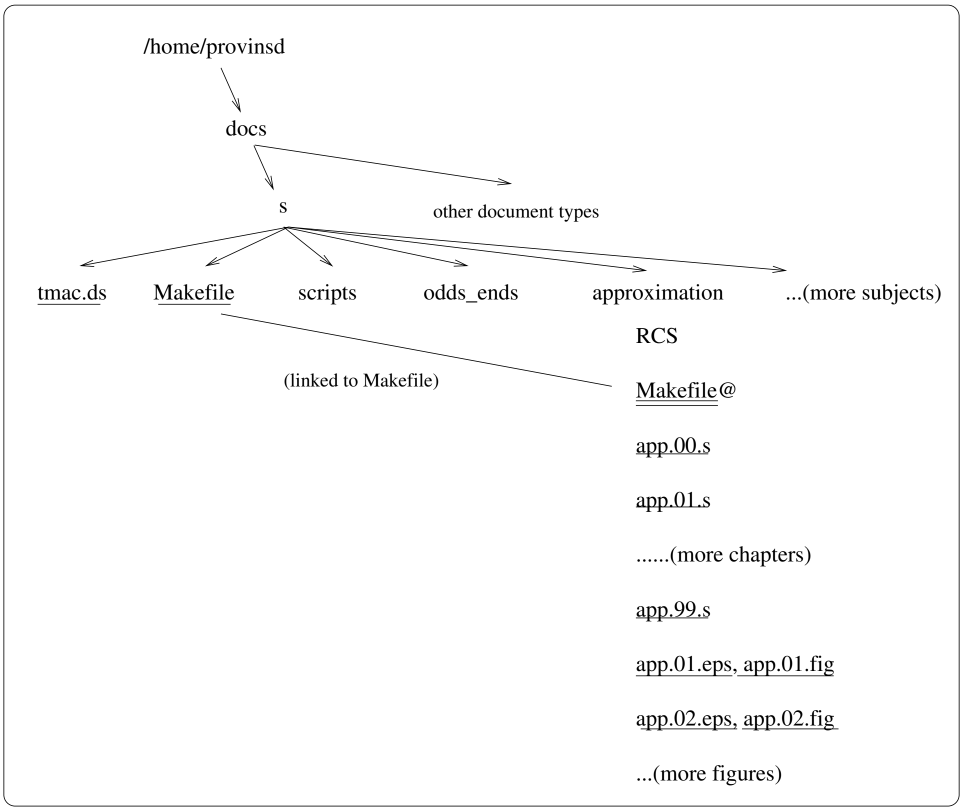


Figure 3: *How documents are organized on my system. Files are underlined, and the Makefile in the exploded subdirectory is linked to the Makefile in the parent directory. The file named* **tmac.ds** *contains my personal macros, and my customizations to the ms macros. Directory RCS contains all revisions to text, figures (.fig) and also gzipped versions of the encapsulated PostScript figures (.eps).*

One thing to remember about **xfig** is that if you're not set up properly, you may end up with **landscape** format for your diagrams. This probably won't be what you want. Simply start **xfig** as **xfig -portrait file.fig**.

13.1 The Makefile

The **makefile** makes everything go. The following is a stripped down version, with one document in sufficient detail to illustrate the process that I use. You may have a much better way, and I would be pleased to hear from you. The listing is quite lengthy; it is the result of trial and error (you know: two forward, one back). If you have a better way, and I'm sure that someone out there does, please let me know so that I for one will be better informed, and other readers will also benefit from your experience.

I tried reducing the font size by 4 points for the following listing in order to make it a little shorter, and I tried setting it in two columns, but unfortunately, the text became too small to read, even though it met my requirement of a fewer number of pages.

