# tmac.diss

troff macros for
scientific articles and theses

Tadziu Hoffmann
**hoffmann@usm.uni-muenchen.de**

April 1, 2004

**Abstract**

This article describes the usage (and partly the implementation) of the **diss** macro package for troff. I have originally written this macro package for typesetting my doctoral thesis; it should, however, be easily adaptable to many other types of articles or books. The package was written entirely from scratch, based on ideas from the Troff User's Manual (Ossanna and Kernighan, 1992) and the Unix manpage macros (see *man*(5)). No special customization features have been implemented: customization is expected to be done by simply copying and changing or rewriting the macros whose functionality needs to be adapted.

# Contents

# 1

# Introduction

The reason for endowing text formatters with a macro facility is that no predefined set of high-level formatting functions will be appropriate for all purposes (the same also holds for macro packages), and marking up all text with low-level functions is too cumbersome and error-prone. The macro package described here has been specifically designed for typesetting thesis-like papers in a book-style layout. Of course, many macro packages already exist for similar purposes, but these are usually very complex and difficult to customize. I have therefore decided to write my own macro package.

The question remains why one would use troff for this, and not, for example, TEX. In fact, there is little doubt that for most purposes TEX is the best text formatter generally available. Troff, on the other hand, is antiquated, primitive and rather low-level, and lacks TEX's typographical finesse. But troff's primitiveness also makes it "easy to understand", and hacking together a workable macro for some specific task is usually conceptually simple and straightforward. The real fun, however, in tinkering around with troff is that it (and your macros) sometimes behave strangely different from what you'd expect — analyzing these quirks provides hours of entertainment (much better than doing crossword puzzles).

This macro package is not intended as a "standard" like **mm** or **ms**. Rather, it should be seen as a useful starting point for deriving macros more suited to your specific needs. Extensive customization via user-settable parameters is not implemented; instead, editing or augmenting the macros is encouraged. What this means is that this package is definitely not targeted at beginners — some understanding of troff's fundamental typesetting concepts is assumed. Also, no attempt has been made at catching errors during processing. All people crazy enough to voluntarily use troff should be reasonably expected to be versed enough in its odd ways to be able to debug what they (and the authors of macro packages :-) write. Neither are the macros designed to cover all contingencies. Manual intervention may be necessary in certain cases.

This macro package has only been tested with GNU troff. I have, however, attempted to limit the use of GNU troff specific requests in the hope that these macros can be adapted to Unix troff without too much effort and with little loss of functionality.

# 2

# Document structure

The general structure of a document is as follows (comments in italics):

**.CH** *(chapter)*
chapter heading text
**.SE** *(section)*
section heading text
**.PP** *(paragraph)*
text and control lines
. . .
more chapter and section headings and text
. . .
**.AX** *(appendix)*
more chapters . . .
**.IX** *(index) (not yet implemented)*
**.TP** *(title page) (—front matter begins here—)*
title page text and control lines
**.AS** *(abstract)*
abstract text and control lines
**.CT** *(table of contents)*

The reason for requesting the table of contents at the end is that **troff** has no means of writing large chunks of stuff (e.g., entire diversions) to external files. Thus, it's easier to format the table of contents into a diversion while processing the main text part, output it at the end, and then rearrange the pages (using an external program, such as **sed**) before running the device postprocessor. To simplify this rearranging, the whole front matter gets formatted as a single block at the end of the file. The title page macro resets the page counter to 1 and sets the page number format to roman numerals. Otherwise, the title page and abstract macros do little more than begin a new page and set the "no page header"

flag. You are free to design your own title page with whatever font and size controls you require. The abstract macro **AS** can be given an optional argument, which will be used instead of the word "Abstract".

The **AX** macro begins the appendix part of the paper: it resets the chapter counter to 1 and sets the chapter number format to alphabetic; appendices are normal chapters with a different numbering scheme.

## 2.1. Sectioning macros

Four levels of sectioning are provided: chapter (**CH**), section (**SE**), subsection (**SU**), and sub-subsection (**SS**). Section headers are not output until a paragraph (**PP**) occurs. This allows consecutive headers (for example, section, subsection, and sub-subsection) to be output together, if necessary on a new page should the available space not suffice, without a page break occurring within this block. (Section headers at the very bottom of a page without text following on that page don't make for a particularly elegant layout.) Multiple consecutive section headers of the same level, while admittedly being a rather stupid thing to do, will usually work, except in the case of chapters, due to complications with forcing a page break. If you need this for some reason, then simply insert an empty paragraph between the two chapters.

## 2.2. Headers may contain math: $\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!}$

. . . as the example shows. This is one of the primary reasons for me wanting to write my own macro package[1], as in many standard packages the section header text is given as an *argument* to the header macro, instead of being read as input *following* the macro call, and thus may have problems with **eqn**-processed material. Nevertheless, a little cosmetic problem remains: the math is printed in the regular typeface, instead of in bold as would be appropriate for the style chosen here for the section headers. This would have been easy to remedy by switching **eqn** to the appropriate bold and bold italic fonts; however, as mathematical symbols are taken from the symbol font, which seldom is available in a bold variant, the resulting equations usually appear extremely uneven in density. Thus, leaving the math in the non-bold fonts is still the more acceptable solution.