I hope that the comments included in the listing are sufficiently clear (naturally, they are to me). If you find them unclear, please drop me a line with your questions and suggestions for change. I will not (in this version of the **HOWTO**), explain the **makefile** in detail. Suffice to say that you read the dependencies such as **.s.ps: $*.s** as a file ending in .**ps** depends on one depending on .**s**, and that file is named *something like name*.**s**. The word *name* in *make-speak* is the name without a suffix of the current prerequisite that has been modified more recently than the current target. Please refer to [Oram 1991] for all you might want to know about **make**.

```
#     This 'generic' makefile for GROFF source, using the 'gs' macro package
#     (which is the equivalent of the 'ms' macros) will generate from
#     a file name 'file.s' , the following:

#     Starting output page number is defined by Make Environment variable P
#     To set it from Make's command line, use:
#     >make file P=n       where 'n' is the page #

#     To output selected pages, use Make Environment variable O
#     To set it from Make's command line, use (an optionally comma
#     delimited list):
#     make file O=n,n-m,-m,n-   where 'n' is a specific page, 'n-m' is a range
#                    '-m' is UP TO page 'm', and 'n-' is FROM page 'n'

#     >make file.gvx       output groff to ghostview with TOC/INDEX
#     >make file.psx       output groff to PS file   with TOX/IDX

#     >make file.gv        output groff to ghostview  without TOC/IDX
#     >make file.lpr       output groff to PS printer without TOX/IDX
#     >make file.ps        output groff to PS file    without TOX/IDX

#     >make file.lp        output groff to more       without TOC/IDX
#     >make file.txt       output groff to text file  without TOC/IDX

#     >make file.p         check the input '.s' file for picture  errors
#     >make file.t         check the input '.s' file for table    errors
#     >make file.e         check the input '.s' file for equation errors

#     >make file.x         create an index for the specified source file
#     >make file.toc       add TOC and index entries to      source file

#     >make file.egz       compress an EPS file and store in RCS
#     >make file.pgz       compress a  PS  file and store in RCS
#     >make file.html      make an HTML version of a groff source file

#     This makefile makes use of several 'scripts', namely:

#     @${SCRIPTS}/error.script $< $*.nul
#     sed -f ${SCRIPTS}/toc.script $< > $*.toc
#     awk -f ${SCRIPTS}/index.script            | ${tee_d} \
#     awk -f ${SCRIPTS}/format.script           | ${tee_e} \
#     sed -f ${SCRIPTS}/title.script            | ${tee_x} \
#     @${SCRIPTS}/clean.script

#     Following are documents that are composed to chapters, each of
#     which is in a separate file.

#     I show just one here, to keep what is a busy makefile shorter!!
#     It says that the word '407'

407  =    407.??.s
```

```
#     The following are used for debugging.  Uncomment them, and comment
#     out the succeeding set.

# tee_a    = tee /var/tmp/a |
# tee_b    = tee /var/tmp/b |
# tee_c    = tee /var/tmp/c |
# tee_d    = tee /var/tmp/d |
# tee_e    = tee /var/tmp/e |
# tee_f    = tee /var/tmp/f |
# tee_g    = tee /var/tmp/g |

tee_a=
tee_b=
tee_c=
tee_d=
tee_e=
tee_f=
tee_g=

tee_x= tee /var/tmp/x |

.SUFFIXES: .s .ps .lp .gv .e .p .t .lpr  .x   \
           .txt .toc .nul .gvx .sp .lpr2 .pic \
        .eps .psx .gz .egz .pgz .html

RM    = rm -f

P     = 1
O     = 1-

#     TMAC_DIR names the directory in which to search for local macro
#     definitions, or the re-definitions of macros defined in the standard
#     macro packages (such as the 'ms' package, sometimes called 'gs').

TMAC_DIR= /home/provinsd/docs/s

#     TMAC names the local macro file that is to be included on the
#     command line.  Groff looks for it in TMAC_DIR.  Our local macro file
#     is named 'tmac.ds'

TMAC = -M${TMAC_DIR} -mds

#     MACRO names the macro packages to be read from the standard location
#     which is '/usr/lib/groff/tmac', or from the user's own library
#     of macros.
#     As of Debian 2.0, the macros are in /usr/share/groff/tmac, and
#     the -ms macro, which WAS called -gs, is again called -ms

MACRO= -ms ${TMAC}

#     SOELIM names the processor that precedes the execution of GROFF
#     and include the contents of '.so filename' statements in the
#     input stream.  Note that PREPROCESSOR commands intended for TBL or EQN
#     will NOT be processes unless 'soelim' (or some other processor)
```

```
#    expands the '.so filename' statements

#SOELIM   = cat

#    Version 1.11 of groff (from Debian, at least) is in /usr/bin
#    The current version (1.15: Dec 20, 1999) is in /usr/local/bin

#PREFIX   = /usr/bin
PREFIX    = /usr/local/bin

SOELIM    = ${PREFIX}/soelim
TROFF= ${PREFIX}/groff
PIC  = ${PREFIX}/pic
EQN  = ${PREFIX}/eqn
TBL  = ${PREFIX}/tbl

#    groff 1.11 now calls 'grefer' 'refer'
#    Note that as 'refer' is a pre-processor, it isn't controlled by
#    '.if', or 'ie' constructs.

REFER= ${PREFIX}/refer

TROFF_OPTS = -etpU -n${P} -o${O}

HTML_OPTS  = ${TROFF_OPTS} -P-a

VPATH= RCS
CO   = co

#    The refer options are
#    -e   accumulate references to the end or until
#         .[
#         $LIST$
#         .]
#         is encountered

#    -S   set default bracket labelling, and label contents

REFER_OPTS = -e -S

#    Where my bibliography is stored.  File 'refer' is the
#    index into the actual bibliography.  'refer' will complain, but find
#    the text file it needds anyway, as the path is stored in the index.

DATABASE= /home/provinsd/bibtex/refer

#    Various scripts that I use to clean up or modify the groff input or
#    ouput are contained in this directory.

SCRIPTS   = /home/provinsd/docs/s/scripts

#    Send output to the printer via lpr2 (double sided printing)
#    Invoke it for groff file 'file.s' as 'make file.lpr2'.  This will
#    generate a PostScript file, and then run the shell script 'lpr2'
```

```
#    onthe output.

.ps.lpr2:  $*.s
     lpr2 $<

#    Check spelling in the source file

.s.sp:     $*.s
     ispell $<

#    Format -mgs groff source file and send to ghostview

.ps.gv:    $*.s
     gv $*.ps

#    Format -mgs groff source file and send to Postscript printer 'lpr'

.s.lpr: $*.s
     lpr $*.ps
     @echo "----------- Print job for file $*.ps spooled -  ----------"

#    Format -mgs groff source file for HTML (input FILE.s, output FILE.html)

.s.html:  $*.s