In very simple cases, typesetting the math "manually" can yield style appropriate for the headers. For example,

```
.SU
Example: The
.EM n
= 3 wavefunctions of hydrogen
.PP
```

gives

### 2.2.1. Example: The *n* = 3 wavefunctions of hydrogen

and also appears correctly in the table of contents, since the **EM** macro automatically chooses the italic or bold-italic font depending on whether the current font is roman or bold. Alternatively, you could test for the header processing type and set the font accordingly (italic or bold italic). Testing for the header processing type is also useful for introducing strategic breaks that should only be effective in the section headers, as in

[1] The other reason is, of course, that this promotes me to the level of "wizard" in the hierarchy of Unix users (well, except for that matter of the device drivers . . . ).

```
.CH
Footnotes, cross references,
.if \n(#==1 .br
and indexing
.PP
```

the result of which can be seen on page 19. The test shown in the example will be true for section headers and false otherwise (page headers, table of contents, and normal text). Without the explicitly requested break, the break would have occured between "and" and "indexing", which is less pleasing to read. (Of course, the same effect could have been achieved here with an unbreakable space between "and" and "indexing", but in the example shown the break *always* occurs in the section header, even if everything would fit on a single line, in contrast to the solution with the unbreakable space.)

# 2. 3. Long (multi-line) section headers are allowed, but will be truncated in the page header; truncation is indicated with an ellipsis

Okay, I'll admit this is a rather artificial sounding section header, but I had to make my point somehow. There are two good reasons for implementing this behavior. The first one has to do with æsthetics: the page header should indicate what the page contains, but should be unobtrusive otherwise; its size and shape should be somewhat predictable. And facing page headers on even/odd pages should definitely have the same height to avoid an "unbalanced" look.

The second reason is purely technical: the page header is actually printed last, after the body has already been laid down. This allows the page header to list (sub-)sections that are *begun* on that page. (It's the last one that counts.) The downside of this is that space for the page header is limited, since it is allocated independently of the page header's contents[2].

Perhaps the better solution would be simply to have the sectioning macros accept an optional "short version" to be used in the page headers (and maybe in the table of contents); it would be the user's responsibility to see to it that this "short version" does not exceed the allotted space for the page header. Alternatively, the user can always redefine the strings **LH** and **RH**, which are normally set by the section header macros.

[2] Printing the page header in its "natural" height could of course be achieved with one more level of diversion, but I have not yet encountered a case where this was really necessary. I have therefore stuck with the simpler implementation.

# 3

# Page layout

## 3.1. The "margin" macros, PO, PE, LL, TM, BM, TH, and PL

All macros discussed in this section require a single argument. The default units for the horizontally oriented macros **LL**, **PO**, and **PE** are ems (same as for the **ll** and **po** formatter requests); the default units for the vertically oriented macros **PL**, **TH**, **TM**, and **BM** are "*V*"s (same as for the **pl** request).

The **PO** macro sets the page offset (left margin) for both odd (right) and even (left) pages; the **PE** macro overrides this page offset for even pages. The **LL** macro sets the line length; it also stores this value in a number register to be used as a default, to which the line length will be reset by the section heading macros. This is done to prevent local changes of the line length with the **ll** request from affecting succeeding sections. No macro exists for setting the right margin; the right margin will simply be *paper width – page offset – line length*.

The **PL** macro sets the page length; defaults are A4 (29.7 cm) for troff, 66 lines for nroff. The **TM**, **BM**, and **TH** macros set the top margin, bottom margin, and text height, respectively. These quantities are not all independent; the following table gives an overview of which numbers are affected by which macros.

| macro | sets | keeps | adapts |
|---|---|---|---|
| **PL** | page length | top margin, bottom margin | text height |
| **TM** | top margin | page length, bottom margin | text height |
| **BM** | bottom margin | page length, top margin | text height |
| **TH** | text height | page length, top margin | bottom margin |

The page header is approximately centered in the top margin.

## 3.2. The page style macros, PY, PX, and NH

The **PY** and **PX** macros require a single numeric argument specifying the desired page header and footer style. **PY** sets the default style; **PX** overrides this style for the current page only. The standard setup uses the following style numbers: 0 — neither page header nor footer. 1 — page header with page number and chapter/section name and a decorative rule; no page footer. 2 — no page header; page footer with page number (to be used on the first page of a chapter).

As a mnemonic shortcut, the macro **NH** is provided to request "no header" for the current page. This is an abbreviation for "**PX 0**".

## 3.3. The page header and footer macros, HD and FT

These macros are called by the page finalization macro to draw the page header and footer. They can be redefined by the user to change the way page headers and footers appear. The macros must space absolutely to the desired postion on the page, and should query the **PX** number register to act according to the current requested page style (see previous section).

## 3.4. The "stretchable space" macro, SP

If called with zero or one argument(s), this macro simply invokes the **sp** formatter request with the given argument (but always causes a break). If called with an optional second argument, it additionally provides stretchable space up to the amount specified by that argument, for the purpose of aligning the page text with the bottom margin: if the distance of the unstretched text to the bottom margin (including footnotes) is less than the total stretchable space requested for that page, then that much space will be proportionately divided among the stretchable spaces; otherwise, no stretching occurs.

This macro is mainly intended to be called from within high-level formatting macros, such as used for specifying section headers or paragraphs; however, it is also recommended to use this macro instead of the "raw" **sp** request, since it incorporates an **ne** invocation that prevents space from being output at the bottom of the page, necessary for proper functioning of the bottom-margin-alignment mechanism.

## 3.5. The paragraph style macros, PI, NI, PD, and DD

The **PI** macro sets the indent to be used on the first line of all paragraphs, except those immediately following a section header (which are always unindented). Additionally, the **NI** macro can be used to explicitly request that a paragraph be not indented. The **PD** macro sets the spacing (interparagraph distance) between paragraphs. For reasons of æsthetics, some manuals of style recommend setting either the paragraph indent or the paragraph spacing (but not both) to zero.

The paragraph macro **PP** automatically provides stretchable space of the same size as the interparagraph distance. Thus, if a paragraph spacing of zero is requested, this will be honored even if other space is stretched. The behavior can be changed, of course, by editing the paragraph macro.

The "display distance", set by the **DD** macro, is similar to the paragraph spacing, but is used as space separating displayed equations, lists, etc. from the surrounding text.

# 4

# Font control

## 4.1. The “font family” macro, FF

In this macro package, fonts are referenced mainly by number, not name. This allows changing fonts globally by mounting them on the appropriate positions, without having to change other macros. The current scheme makes use of the following convention:

| font position | font mounted | examples |
|---|---|---|
| 1 | roman/regular | Utopia regular, Helvetica |
| 2 | italic/oblique | *Utopia italic*, *Helvetica oblique* |
| 3 | bold | **Utopia semibold**, **Helvetica bold** |
| 4 | bold italic/bold oblique | ***Utopia semibold italic***, ***Helvetica bold oblique*** |
| 5 | computer (typewriter type) | `Courier` |
| 6 | slanted symbol | *Συμβολ σλαντεδ* |
| 7 | symbol | Συμβολ |

The macro **FF**, given the family name as argument, mounts the corresponding fonts on the first four positions. For this to work, the four chosen fonts must be “registered” under the family name; this is accomplished by editing the macro and adding an appropriate line.

The normal symbol font is used for upper case greek letters and other mathematical symbols. The slanted symbol font is used for the lower case greek letters, and should be scaled so that these match the x-height of the main font family, and slanted to match the slant of the italic font. (Edit **symbolsl.pfa** in the groff distribution files; the default is tuned to the Times family. Or create an analogous file for each font family you use, name this new slanted symbol font after the family (for example, Times-Symbol-Slanted), and have the **FF** macro mount this font instead. Generate a metrics file for this font (**getafm**, **afmtodit**) and make a corresponding entry in the file **download**.)

## 4.2. The font switching macros, R, I, B, C, RI, RB, . . .

For convenience, macros are provided to set one or several words in a different style. The macros **R**, **I**, **B**, and **C** set their argument in roman, italic, bold, and computer-font, respectively. If called without arguments, they will switch to the requested font, which then remains in effect until switched to a different font by a similar call to another of these macros, an "**f**"-escape, or an explicit "**ft** *n*" request. If called with two (resp. three) arguments, only the first (resp. only the second) argument will be set in the requested font; the remaining argument(s) will be set in the current font but concatenated (without space) to the font-switched text. This is useful, for example, to attach punctuation in the current font to a word in a different font, as in "**C \(lq vi \(rq**", which will put quotes around the word "vi" in the computer font.

To alternate (up to six) arguments between two specific fonts, without intervening space, the macros **RI**, **RB**, **RC**, **IR**, **IB**, **IC**, **BR**, **BI**, **BC**, **CR**, **CI**, and **CB** may be used.

## 4.3. The "*emphasize*" macro, EM

This macro is used to set apart a word (or several words) from its surroundings by temporarily switching to a different font (roman vs. italic, or bold vs. bold italic). It works similarly to the explicit font-switching macros, but the advantage of using this macro is that you do not need to keep track of what the current font is, and that it thus works correctly also in section headers. (Note how the word "emphasize" is set in bold italic in the section header above, but in regular italic in the table of contents.)

## 4.4. The font size and line spacing macros, SI, FS, and VS

The **SI** macro sets both the font size and baseline spacing. It must be called with one argument, the requested point size; the baseline spacing is set correspondingly. The baseline spacing is deliberately not set directly proportional to the font size: experience shows that larger fonts look better with a proportionately smaller baseline spacing than smaller fonts. The apparently trivial recipe "baseline spacing is point size plus two" works quite well under most circumstances. (Actually, **SI** preserves the leading defined by **FS** and **VS**, whatever this may be. The default is a 10-point font size on a 12-point baseline spacing, specifying a leading of 2 points.)

**FS** sets the global default font size for the document text; **VS** sets the corresponding default baseline spacing. (Note that **FS** is not called **PS** because the latter is reserved for the **pic** preprocessor.)