     cat $<                                   | \
     sed -f ${SCRIPTS}/refer.script                |  \
     sed -f ${SCRIPTS}/eqn2.script                 |  \
     ${SOELIM}                           | \
     ${REFER} ${REFER_OPTS} -p${DATABASE} 2> $*.nul  | \
     ${TROFF} ${HTML_OPTS}  ${MACRO} -markup -Thtml - > $*.html 2>> $*.nul
     @${SCRIPTS}/error.script $< $*.nul
     @echo "------------- HTML file $*.html created  -------------"

#    Format -mgs groff source file for PS (input FILE.s, output FILE.ps)

.s.ps:     $*.s
     cat $<                                   | \
     sed -f ${SCRIPTS}/refer.script                |  \
     sed -f ${SCRIPTS}/eqn2.script                 |  \
     ${SOELIM}                           | \
     ${REFER} ${REFER_OPTS} -p${DATABASE} 2> $*.nul  | \
     ${TROFF} ${TROFF_OPTS}  ${MACRO} -Tps - > $*.ps 2>> $*.nul
     @${SCRIPTS}/error.script $< $*.nul
     @${RM} $*.nul
     @echo "------------- PostScript file $*.ps created  -------------"

#    Format -mgs groff source file and send to screen (line printer mode)

.s.lp:
     cat $<                                   | \
     sed -f ${SCRIPTS}/refer.script                |  \
     sed -f ${SCRIPTS}/eqn2.script                 |  \
     ${SOELIM}                           | \
     ${REFER} ${REFER_OPTS} -p${DATABASE}          |  \
```

```
     ${TROFF} ${TROFF_OPTS}  ${MACRO} -Tascii - > $*.lp 2> $*.nul
     @${SCRIPTS}/error.script $< $*.nul
     more $*.lp
     -rm $*.lp

#    Format -mgs groff source file and send to a text file
.s.txt:
     cat $<                                    | \
     sed -f ${SCRIPTS}/refer.script                 | \
     sed -f ${SCRIPTS}/eqn2.script                  | \
     ${SOELIM}                                | \
     ${REFER} ${REFER_OPTS} -p${DATABASE}           | \
     ${TROFF} ${TROFF_OPTS}  ${MACRO} -Tascii - > $*.txt 2> $*.nul
     @${SCRIPTS}/error.script $< $*.nul
     @echo "---------------- Text file $*.txt created  ---------------"

#    Run groff source through geqn, gtbl and tpic to check for errors
.s.p:
     echo Picture check
     echo
     ${SOELIM} $< | ${PIC} > /dev/null
     echo

.s.t:
     echo Table check
     echo
     ${SOELIM} $< | ${TBL} > /dev/null
     echo
.s.e:
     echo Equation check
     echo
     ${SOELIM} $< | ${EQN} > /dev/null

#    Add Table of Contents and Index entries

.s.toc:
     @echo
     @echo Adding Table of Contents and Index entries to a groff source file
     @echo
     sed -f ${SCRIPTS}/toc.script $< > $*.toc
     @echo "---------------- New file $*.toc created  ----------------"

#    Format -mgs groff source file for processing the TOC and Index

.toc.nul:
     @echo
     @echo Processing groff source for Table of Contents and Index entries
     @echo
     > /var/tmp/x
     cat $<                                    | \
     sed -f ${SCRIPTS}/refer.script                 | \
     sed -f ${SCRIPTS}/eqn2.script                  | \
     ${SOELIM}                                | \
     ${REFER} ${REFER_OPTS} -p${DATABASE}           | \
```

```
	${TROFF} ${TROFF_OPTS}  ${MACRO} -Tps - > /dev/null 2> $*.nul
	@${SCRIPTS}/error.script $< $*.nul
	@echo "--------------- Index file $*.nul created  ---------------"

#	Process the groff source to make up an index:  index to /var/tmp/x
#	which is read via '.so' inserted by toc.script

.nul.x:
	@echo
	@echo Reading index file $*.nul and formatting for groff
	@echo
	sed -e 's/ *\.\.\. */	/' $*.nul		| ${tee_a} \
	sed -e 's/^\/\(.*\)/\1\//'		| ${tee_b} \
	sort -bf -t '	' +0 -4 +4n		| ${tee_c} \
	uniq				| ${tee_d} \
	sed -e 's%^\(.*/%/\1%'		| ${tee_d} \
	awk -f ${SCRIPTS}/index.script		| ${tee_e} \
	awk -f ${SCRIPTS}/format.script		| ${tee_f} \
	sed -f ${SCRIPTS}/title.script		| ${tee_x} \
	cat - > $*.x
	@echo "--------- File /var/tmp/x created for use by groff  ----------"

#	Format -mgs groff source file which now includes an
#	Index and Table of Contents and send to ghostview

.x.gvx: /var/tmp/x
	@echo
	@echo Running groff on updated source file
	@echo
	cat $*.toc				| \
	sed -f ${SCRIPTS}/refer.script			| \
	sed -f ${SCRIPTS}/eqn2.script			| \
	${SOELIM}				| \
	${REFER} ${REFER_OPTS} -p${DATABASE}		| \
	${TROFF} ${TROFF_OPTS}  ${MACRO} -Tps > $*.ps 2> /dev/null
	gv $*.ps
	@echo "------- Processing of updated source file complete  ------"

#	Create a postscript file with Table of Contents and Index

.x.psx: /var/tmp/x
	@echo
	@echo Running groff on updated source file
	@echo
	cat $*.toc				| \
	sed -f ${SCRIPTS}/refer.script			| \
	sed -f ${SCRIPTS}/eqn2.script			| \
	${SOELIM}				| \
	${REFER} ${REFER_OPTS} -p${DATABASE}		| \
	${TROFF} ${TROFF_OPTS}  ${MACRO} -Tps > $*.ps 2> /dev/null
	@echo "------- Processing of updated source file complete  ------"

#	 Do a make file.egz to copy into RCS an eps file as a compressed
#	 version.  Note that the dependent source file (file.s) needs
```

```
#     to have the eps files listed as dependencies (does that make
#     sense?).  We can remove the copies by running 'make clean'.

.eps.egz:
	@touch .extracted_eps_files
	@touch TMPFILE
	-@grep $< .extracted_eps_files > TMPFILE 2>&1
	if [ ! -s TMPFILE -a ! -L $< ]; then                \
	  echo "File $< not extracted: do";            \
	  mv $< $*;                             \
	  gzip $*;                         \
	  ci -m"gzipped version of $<" $*.gz;          \
	  echo "$< checked in";                        \
	fi
	rm TMPFILE

.gz,v.gz:
	${CO} $<

.gz.eps:
	gunzip $<
	mv $* $*.eps
	if [ -f ./RCS/$*.gz,v ]; then echo "$*.eps" >> .extracted_eps_files; fi

clean:
	@${SCRIPTS}/clean.script
	rm -f *.html

#    Here is the document I call '407'.  There are several chapters, each
#    in their own file (e.g. 407.01.2, and so on).  To build the entire
#    document, we'll make something called '407.s' using the alias '407'
#    defined at the beginning of this makefile.

#    Notice that certain chapters have dependencies on encapsulated
#    postscript files.  Ig we make those chapters, and the files are not
#    in the current directory, they'll be copied out from the RCS directory
#    (i.e. by 'co').  If we make the entire document, they'll also be
#    copied out.  A 'make clean' will remove them from teh current
#    directory (but not the RCS directory).

407.02.s:  407_fig_02_01.eps 407_fig_02_02.eps
407.03.s:  407_fig_03_01.eps 407_fig_03_02.eps
407.07.s:  407_fig_07_01.eps 407_fig_07_02.eps

407.s:     ${407}  407_fig_02_01.eps 407_fig_02_02.eps \
                407_fig_03_01.eps 407_fig_03_02.eps \
           407_fig_07_01.eps 407_fig_07_02.eps
	cat ${407} > $@
```

14 Formats Other Than PostScript

The default output device for *groff* is PostScript, a page description language developed by Adobe Systems Incorporated. PostScript is the traditional output format for many UNIX programs that generate formatted text. For information about PostScript, see either (or both) [Geschke 1985] or [Warnock 1985].

The **-T** option on the *groff* command line allows the user to select any of several output devices. Those available include the TEX **dvi format**, a device independent format which requires further processing before printing, much like the the *groff* system works, **X75** and **X100** X11 preview formats, **ascii** for typewriter-like devices, and **latin1** for typewriter devices using the **ISO Latin-1** character set.

With release 1.15 of *groff*, comes also **html** format.

15 Questions About *groff*; Some With Answers

15.1 How Did You Create the TEX symbol?

To be honest, I had to copy it. I found the appropriate *troff* codes in a manual page (which one, I have forgotten, but it was on my Linux system). The string definitions are:

```
.if t .ds TX \fRT\\h'-0.1667m'\\v'0.20v'E\\v'-0.20v'\\h'-0.125m'X\fP
.if n .ds TX TeX
.if t .ds LX \fRL\\h'-0.36m'\\v'-0.15v'\s-2A\s0\\h'-0.15m'\\v'0.15v'\fP\*(TX
.if n .ds LX LaTeX
```

The **.if** statements mean that if *troff* (or *groff*) is running, then use the first and third definitions for **TX** and **LX**. If *nroff* is running, then use the second and fourth.

Assuming PostScript output, the first and third string definitions generate TEX and LATEX respectively. As we saw earlier, they would be invoked as *[TX] and *[LX] respectively.

15.2 How Do I Convert This HOWTO's Source into Format XX?

It isn't quite that easy, as I have some custom scripts and **so**. files that I use for all my documents, but I'll see if I can cut it down to size and make it available. Be warned though - they look a little (a lot??) messy. To make them useful, I'd have to post a bunch of stuff, and the user (you) would have to put everything in the right place.