## 4.5. The "smaller" macro, SM

This macro helps in constructing "fake small capitals" for those cases where you don't have a real small caps font. It does this by a combination of scaling down and reducing the height of the font. Compare the following examples:

| | |
|---|---|
| OSSANNA and KERNIGHAN, 1992 | isotropic downscaling |
| OSSANNA and KERNIGHAN, 1992 | height reduction only |
| OSSANNA and KERNIGHAN, 1992 | combination of the above |

The last one is to be preferred over the other two. (For some fonts, the second works best. I haven't yet seen a font with which the first one looks good.) Needless to say, the scaling factors should be adapted to the font used. (It also depends on whether you want to scale the small caps all the way down to the x-height, as in this example, or leave them somewhere between the x-height and the cap-height.) At the

moment, this must be done by editing the macro; in the future, I might add these factors to the list of registered fonts, so that the correct factors will be automatically set with the font family selection macro.

For convenience, the macro may be called with one, two, or three arguments. If called with two (resp. three) arguments, the first (resp. the first and last) argument(s) will not be size-reduced, but simply concatenated to the rest. For example,

```
There has been mounting evidence of a prominent discrepancy
when trying to explain the intensities of
.SM [Ne ~III ]~36~\(*mm
and 3868~\(oA in
.SM H ~II ~regions.
```

will give (assuming that “`tr ~`” is in effect)

> There has been mounting evidence of a prominent discrepancy when trying to explain the intensities of [Ne III] 36 μm and 3868 Å in H II regions.

Of course, this entire scheme is only practical for isolated words. If you need to set long passages in small caps, obtaining a real small caps font is strongly recommended.

A hint for the æsthetically impaired: don’t use these fake small caps with a font of (nearly) uniform line thickness, as most sans-serif fonts, such as Helvetica. This will usually not give acceptable results. In the following example with fake small caps the big caps appear too black in proportion to the other letters:

> SHORT TEXT WRITTEN ENTIRELY IN FAKE SMALL CAPS.

(Actually, the fake small caps are too light, but as regular text would normally contain more small caps than big caps, it’s the big caps that stand out.) This non-uniformness in density makes such text very irritating to read. In most serif fonts with variable stroke widths the effect isn’t *quite* as bad, but it’s still pretty obvious. (You can’t fool a trained eye.)

# 5

# Lists

## 5.1. The list macros, BL, NL, TL, LE, and LI

Here is a summary of the list formatting macros in list style:

- ▷ Three types of lists are provided:
  - bulleted lists (**BL**)
  - numbered lists (**NL**)
  - tagged lists (**TL**)
- ▷ Lists are begun with **BL**, **NL**, or **TL**, and are ended with **LE**. List items are introduced with **LI**.
- ▷ Lists may be nested up to a total of six levels deep, irrespective of type.
- ▷ The list macros accept the following optional arguments:

  **bulleted list:**
  List indent, bullet type.

  **numbered list:**
  List indent, numbering style.

  **tagged list:**
  List indent.

  **list item (in case of tagged list):**
  Tag. If more than one word is used as the tag, they must be surrounded by double quotes to be treated as a single argument.

▷ For these arguments, the following considerations apply:

**list indent**

i. The list indent is the distance from the current indent to the indent of the list *text*. Bullets, numbers, and tags extend to the *left* of this indent. The list indent should have an explicit "**+**" sign.

ii. Changing the list indent is mainly intended for tagged lists, for example to adapt to the longest tag present. (Useful if all tags are relatively short.) Tags themselves are indented by 2 ens, so the list indent for tagged lists should always be greater than this.

iii. As all list macros use the indent macro (see below) internally, special care must be taken not to mangle the layout with stray or improperly nested **IN** requests within lists.

iv. To change the bullet type or the numbering style (second argument) but keep the default list indent (first argument), supply an empty (**""**) first argument.

**numbering style** Must be a number format as accepted by the "**af**" formatter request. The numbering style is remembered for future numbered lists at this nesting level. The default style is arabic.

**bullet type** Can be any string (and may include escape sequences). The bullet type is remembered for future bulleted lists at this nesting level. The default bullets are •, □, ○, ∗, –, and →.

And that's basically it.

# 5.2. The "indent" macro, IN

This macro implements a simple "indentation stack". Usage is "**IN +***n*" to begin a block of text indented by a width *n*, and "**IN**" without arguments to get back to the previous indent level. Indentation may be nested.

# 6

# Figures, tables, and equations

## 6.1. Floats and keeps

The float mechanism implemented here isn't quite as sophisticated as that of LaTeX, but it works reasonably well. Floats are encapsulated with **BF** and **EF** (see the example in Figure 6-1 on page 16), and are processed in a separate environment (like footnotes) to minimize interaction with the normal text. If floats are available, they are always output (as many as can fit) at the beginning of the next page. At the moment, no provision exists to place floats at the bottom of the page. (However, a new chapter flushes all pending floats, beginning on the current page.)

A keep is like a float except that it doesn't float. It always appears within the text at the place where it is requested; if not enough space remains on the page, the remaining space on the page is left empty and a page break is forced. Keeps are created with **BK** and **EK**. Keeps (and floats) *cannot* be nested.

Figure and table captions (to be used within floats and keeps) are begun with **FC** (figure caption) and **TC** (table caption) and ended with **EC**. Figures and tables are numbered automatically. Note that it is possible to place more than one figure and/or table within one float, if it is deemed necessary that they not be separated. However, the user must make sure that the float can still completely fit on one page; otherwise, horrible things may occur. (Essentially, what happens is that the float (and all subsequent ones) will never be output, because each page decides the float is too large to fit and thus saves it for the next page, which, alas, has very much the same opinion.)