Perhaps though I can put the needed macros in the source file (not sure if that is possible, but think so), and I may be able to skip the index, which I think is very useful (I hate having to rely on a table of contents to find things - it takes so much reading to find a reference again!).

Skipping the index (for now at least) would mean few (perhaps no) scripts would be required.

I'll look into the whole mess and return to this question again. hopefully, the result will be acceptable.

15.3 How Do I Change the Paper Size?

Changing the paper size for the output of *groff* requires altering a single value in the device description file for a particular device. As shipped, *groff* supports several different devices, including PostScript (devps), LaserJet4 (devlj4) and DVI (devdvi). Inside each of subdirectories corresponding to the supported devices (all are found in **/usr/share/groff/font/**), is a file named **DESC**.

The **DESC** file contains name and values, such as **print** and **paperlength**. This latter name indicates the length of a page in **points** per inch times **sizescale** units. On my machine here in Canada, the default paper size is **US letter**. On machines in Europe, and elsewhere, **A4** is commonly used.

Since **A4** is longer than **US letter**, the value of 792000, which is 11 inches multiplied by 72 points per inch times 1000 (the value of **sizescale**). **A4** would use a value of 841890, which corresponds to a paperlength of 297 mm.

To account for a different **paperwidth**, (**US letter** is 216 mm, or 8.5 inches, and **A4** is 210 mm, or 8.27 inches), macro calls for page offset, line length and other parameters, should be made. For the *ms* macros, see the table listed earlier in this document.

15.4 How Do I Convert PostScript to ASCII or HTML?

PostScript to ASCII conversions may be performed via *ps2ascii*. This program is a Ghostscript translator from PostScript or PDF to ASCII (i.e. plain text). Apart from first converting PostScript to ASCII, and then adding suitable formatting commands to generate HTML, I have no idea how one can convert directly to HTML.

15.5 How Do I Convert *groff* Source to HTML?

From Gaius Mulley comes the following answer.

Use the -Thtml device option ie:

```
groff −Thtml filename.troff > filename.html
```

15.6 How Do I Convert *groff* Source to ASCII?

From Gaius Mulley comes the following answer.

To generate ASCII from *groff* or *troff* source, use the **-Tascii** option on the command line to *groff* and pipe the result to **col**. The command would be,

```
groff −Tascii filename.troff | col −bx > filename.ascii
```

where the **-bx** option replaces tabs with spaces and removes overstrike characters.

15.7 Is There a *groff info* File?

Trent Fisher has contributed a rudimentary groff.texinfo file and this is now a part of the *groff* distribution. Should an interested party wish to extend this work, their assistance would be gratefully received.

15.8 How can I do Cyrillic with *groff*?

Recently the following question was posed:

```
On 29−Oct−99 Grigoriy Strokin wrote:
> Hello,
>
> How do I prepare Cyrillic documents in PostScript
> using groff, provided I have Cyrillic fonts for Ghostscript?
>
> In fact, some cyrillic letters appear as expected,
> but some other are replaced by groff to funny characters
> such as Thorn or multiply sign.
```

The following is a transcription of the response:

> I'm not familiar with Cyrillic fonts for Ghostscript, and these may possibly have a problem with missing characters, for all I know. The following assumes that you have a PostScript font with a complete Cyrillic character set (possibly from ghostscript, of course).
>
> Provided you have "copied" the Cyrillic fonts to groff (i.e. you have installed them as groff fonts in ..../groff/fonts/devps) you could handle this according to some variant of one of the two following methods.
>
> METHOD 1
>
> Look, in this directory, at the font file for a particular font (e.g. I have a font which I pulled from the Web called "College Cyrillic" whose font file I have called "CCy", quite arbitrarily, to

which the following applies).

This may start with a list of "kernpairs", but in any case you will come to a line with the word "charset".

After the "charset" line you will see several columns of information (the number of columns depends on the font), most of which is "font metrics", one line for each character. Some characters will be Latin, others Cyrillic.

The first column will be either a single ASCII character, a groff name for the character, or "---". The last column but one will be an octal number of the form "0pqr". For instance, the standard octal code for the letter "A" in English is "0101". In decimal, this is "65" which is the ASCII code for "A". The rule for conversion from octal "0pqr" to decimal is 64*p + 8*q + r.

You now need to identify which decimal codes correspond to which Cyrillic characters (if they also have groff names, so much the better; but it doesn't matter). The most direct way to do this is to print out a table consisting (in groff input) of lines like

241   \N'241'

where (in this example) the "241" is the decimal form of the octal 0361.

The groff escape sequence "\N'237'" causes the character with octal code "0361" in the correct font to be printed. So you must make sure that the current font is the Cyrillic font when this is printed (e.g. by using "\f[...]", or the request ".ft ..." on a line by itself, where "..." is the name of the font, i.e. the name of the font file you were looking at above, such as "\f[CCy]" or ".ft CCy").

Once you have got this information, you can then link this to your input.