Both **BF** and **BK** accept an optional argument, the name of the macro called in the input stream to terminate the float/keep. (The defaults are, naturally, **EF** and **EK**.) These are necessary for constructing "wrappers" around the float/keep macros.

```
.BF
.PSPIC figure.ps
.FC
.LB fig:example
PostScript graphic included with the
.C PSPIC
macro.
.EC
.EF
```

**FIGURE 6-1.** — Example showing the use of the float and figure caption macros[3].

# 6.2. Equations

Displayed equations (**eqn** code between **EQ** and **EN** macros) are centered and numbered automatically. Here, the numbering style is (*chapter.number*), where *chapter* is the chapter number and *number* is a running number beginning with 1 in each chapter:

$$\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!} = \lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n \tag{6.1}$$

The box macro is taken from the GNU **eqn** manual page. (Equations are not boxed by default.)

Multi-line displayed equations must have each line of the equation enclosed by **EQ** and **EN**. For example, to get

$$H_c = \frac{1}{2n} \sum_{l=0}^{n} (-1)^l (n-l)^{p-2} \sum_{l_1 + \ldots + l_p = l} \prod_{i=1}^{p} \binom{n_i}{l_i}$$
$$\cdot \left[(n-l) - (n_i - l_i)\right]^{n_i - l_i} \cdot \left[(n-l)^2 - \sum_{j=1}^{p} (n_i - l_i)^2\right] \tag{6.2}$$

we type

```
.EQ
H sub c = mark
1 over 2n sum from {l^=^0} to n ( - 1) sup l (n - l) sup {p - 2}
back 50 "" sum from {l sub 1 + ... +^ l sub p ^=^ l}
~prod from {i^=^1} to p ~ paren { pile {n sub i above l sub i} }
fwd 250 ""
.EN -
.EQ
lineup
cdot bracket { (n - l) - (n sub i - l sub i ) }
sup {n sub i - l sub i}
cdot bracket { (n - l) sup 2 - sum from {j^=^1} to p
(n sub i - l sub i ) sup 2 }
.EN
```

The "**-**" argument to **EN** says not to number this line. Since the mark/lineup mechanism doesn't look ahead, only the first line will be centered. The "**fwd 250 ""**" is a simple hack to compensate for the greater length of the second line; its numeric value is determined by trial and error.

[3] By the way, footnotes may also appear within floats.

Here's another nice example:

$$
\begin{aligned}
f_{h,\varepsilon}(x,y) &= \varepsilon \mathbf{E}_{x,y} \int_0^{t_\varepsilon} \mathrm{L}_{x,y_\varepsilon(\varepsilon u)} \phi(x)\, \mathrm{d}u && (6.3)\\
&= \hbar \int \mathrm{L}_{x,z} \phi(x) \rho_x\, (\mathrm{d}z) \\
&\quad + h \left[ \frac{1}{t_\varepsilon} \left( \mathbf{E}_y \int_0^{t_\varepsilon} \mathrm{L}_{x,y^x(s)} \phi(x)\, \mathrm{d}s - t_\varepsilon \int \mathrm{L}_{x,z} \phi(x) \rho_x\, (\mathrm{d}z) \right) \right. \\
&\qquad \left. + \frac{1}{t_\varepsilon} \left( \mathbf{E}_y \int_0^{t_\varepsilon} \mathrm{L}_{x,y^x(s)} \phi(x)\, \mathrm{d}s - \mathbf{E}_{x,y} \int_0^{t_\varepsilon} \mathrm{L}_{x,y_\varepsilon(\varepsilon s)} \phi(x)\, \mathrm{d}s \right) \right] && (6.4)\\
&= h \hat{\mathrm{L}}_x \phi(x) + h \theta_\varepsilon(x,y) && (6.5)
\end{aligned}
$$

Since GNU eqn and troff make it easy to embed device code in the math, we can use this feature with PostScript to draw arbitrarily scalable delimiters that look much nicer (right hand side of following equation) than the ones built up from pieces in the standard Adobe symbol font (left side):

$$
\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \tag{6.6}
$$

(But they're not hinted, so they look significantly better on high-resolution devices.)

# 6. 3. Tables

Tables are created with the usual **TS**/**TE** macros in conjunction with **tbl**. It is suggested to put tables inside a float or keep, as the **TS** and **TE** macros, which can in principle be implemented to offer a keep/release functionality, are here implemented as no-ops. This is because we often want a table to be accompanied by a caption which must not be separated from the actual table, and since this requires a keep (or float) enclosing both table and caption, any such functionality in the **TS**/**TE** macros would be redundant.

Here is a sample table:

**TABLE 6-1.** — Sample table. (Based on one of the many examples given in *Tbl — A Program to Format Tables* by L. L. Cherry and M. E. Lesk.)

| **Composition of Foods** | | | |
|---|---|---|---|
| Food | Percent by Weight | | |
| | Protein | Fat | Carbo-hydrate |
| Apples | .4 | .5 | 13.0 |
| Halibut | 18.4 | 5.2 | . . . |
| Lima beans | 7.5 | .8 | 22.0 |
| Milk | 3.3 | 4.0 | 5.0 |
| Mushrooms | 3.5 | .4 | 6.0 |
| Rye bread | 9.0 | .6 | 52.7 |

# 7

# Footnotes[4], cross references, and indexing

## 7.1. The footnote macros, FN and FE

Footnotes[5] are embedded in the text via

```
This is some text\c
.FN
And this is a footnote.
.FE .
```

which produces

This is some text[6].

in the normal text, and a corresponding footnote at the bottom. If **FN** is given an argument, then that will be taken as the footnote mark instead of a running number*. The "**\c**" is necessary to attach the footnote mark to the word preceding it in the text. I would have liked to provide an auto-concatenation feature as used in the **SM** and **RF** macros, but that would have been ambiguous, since the footnote mark can also be specified optionally. Concatenation is, however, available for the **FE** macro, as shown in the example above. Here's some more gratuitous text intended to cause the next footnote to overflow[7].

---

[4] Footnotes in section headers are ignored in the table of contents and the page headers.

[5] which, of course, may contain control lines and math: $F_0(x) = \int_0^x f(t)\,dt$.

[6] And this is a footnote.

[*] But it's the user's responsibility to ensure that the mark is unique on the page it occurs on.

[7] An overflowing footnote is one that will not completely fit on the current page because its size is larger than the

In troff, the mark of a footnote immediately following another one will be automatically separated from the preceding mark with a comma. (In nroff, this is handled a bit differently.) For example,

```
Some text\c
.FN
Footnote one.
.FE \c
.FN
Footnote two.
.FE .
```

gives

Some text[8,9].

The test is carried out using the current horizontal and vertical output position. I hope this works under all reasonable circumstances.

## 7.2. The cross-reference macros, LB, RF, and PG

These macros allow referencing sections, equations, tables, and figures, and the pages they occur on, by name. To reference such an entity, it must first be labeled by placing a "**LB** *name*" after the item to be referenced. Then, "**RF** *name*" will insert the number of the corresponding labeled entity (section, equation, etc.), and "**PG** *name*" the page number it occurs on. (A reference to an unknown label will result in "???".) For convenience, the macros **RF** and **PG** may be called with one, two, or three arguments. If called with two or three arguments, the second argument will be taken as the label, and the other argument(s) will be concatenated together with the resolved reference. This is useful, for example, to place referenced equation numbers in parentheses or to add punctuation, as in

```
.RF Equation~( eqn:schrodinger ),
together
.RF with~( eqn:oscillator ),
can be solved using the following ansatz:
```

which will give something like

Equation (1.3), together with (1.17), can be solved using the following ansatz:

(Or you edit the macro **EQ**, and make the reference itself contain the parentheses.)

To enable forward references, the labels are written to two auxiliary files, **troff.crossrefs** and **troff.pagerefs**, which are read at the beginning of processing; the document must be **troff**ed twice for forward references to be resolved. (Within the usual repeated edit/format/view cycle, if only small changes are made to the document, a single **troff** pass will normally suffice, since references are already available from a previous run.)

---

available space to the bottom margin (including already-present footnotes) from the place it was referenced in the normal text.

[8] Footnote one.

[9] Footnote two.

# 8

# Other nifty and/or useful macros

## 8.1. The "drop capital" macro, DC

For some nice effects, a "drop capital" can be placed into the text with this macro. It is intended to be used in the first line of a paragraph immediately following a chapter header. (No guarantee is given that it won't screw up if used somewhere else; of course, no guarantee is given that it won't screw up here, either.) Usage is "**DC** *C*", where *C* is the character to be used as the drop capital (using a capital letter is recommended). Don't forget to begin the first word without its initial letter, as that is supposed to be already represented by the drop capital. The size of the drop capital must be tweaked to fit the font used; for this purpose, a size factor (e.g., **43/10**) can be given as an optional second argument, which will be remembered for future invocations of the macro.

Of course, the macro also works with more than one letter (the argument can even contain local motion), which is quite useful if the drop capital must include punctuation, as in

*'Twas brillig, and the slithy toves*
*Did gyre and gimble in the wabe:*
*All mimsy were the borogoves,*
*And the mome raths outgrabe.*

*"Beware the Jabberwock, my son!*
*The jaws that bite, the claws that catch!*
*Beware the Jubjub bird, and shun*
*The frumious Bandersnatch."*

*— Lewis Carroll*

Because **DC** uses **in** and the previous indent value internally, you can't rely on the previous value to undo the indent after the poem. Use an explicit exdent, or use **IN**.

# A

# List of used register and macro/string names

The following naming convention is used in this macro package: User-callable macros and user-settable registers and strings have names consisting of one or two upper-case letters. (Raw troff requests have names with lower-case letters only.) Names of internally used macros and registers contain at least one non-alphabetic character. Names consisting of combinations of lower-case and upper-case letters are not used in this package and are available for the user's own purposes.