Myself, being primarily in a Latin environment with Cyrillic being required exceptionally, I use two schemes. One is for single Russian words or very short pieces of Russian text, and is based on transliteration. For instance, the Russian "YA" (looking like an upper-case reversed "R") is denoted by the groff escape "\(YA". Similarly, the lower-case version is "\(ya".

This is pre-defined (in private macros) by

.char \(YA \N'241'

(This character is at 241 in most -- but not all -- of the Cyrillic PostScript fonts which I have. Therefore, if that is also the case with you, you should switch to Cyrillic using a macro like

```
.Cyr
.fnt fontname
.tr \(xx \N'nnn'
.tr ...  ...
  ...
.
```

If the Cyrillic character encoding depends on which font you are using, it might be better to make the name of the macro the same as the name of the font. Then you can use font-specific translations.

The other (for extended text where using escapes like "\(YA" throughout would be too tedious) is based on a similar character translation which allows me to type in Latin characters using a transliteration-like convention (I can't be bothered to fiddle with keyboards) like (using Russian alphabetical order)

```
.tr A \N'225'           [Cyrillic char identical to Latin "A"]
.tr B \N'226'           [Cyrillic char like large 'b']
.tr V \N'247'           [Cyrillic char identical to Latin "B"]
.tr G \N'231'           [Cyrillic char identical to Greek GAMMA]
..
.tr \(YA \N'241'
```

so that, for instance, the word for "Probability" would be entered as "Vero\(yatnoste\(i:". This reduces the use of escape sequences to the minimum. (The sequence "\(i:" stands for "i-kratkoi"

You could also, of course, Grigoriy, define the compound character

.char \(iy i\(i:

so that your name would be typed "Grigor\(iy".

METHOD 2 (which depends on info from Method 1)

In your environment, which is presumably primarily Cyrillic, you will probably be using a Russian keyboard, and a screen font based on the ISO font encoding iso-8859-5 (Cyrillic) where various numerical codes above 128 correspond to Cyrillic characters.

For the reasons given above, the iso-latin-5 character encoding may differ from the encoding in the PostScript font you are using. Therefore, you should carry out the above check as to which decimal codes correspond to which printed characters in the Cyrillic PostScript font. Some of these may coincide with the iso-8859-5 encoding; others may be different. In that case, what you need to do is to define a translation for the anomalous codes, as above, so that you can type the characters normally using the keyboard, they will print correctly, and you will also see them on the screen correctly.

In the first place, this is a lot of work; but once it has been done you will have an easy life!

The best of luck, Ted.

15.9 What is the Meaning of Double Quotes in Macro Arguments?

Ted Harding answers this curious question.

> The special meaning of the double-quote character in macro arguments is to group space-separated elements into a single argument. However, in that context the opening " should be preceded by a space and the closing " should be followed by a space.

15.10 Footnotes and Indented Paragraphs

An unfortunate bug exists in *groff*, in that attempting to set an automatically numbered footnote on the hanging text of an indented paragraph exists. Specifically, trying the following,

```
.IP word[20]:
some text...
```

will result in the correct footnote number in the footnote, but an incorrect number in the text (it is incremented by one). From that point on, the footnote numbers in the text are in error.

15.11 Where can I get Windows binaries of *groff*?

Thor M. Gil has ported *groff* to Microsoft's most recent operating systems (95, 98 and NT). You can obtain it, plus some installation information at **http://www.cs.vu.nl/˜tmgil/misc/wingroff.html**. Other recommended software, such as **Winzip** and **GSview** are also available at that site.

15.12 Is There a Document Like the "Troff User's Manual" for *groff*?

One of the maintainers (Ted Harding) has embarked on such a document. As this is not a trivial undertaking, it may not be available for some time. Readers are encouraged to subscribe to the *groff* mail list to follow the development, or to volunteer to assist Ted in order to speed it along.

For immediate use, are a series of overheads that Ted used in a recent presentation to his local Linux user's group. These foils, with their *groff* source files, are available from the following European site: **http://www.manlug.mcc.ac.uk/19991120/talk.tar.gz**. See the file **talk.txt** for a description of the contents.

15.13 How Should I Name My Macros?

Werner Lemberg suggested the following to an inquirer:

> "The usual thing in high level languages like C is to always have a unique prefix, say, 'www-'. I think that your choice to have '<' as a prefix is optimal. It is, AFAIK, unique, and it always reminds the user that it has something to do with HTML."

15.14 How Can I Avoid Name Clashes with Macros in Use?

Werner Lemberg suggested the following (in the context of name clashes when creating a new macro package which might be included with an existing package such as **ms**):

> "The only possibility to avoid name clashes is to check all macro definitions in the standard macro packages which come with *groff*. There is a small tool which might help you in debugging name clashes.

```
groff -mtrace -man -Tlatin1 2>groff.log > groff.txt
```

> "This will get a trace of all macro calls/definitions in tmac.an. You can for example, get an ordered list of all used macros with:

```
cut -d ' ' -f 5 groff.log | sort | uniq > man.log.macrolist
```

> "Note that if errors occur, these will be included in the list. You will have to weed these out yourself."

Let me add that it might be prudent to avoid two letter, upper case macro names when modifying or adding to the **ms** package. Since *groff* accepts long names, any string longer than two characters, whether upper or lower case, is recommended.

Anyone reading to this point will have observed that I did not follow my own advice (see the earlier section on macro customization). The macros developed and exhibited earlier were created while I was still a novice. I have no doubt that someday I'll be changing them to reflect my advice to you.

15.15 How Can I Get a List of All My References?

Unless you write your own shell script, or perhaps a Perl program, thereby allowing yourself the luxury of being selective, there is only one way of getting such a list that I know of. The following **sed** script is an example.

```
#n   A script used in the command:
#
#    cat *.home *.uc | sed -f sed.script > all_titles.s
#
```

```
#    The file 'all_titles.s' is ready for processing by 'groff -ms'

#    Begin by inserting two lines in the output
1{
i\
.so /home/provinsd/docs/s/ref.s\
.LP
}

#    When a blank line is reached, insert
#    .[
#    copy the hold space into pattern space
#    replace all embedded newlines with spaces and append
#    .]
#    print the line
#    delete the contents of the pattern space
#    place the now empty patter space into the hold space for the next cycle
#
/^[  ]*$/{
i\
.[
g
s/\n/ /g
a\
.]
p
s/.*//
x
}

#    Append the last name of each author or editor to the hold space
/^%[EA] /{
s/.* \(\&[a-zA-Z].*\)/\1/
H
}

#    Append the date to the hold space
/^%D /{
s/^%D //
H
}

#    Append the title to the hold space
/^%T /{
s/^%T //
H
}
```

and associated command,

```
cat /home/provinsd/bibtex/refer_bib.{home,uc} | sed -f sed.script > all_titles.s
```

will do the trick. As described below, it grabs the author's surnames, the date of publication and the full title of the work. This seems to be enough to make each entry unique. The beginning of the output from this commands, for my bibliographic database looks like,

```
.so /home/provinsd/docs/s/ref.s
.LP
.[
Cruz June 1983 Experiences with Altimeter Data Gridding
.]
.[
Heiskanen Moritz 1966 Physical Geodesy
.]
.[
Krakiwsky 1990 The Method of Least Squares:  A Synthesis of Advances
.]
.
.   ... and many more...
```

After running those through *groff* with the **ms** macro package, I get pages of references neatly sorted by author and date. These look like,

```
Acton 1970.
    Forman S.  Acton, Numerical Methods that Work, Harper &
    Row, Publishers, New York, USA (1970).

N.R. Adam et al. 1997.
    N.R. Adam  and  A.  Gabgopadhyay,  Database  Issues  in
    Information Systems, Kluwer Academic Publishers (1997).

........and so on.......
```

In this example, I have found all the title entries (%T) in the two files in which I keep my database (/home/provinsd/bibtex/refer_bib.home and refer_bib.uc). The **cat** command concatenates the two files and passes all the text to **grep** which searches for the '%T' commencing in column 1. The lines that are found are then passed to **sed** for processing. This script inserts two lines **before** line 1 (the .so and the .LP commands). It then watches for %T in column 1, and when this is found, it inserts .[before the line, removes the '$T ' and follows it with the closing macro that **refer** expects (.]). The .**so** command reads my specific formatting requirements for references from the file I have indicated.

The output (I called it all_titles.s) is then run through *groff* with the **ms** macro package. The final result is as you see it above: a sorted list of references.

Are there any problems using this method? Unfortunately there is one. It shows up errors that you have made in your entries. If, for example, there are duplicates, **refer** will complain and drop the duplicate. If you missed entering an author, the output will show only a date. If, as I do, you use the database to record journal names and library call numbers, then strange looking references will appear in the output. These really aren't **refer** problems, but they are annoyances.

If you would like to recommend a replacement **sed** script that corrects any problems that you encounter, I'll be happy to include it in the next release of this HOWTO, *and* give you credit!

15.16 How Do I Make the Output of gv Bigger?

If you're unfamiliar with **gv**, then I recommend it to you, and suggest that you explore its lengthy documentation (i.e. the 'man' page). But to answer the question, and thus get you going, try using the numeric keys above the QWERTY section of the keyboard. Numbers 1 through 5 increase the font size in reasonable increments (1.414, 2, 4, 8 and 10 times the original)[21].

[20] The plus and minus keys do the same thing.

15.17 How Can I Defer Page Numbers in a Floating Keep

Steve Izma provided a solution to this perplexing (at least to me) problem. It involves the use of *transparent throughput* (section 10.6 of [Ossanna 1992].

In his reply to a writer who had created a figure macro called *Fc* which sent to standard error certain information, Steve suggested the following:

```
Since .KF uses a diversion to capture the floating object, you
can delay the interpretation of \n(PN with something like:


.de fig_toc
.ie ’\\n[.z]’’ \{\
.    tm FIG:.sp
.    tm FIG:\\$1\tT{
.    tm FIG:\\$2
.    tm FIG:T}\t\\n(PN
.\}
.el \!.fig_toc \\$*
..

and call fig_toc in your Fc macro rather than immediate calls to
tm. You need to give fig_toc the 2nd and 3rd arguments to Fc
```

This is interpreted as defining an additional macro called *fig_toc* which upon invocation immediately tests to see if it is in a diversion (the **.ie** statement). If not, it executes 4 **.tm** statements which place the desired information into the standard error file. If it is in a diversion, then it places an invocation of itself into the floating keep (the **.el** statement).