**fonts**

registers:

**FS** — global default font size
**VS** — global default baseline spacing

macros and strings:

**SI** — user font size macro
**FF** — user font family selection macro
**R,I,B,C** — user font face style selection macros
**R[IBC], I[RBC], B[RIC], C[RIB]** — user two-font alternating macros
**EM** — user "emphasize" macro
**SM** — user "smaller" macro
**FS** — user global default font size setting macro
**VS** — user global default baseline spacing setting macro
**&?** — requested font family name string
**&!** — current font family name string
**&~** — generic two-font alternating macro
**&=** — font family registration/resolution macro

**&/** — discretionary italic correction string
**&,** — discretionary left italic correction string
**&^** — alternate font for emphasize macro
**&1** — two-font switching font 1
**&2** — two-font switching font 2

**page layout**

registers:

**HY** — default hyphenation mode
**PI** — paragraph indent
**PD** — paragraph spacing
**DD** — display spacing
**PO** — default page offset for odd pages
**PE** — default page offset for even pages
**LL** — default line length
**TM** — top margin
**BM** — bottom margin
**TH** — text height
**PL** — page length
**%-** — no stretch flag
**%?** — accumulated stretchable space
**%!** — available space for stretching
**%_** — available space for footnotes
**%%** — temporary computation register
**?!** — debugging flag

macros and strings:

**PI** — paragraph indent setting macro
**PD** — paragraph spacing setting macro
**DD** — display spacing setting macro
**PO** — default page offset setting macro (for odd and even pages)
**PE** — default page offset setting macro (for even pages only)
**LL** — default line length setting macro
**TM** — top margin setting macro
**BM** — bottom margin setting macro
**TH** — text height setting macro
**PL** — page length setting macro
**SP** — user (stretchable) space macro
**NS** — user “no stretch on this page” macro
**%=** — page diversion
**%?** — stretchable space macro
**%:** — auxiliary macro for SP
**?=** — stretchable-space debugging macro
**?-** — stretchable-space debugging macro

**page header and footer**

registers:

**PY** — default page style
**PX** — current page style
**%+** — first page flag

macros and strings:

**BP** — user begin page macro
**HD** — user page header macro
**FT** — user page footer macro
**NH** — user “no page header” request macro
**LH** — header string for left (even) pages (set by chapter macro)

**RH** — header string for right (odd) pages (set by section macros)
**PY** — user default page style setting macro
**PX** — user current page style setting macro
**%<** — page initialization macro
**%>** — page finalization macro

**footnotes**

registers:

**_#** — global footnote counter (only for numbered footnotes)
**_%** — per-page footnote counter
**_^** — height of collected footnotes
**_&** — overflowed footnote indicator flag
**___** — horizontal output position at most recent footnote mark
**_|** — vertical output position at most recent footnote mark

macros and strings:

**FN** — user footnote begin macro
**FE** — user footnote end macro
**_=** — footnote gathering diversion
**_-** — footnote reprocessing diversion
**_!** — footnote overflow trap macro
**_+** — footnote overflow gathering diversion
**_&** — overflowed footnote reprocessing macro
**_@** — footnote mark generating macro
**_$** — footnote mark printing macro
**_[** — internal footnote begin macro
**_]** — internal footnote end macro
**_(** — internal footnote begin macro
**_)** — internal footnote end macro
**_*** — current footnote mark string
**_,** — footnote mark separator string
**_<** — footnote mark begin string
**_>** — footnote mark end string

**indents**

registers:

**>+** — current relative indent (used by list item macro)

macros and strings:

**IN** — user indent/exdent macro
**>=** — absolute indent depth stack string
**>+** — relative indent depth stack string
**>>** — indent macro
**><** — exdent macro (absolute)
**>-** — exdent macro (relative)

**other stuff**

registers:

**>#** — fill mode flag for DC

macros and strings:

**HR** — horizontal rule macro
**DC** — drop capital macro
**>@** — temporary diversion for DC
**>&** — temporary diversion for DC
**>|** — restore indent macro for DC
**>*** — drop capital character size factor

**lists**

registers:

***#** — list nest depth level
***[1-6]** — list counters
***=** — current list type
***_** — last list-related horizontal output position
***|** — last list-related vertical output position

macros and strings:

**BL** — user bulleted list macro
**NL** — user numbered list macro
**TL** — user tagged list macro
**LE** — user list end macro
**LI** — user list item macro
****** — current item bullet string
***[1-6]** — list bullet strings
***=** — list type stack string
***-** — list type stack auxiliary macro
***+** — vertical spacing macro

**cross references**

registers:

(none)

macros and strings:

**LB** — user item labelling macro
**RF** — user item reference resolution macro
**PG** — user page reference resolution macro
**@+** — label saving macro
**@(** — backward page reference resolution macro (from current run)
**@)** — forward page reference resolution macro (read from file)
**@[** — backward item reference resolution macro (from current run)
**@]** — forward item reference resolution macro (read from file)
**@:** — reference comparison macro
**@=** — temporary labeled-item string (set by section headers, etc.)
**@?** — temporary label-to-be-resolved string (used by PG and RF)
**@!** — resolved reference (returned by @[, @], @(, or @))

**floats and keeps**

registers:

**~F** — figure counter
**~T** — table counter
**~#** — pending float counter
**~+** — float counter
**~%** — per-page float counter
**~-** — float pre-space
**~=** — float-only page flag
**~@** — suppress caption dash flag
**&f** — eqn font number
**&F** — eqn previous font number
**&s** — eqn font size
**&S** — eqn previous font size

macros and strings:

**BF** — user begin float macro
**EF** — user end float macro
**BK** — user begin keep macro

**EK** — user end keep macro
**FC** — user figure caption macro
**TC** — user table caption macro
**EC** — user end caption macro
**TS** — table start macro (dummy, only for tbl)
**TE** — table end macro (dummy, only for tbl)
**T&** — table continue macro (dummy, only for tbl)
**NE** — "need space" macro taking floats on next page into account
**FL** — output all pending floats macro (called by chapter)
**~@** — optional caption supplement string
**~_** — temporary caption gathering macro
**~|** — caption size testing diversion
**~-** — temporary float/keep gathering macro
**~=** — float gathering macro
**~+** — overflow float gathering macro
**~&** — float processing macro, called by page header
**~?** — available-space testing macro
**~!** — float/keep size testing diversion
**~)** — dummy marker for end of overflowed float
**~]** — dummy marker for end of appended float
**~}** — float/keep ending macro name
**~%** — NE internal macro

**sections**

registers:

**#[1-4]** — section counters
**$[1-4]** — mirrored section counters
**#=** — section header processing type
**FI** — default paragraph fill mode

macros and strings:

**PP** — user begin paragraph macro
**CH** — user begin chapter macro
**SE** — user begin section macro
**SU** — user begin subsection macro
**SS** — user begin subsubsection macro
**AX** — user begin appendix macro
**TP** — user begin title page macro
**AS** — user begin abstract macro
**CT** — user table of contents macro
**IX** — user index macro (not yet implemented)
**LF** — user list of figures macro (not yet implemented)
**LT** — user list of tables macro (not yet implemented)
**NI** — user "don't indent next paragraph" macro
**NF** — user "don't fill paragraphs" macro
**FI** — user "fill paragraphs" macro
**$=** — section counters mirroring macro
**##** — process section headers macro
**#=** — section headers gathering macro
**#!** — temporary section header processing diversion
**#?** — temporary section header processing diversion
**#*** — previous header finishing macro name string
**#[** — begin section processing macro
**#]** — finish section processing macro
**#{** — begin table of contents entry processing macro
**#}** — finish table of contents entry processing macro

**#@** — table of contents gathering diversion
**#[1-4]** — section number strings for headers
**$[1-4]** — section number strings for references
**[[1-4]** — begin section header processing macros
**][1-4]** — finish section header processing macros
**([1-4]** — begin page header entry processing macros
**)[1-4]** — finish page header entry processing macros
**{[1-4]** — begin table of contents entry processing macros
**}[1-4]** — finish table of contents entry processing macros

**equations**

registers:

**=#** — equation counter
**=*** — temporary equation number
**=+** — length of last output text line
**=^** — line length for displayed equation

macros and strings:

**EQ** — user begin displayed equation macro
**EN** — user end displayed equation macro
**=*** — temporary equation number string
**==** — equation size testing diversion and math string

# B

# Utility scripts

## B. 1. Resorting pages

The following **sed** script (saved to a file "**resort.sed**") is used to resort the pages, i.e., to move the front matter, which is actually formatted at the end, to the beginning. It operates on the intermediate troff output before it is fed to the device postprocessor (see the "driver" script in the next section), the rationale being that manipulating the device-independent troff output, which has a simple structure, is much easier than manipulating the final device code. Of course, the script relies on the document satisfying a number of conditions, in particular that page number 1 occurs exactly twice: once in the regular text and once again in the front matter (title page).

```
#!/bin/sed -f

/^p1$/s/p1/# -- cut here --/
/^# -- cut here --$/,/^# -- cut here --$/w troff.sort
/^# -- cut here --$/,/^# -- cut here --$/c\
p1
/^x trailer$/d
/^x stop$/{
s/.*/p1/
r troff.sort
a\
x trailer\
x stop
}
```

## B. 2. Running groff

Save the following Bourne shell script as "**roff**". Then,

```
:map <F1> :w<CR>:!roff %<CR><CR>
```

in **vim** will allow formatting and viewing of the document at the press of a single button (F1). Instead of loading the macros with the troff **-m** command line flag, the macro file should be **so**urced from the document, since the macros contain a few math definitions that need to be processed by **eqn**.

```
#!/bin/sh

# set -x
  trap "" 1

  roff=$1
  post=`basename $1 .ro`.ps

  refresh(){
    test -z "$1" && { gv -nowatch $post & } || kill -HUP $1
  }

# -- uncomment below for diagnostic output of intermediate steps --

  soelim $roff |
    pic |
    grep -v '^\\D.Fg' |
    tbl |
    eqn -Tps |
#   tee troff.preproc |
    troff -U -Tps -ww -b |
#   tee troff.postproc |
    sed -f resort.sed |
#   tee troff.device |
    grops >$post
  rm troff.sort

# -- ps output is like "1308677 pts/19  S  0:00.38 gv test.ps" --
# -- first item is PID, used in refresh function as $1 --
# -- [0-9] matches time and is used to exclude the grep process --

  pid=`ps x | grep "[0-9] gv -nowatch $post"`
  refresh $pid
```