Notice that that this last statement begins with the two character sequence \!. This tells *groff* not to execute the *fig_toc* macro now, but only to strip off the \! characters.

When the diversion has ended and the floating keep is being positioned on the page, the macro is invoked. Only at this time is the page number (\\n(PN) set to its current numeric value.

16 Other Sources of Information

Several of the references cited earlier are worth looking at, if copies can be located in your local library. In particular, the book by Dougherty and O'Reilly is useful, although not particularly so as a reference[22]. The Sun Microsystems document is very good, and is based on the PostScript files mentioned below.

There are several documents available in PostScript format which describe the use of *troff* and associated processors. These are available at **http://www.kohala.com/start/** in the *troff resources* link and at **http://cm.bell-labs.com/cm/cs/cstr.html**. Of particular use is [Ossanna 1992].

At **http://plan9.bell-labs.com/7thEdMan/index.html** may be found the original *troff* sources for the documentation that was shipped with the **Unix Seventh Edition Manual**. Included is material on *troff* and related processors.

A selection of *groff* gems may be found at **http://www.eas.slu.edu/People/RBHerrmann/GROFF/index.html**. Things like creating business cards and using groff to make large format posters for presentations. This has been provided by rbh@eas.slu.edu.

In a recent posting to the *groff* mail list, the following URL was cited: **http://mirror.viii-**

[21] This writer finds the text too wordy, and the index not sufficiently so.

lo.krakow.pl/bugs/security/bugs/mUNIXes/groff.html. It discusses certain security issues surrounding some *groff* commands. Note that these are disabled by default via tne **-S** option.

The *groff* maintainers have set up a mailing list and web site to provide interested users with the latest releases, bug fixes and problem solving facilities. In addition, a moderated newsgroup named **gnu.groff.bug** is available.

To subscribe to the **mail list**, send an Email to **groff-request@ffii.org** with the word **subscribe** in either the header or the mail body. Then, to post to the list, send your email to **groff@ffii.org**. An acknowledgement of your request to join will be received, and you will be assigned a password. Mail received by the list will be archived at **http://ffii.org/archive/mails/groff/**.

The web site is **http://groff.ffii.org/**. There, the latest version of *groff* and friends is always available.

References

Dougherty 1987.
: Dale Dougherty and Tim O'Reilly, *UNIX Text Processing,* Hayden Books, A Division of Howrd W. Sams and Company, Indianapolis, Indiana, U.S.A. (1987).

Geschke 1985.
: Charles Geschke, *PostScript{®' Language Tutorial and Cookbook,* Addison-Wesley Publishing Company (1985).

Kernighan 1978.
: B.W. Kernighan, M.E. Lesk, and J.F. Ossanna, Jr., "Document Preparation," *The Bell System Technical Journal,* 57, 6, part 2, pp. 2115-2135, American Telephone and Telegraph Company, Short Hills, New Jersey, U.S.A. (July/August 1978).

Kernighan 1974.
: B.W. Kernighan and P.J. Plauger, *The Elements of Programming Style,* McGraw-Hill, New York, U.S.A. (1974).

Kernighan 1976.
: B.W. Kernighan and P.J. Plauger, *Software Tools,* Addison-Wesley, Reading Massechusetts, U.S.A. (1976).

Kernighan 1991.
: Brian W. Kernighan, "PIC - A Graphics Language for Typesetting User Manual," Computing Science Technical Report No. 116, AT&T Bell Laboratories, Murray Hill, New Jersey 07974, USA (May, 1991).

Knuth 1986.
: Donald E. Knuth, *The TeXbook (ISBN 0-201-13447-0),* Addison-Wesley (1986).

Lamport 1985.
: Leslie Lamport, *LaTeX - A Document Preparation System,* Addison-Wesley (ISBN 0-201-15790-X) (1985).

Oram 1991.
: Andrew Oram and Steve Talbott, *Managing Projects with make,* O'Reilly and Associates, Inc. (1991).

Ossanna 1992.
: Joseph F. Ossanna and Brian W. Kernighan, "Troff User's Manual," Computing Science Technical Report No. 54, AT&T Bell Laboratories, Murray Hill, New Jersey 07974, USA (November, 1992).

Sequin 1975.
: C.H. Sequin and M.F. Tompsett, *Charge Transfer Devices,* Academic Press, New York, U.S.A. (1975).

Sun Microsystems 1988.
Sun Microsystems, “Using NROFF and TROFF” in *Sun Microsystems System Documentation,* Sun Microsystems (May 1988 Revision A).

Warnock 1985.
John Warnock, *PostScript{®} Language Reference Manual,* Addison-Wesley Publishing Company (1985).

Index

Table of Contents